GBASE®

GBase 8s 管理员指南

www.gbase.cn

GBase 8s 管理员指南，南大通用数据技术股份有限公司

版权声明

免责声明

通讯方式

南大通用数据技术股份有限公司

天津市高新区开华道22号普天创新产业园东塔20-23层

电话：400-013-9696　　邮箱：info@gbase.cn

商标声明

目　录

1 简介

1.1 本简介内容

本简介概述了本出版物中的信息，并描述了所使用的约定。

这些主题提供了 GBase 8s 管理员所需信息。

本出版物是为以下用户编写的：

- 数据库用户
- 数据库管理员
- 数据库服务器管理员
- 性能工程师
- 以下类别的程序员
    - 应用程序开发者
    - 用户定义的例程的作者

本出版物编写时假设您已具有下列背景：

- 计算机、操作系统和操作系统提供的实用程序的应用知识
- 关系数据库的一定使用经验或对数据库概念有所了解
- 一些计算机编程经验
- 有数据库服务器管理、操作系统管理或网络管理方面的相关经验

本出版物在编写时假定您要将 GBase 8s 用作数据库服务器。

这些主题来自《GBase 8s 管理员指南》。

1.2 关于本出版物

本出版物描述了有关配置、管理和使用 GBase 8s 的概念与过程。

1.2.1 用户类型

本出版物是为以下用户编写的：

- 数据库用户
- 数据库管理员
- 数据库服务器管理员
- 性能工程师
- 以下类别的程序员
    - 应用程序开发者
    - 用户定义的例程的作者

本出版物编写时假设您已具有下列背景：

- 计算机、操作系统和操作系统提供的实用程序的应用知识

- 关系数据库的一定使用经验或对数据库概念有所了解
- 一些计算机编程经验
- 有数据库服务器管理、操作系统管理或网络管理方面的相关经验

1.2.2 软件依赖性

本出版物在编写时假定您要将GBase 8s V8.8 用作数据库服务器。

1.2.3 关于语言环境的假设

GBase 8s 产品可以支持多种语言、文化和代码集。由给定地域和编码中语言使用的与数字数据、货币、日期和时间的字符集、整理和表示法相关的所有信息都集中在称为 Global Language Support (GLS) 语言环境的单个环境中。

GBase 8s OLE DB Provider 遵循用于日期、时间和货币的 ISO 字符串格式，如 Microsoft™ OLE DB 标准所定义。可以通过设置 GBase 8s 环境变量或注册表项（如 DBDATE）来覆盖该缺省值。

本出版物中的示例在编写时假设您使用以下某种语言环境：UNIX™ 平台上的 en_us.8859-1 (ISO 8859-1) 。这些语言环境支持用于显示和输入日期、时间、数字和货币值的美国英语格式约定。它们还支持 ISO 8859-1 代码集（在 UNIX 和 Linux™ 上），这些代码集包括 ASCII 代码集以及很多 8 位字符（如 é、è 和 ñ）。

如果您计划在数据或 SQL 标识中使用其他语言环境中的字符，或者希望符合字符数据的其他整理规则，那么可以指定其他语言环境。

1.2.4 演示数据库

DB-Access 实用程序随 GBase 8s 数据库服务器产品一起提供，它包括一个或多个以下演示数据库：

- stores_demo 数据库以一家虚构的体育用品批发商的有关信息举例说明了关系模式。GBase 8s 出版物中的许多示例均基于 stores_demo 数据库。
- superstores_demo 数据库举例说明了对象关系模式。superstores_demo 数据库包含扩展数据类型、类型和表继承以及用户定义的例程的示例。

有关如何创建和填充演示数据库的信息，请参阅《GBase 8s DB-Access 用户指南》。有关数据库及其内容的描述，请参阅《GBase 8s SQL 指南：参考》。

用于安装演示数据库的脚本位于 UNIX™ 平台上的 $GBASEDBTDIR/bin 目录和 Windows™ 环境中的 %GBASEDBTDIR%\bin 目录中。

1.3 示例代码约定

SQL 代码的示例在整个出版物中出现。除非另有说明，代码不特定于任何单个的 GBase 8s 应用程序开发工具。

如果示例中仅列出 SQL 语句，那么它们将不用分号定界。例如：您可能看到以下示例中的代码：

```
CONNECT TO stores_demo
...
DELETE FROM customer
      WHERE customer_num = 121
...
COMMIT WORK
DISCONNECT CURRENT
```

要将此 SQL 代码用于特定产品，必须应用该产品的语法规则。例如，如果使用的是 SQL API，那么必须在每条语句的开头使用 EXEC SQL，并在每条语句的结尾使用分号（或其他合适的定界符）。 如果使用的是 DB－Access，那么必须用分号将多条语句隔开。

提示： 代码示例中的省略点表示在整个应用程序中将添加更多的代码，但是不必显示它以描述正在讨论的概念。

有关使用特定应用程序开发工具或 SQL API 的 SQL 语句的详细指导，请参阅您的产品文档。

1.4 其他文档

有关此发行版的产品的文档以各种格式提供。

您可以从产品随附的快速入门指南 CD 来访问或安装产品文档。

1.5 语法图

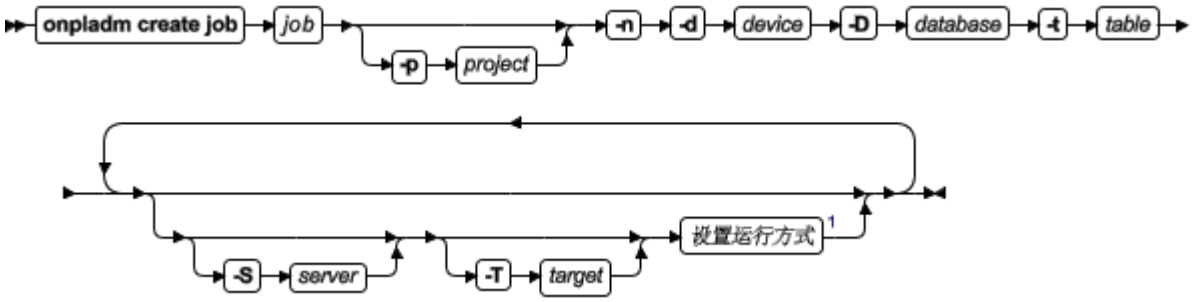
语法图使用特殊组件描述语句和命令的语法。

1.5.1 如何阅读命令行语法图

命令行语法图使用类似于其他语法图元素的元素。

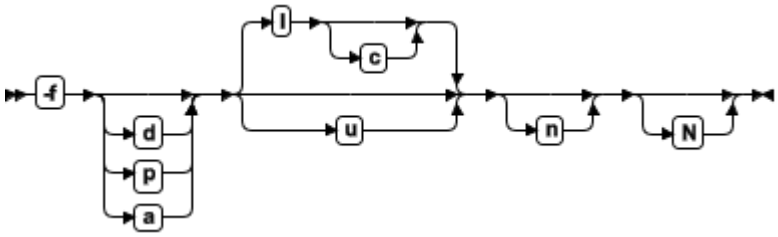
某些元素列于语法图中的表中。

创建非转换作业



此图中有一个名为“设置运行方式”的段，根据图脚注，这个段在第 Z-1 页上。如果这是真正的交叉引用，那么您可以在附录 Z 的第一页上找到此段。但在此处，此段显示在以下段图表中。请注意：该图使用段开头和结束部分。

设置运行方式



要了解如何正确构造命令，请从主图的左上角开始。遵循右边的图表，包括想要的元素。此图中的元素区分大小写，因为它们说明实用程序的语法。其他类型的语法（例如 SQL）则不区分大小写。

“创建非转换作业”图表说明了以下步骤：

1. 输入 onpladm create job，然后输入作业的名称。
2. 或者，输入 -p，然后输入项目的名称。
3. 输入以下所需的元素：
   - -n
   - -d 和设备的名称
   - -D 和数据库的名称
   - -t 和表的名称
4. 或者，可以选择一个或多个以下元素并重复它们任意次：
   - -S 和服务器名称
   - -T 和目标服务器名称
   - 运行方式。要设置运行方式，请遵循“设置运行方式”段图表来输入 -f，或者输入 d、p 或 a，然后可选择输入 l 或 u。
5. 遵循图表直至终止符。

1.5.2 关键字和标点

关键字是为语句和除了系统级别命令的所有命令保留的词语。

当关键字出现在语法图表中时，它以大写字母显示。在命令中使用关键字时，可用大写或小写字母写关键字，但是必须严格按照语法图表中所显示的来拼写关键字。

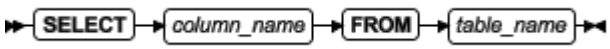
还必须严格按照语法图表中所显示的在语句和命令中使用标点。

1.5.3 标识和名称

变量作为语法图表和示例中标识符和名称的占位符。

根据上下文，可用任意名称、标识符或文字替换变量。变量也用来代表附加语法图表中扩展的复杂语法元素。当变量出现在语法图表、示例或文本中时，它以斜体小写字母显示。

下列语法图使用变量来说明简单 SELECT 语句的一般格式。



当编写此格式的 SELECT 语句时，请使用特定的列和表名称来替换 column_name 和 table_name 变量。

2 数据库服务器

2.1 数据库服务器的安装和配置

安装和初始配置数据库服务器时，有各种选项可供选择。您还可以安装OpenAdmin Tool (OAT) 以从单个位置监视和管理各个数据库服务器实例。

安装 GBase 8s 时，请遵循安装指示信息以确保满足所有先决条件（例如，所有关键文件和目录的许可权均已正确设置）。

安装完数据库服务器的新版本后，您必须对其进行配置。配置是特定于设置的参数，用于针对数据处理环境定制数据库服务器：数据量、表的数量、数据类型、硬件、用户数以及安全性需求。

配置参数已更改或已从数据库服务器中除去。

2.1.1 规划数据库服务器

配置数据库管理系统时需要作出许多决定，例如存储数据的位置，访问数据的方法以及如何保护数据。如何安装和配置 GBase 8s 会在很大程度上影响数据库运行的性能。

可以定制数据库服务器使其能在特定的数据处理环境中以最佳方式运行。例如，将数据库服务器用于为 1000 位执行频繁短事务的用户提供服务与用于为一些执行长时间复杂搜索的用户提供服务有所不同。

当您在规划数据库服务器时，请考虑优先级和环境。

优先级注意事项

当您在准备初始配置和规划备份策略时，请记住数据库服务器的特征：

- 其他计算机上的应用程序是否使用此数据库服务器实例？
- 您可以期望的最大用户数是多少？
- 您希望在多大程度上控制用户环境？
- 您是否在空间、CPU 或者可用的操作程序方面受到限制？
- 需要数据库服务器在不受监视的环境下运行吗？
- 数据库服务器是经常处理许多短事务还是处理较少的长事务？
- 您计划使用哪些新的数据库服务器功能部件或相关产品？

环境注意事项

在开始对数据库服务器进行初始配置之前，必须获取有关环境中计算机的信息。

尽可能多地收集以下信息。要获得这些信息，您可能需要系统管理员的帮助：

- 网络上其他计算机的主机名和 IP 地址
- 您的 UNIX™ 平台是否支持网络信息服务 (NIS)？
- 磁盘控制器配置
- 有多少磁盘驱动器？其中一些磁盘驱动器是否比另一些磁盘驱动器快？
- 有多少磁盘控制器？磁盘控制器的配置如何？
- 是否要升级硬件和操作系统？您的操作系统是 32 位的还是 64 位的？
- 操作系统共享内存和其他资源
  有多少共享内存？其中有多少可以用于数据库服务器？

仅限 UNIX：计算机的 notes 文件指出每种 UNIX 平台的适用参数。

2.1.2 配置操作系统

在可以开始配置数据库服务器之前，必须适当地配置操作系统。为此，您可能需要系统管理员的帮助。

GBase 8s 的 32 位版本可以运行在 64 位或 32 位操作系统上。GBase 8s 的 64 位版本必须运行在 64 位操作系统上。有关更多信息，请参阅 64 位平台上的内存使用。

修改 UNIX 内核参数

机器的 notes 文件包含用于配置操作系统资源的推荐值。可以在配置操作系统时使用这些推荐值。

如果数据库服务器的推荐值与当前环境之间存在很大的不同，请考虑修改操作系统配置。

在一些操作系统上，您可以指定分配给数据库服务器的共享内存的数量。可用内存的数量将影响那些可以为配置文件中的共享内存参数选择的值。通常，增加共享内存的可用空间能够提高性能。您可能还需要指定锁和信号数。

2.1.3 配置磁盘空间

配置磁盘以便在数据集市和数据仓库中获得最佳性能。对于 SQL 操作，磁盘 I/O 所占的响应时间最长。数据库服务器提供对计算机上的多个磁盘的并行访问权。分配磁盘空间之前，请了解操作系统管理指南中关于磁盘空间的信息并参阅块的磁盘分配。

使用大块

缺省情况下，对于 2 KB 页，数据库空间的块大小是 4 太字节。块可以位于 64 位地址空间中的任何位置。

在 UNIX 上创建块文件

在 UNIX™ 上，可以将数据存储在使用未缓冲（原始）磁盘或操作系统文件（又称为已缓冲或熟文件）的块中。

原始或未缓冲的磁盘存取

UNIX 使用字符专用设备（又称为原始磁盘设备）提供未缓冲的磁盘存取。要在 UNIX 上创建原始磁盘设备，请遵循操作系统中提供的指示信息。

数据库服务器使用原始磁盘存取以提高磁盘 I/O 操作的速度和可靠性。原始磁盘存取绕过操作系统提供的文件缓冲机制。数据库服务器本身将管理磁盘和内存之间的数据传送。数据库服务器通过保证连续存储行来优化表的访问。

重要： 虽然必须在 UNIX 上使用原始磁盘设备来获得更好的性能，但是在用于熟写入的 I/O 高速缓存方面的最新进步也可以提供相似的性能（即使不是最好的性能）。要确定最佳设备性能，请在同时带有数据库空间和表布局两类设备的系统上执行基准测试。

要为数据库服务器分配磁盘，请执行以下操作：

1. 为每个磁盘配置原始磁盘设备。
2. 创建标准的设备名或文件名。
3. 为每个原始磁盘设备设置权限、所有权和组。

熟文件

如果不要求最佳性能，那么可以将数据库服务器配置为在熟文件中存储数据。熟文件比原始磁盘设备更容易设置。

设置许可权、所有权和组

数据库服务器使用的文件或原始磁盘设备必须有适当的所有权和权限。

仅限 UNIX： 在 UNIX™ 上，所有者和组必须是 gbasedbt，并且这些权限必须设置为允许用户和组（但不包括其他）进行读写。

如果希望非 gbasedbt或 root 的用户运行 ON-Bar 命令，请创 bargroup 组。只有 bargroup 的成员可以运行 ON-Bar 命令。bargroup 组不会在数据库服务器安装期间自动创建。

创建标准设备名 (UNIX)

使用符号链接为每个原始磁盘设备指定缩写的标准设备名。如果拥有符号链接，那么可以使用新磁盘通过为该新磁盘指定符号名来替换发生故障的磁盘。

要在字符专用设备名和另一个文件名之间创建链接，请使用 UNIX™ 链接命令（通常为 ln）。

在设备目录上运行 UNIX 命令 ls -l 以便验证这些设备和链接都已存在。以下示例显示到原始设备的链接。如果您的操作系统不支持符号链接，那么可以使用硬链接。

```
ls -l
crw-rw---    /dev/rxy0h
crw-rw---    /dev/rxy0a
lrwxrwxrwx /dev/my_root@->/dev/rxy0h
lrwxrwxrwx /dev/raw_dev2@->/dev/rxy0a
```

2.1.4 设置环境变量

要启动、停止或访问数据库服务器，每个用户都必须具有必要的数据库访问特权，且必须设置适当的环境变量。一些环境变量是必需的，其他则是可选的。

需要的环境变量

《GBase 8s SQL 指南：参考》包含环境变量的完整列表。

下表显示了必须在访问数据库服务器或执行大多数管理任务前设置的环境变量。

表 1. 需要的环境变量

环境变量	描述
CLASSPATH	如果您使用的是 J/Foundation，那么指定 jvphome/krakatoa.jar 文件的位置，以便 Java Development Kit (JDK) 可以编译 Java™ 源文件。
GBASEDBTDIR	指定安装 GBase 8s 数据库服务器的目录。

环境变量	描述
GBASEDBTSERVER	指定缺省数据库服务器的名称。 它具有为 DBSERVERNAME 或 DBSERVERALIASES 配置参数指定的值。
JVPHOME	如果您正在使用 J/Foundation，请指定装有 GBase 8s JDBC Driver 的目录。
ONCONFIG	指定活动 onconfig 文件的名称。 所有使用数据库服务器实用程序（如 gstat）的用户都必须设置 ONCONFIG 环境变量。运行客户机应用程序的用户无需设置 ONCONFIG 环境变量。 如果 ONCONFIG 环境变量不存在，数据库服务器将使用 onconfig 文件中的配置值： 在 UNIX™ 上：$GBASEDBTDIR/etc/onconfig
PATH	指定可执行文件的位置。 在 UNIX™ 上：$GBASEDBTDIR/bin
TERM	使 DB-Access 能够识别您正在使用的终端并与其通信。此环境变量无需初始化或启动，但首先必须对其进行设置才可以运行应用程序。
TERMCAP TERMINFO GBASEDBTTERM	指定 DB-Access 是使用 termcap 文件还是 terminfo 目录中的信息。如果系统要求，可能需要获得 UNIX 系统管理员的帮助，才能设置这些变量，这是因为这些变量高度依赖系统。

设置环境变量

提示：在 shell 文件中设置环境变量。

可在配置文件中包含环境变量 $GBASEDBTDIR。该环境变量必须为路径名规范中的第一个路径名值。

要设置必需的环境变量，请执行以下操作：

1. 将 GBASEDBTDIR 设置为安装 GBase 8s 产品的目录。
2. 将 PATH 环境变量设置为包含 $GBASEDBTDIR/bin (UNIX™) 。
3. 将 GBASEDBTSERVER 设置为数据库服务器的名称。

设置 GLS 环境变量

如果要使用非缺省语言（美式英语），请设置 Global Language Support (GLS) 环境变量。

使用以下环境变量处理 GLS：

- CLIENT_LOCALE
- DB_LOCALE
- SERVER_LOCALE

- DBLANG
- C8BITLEVEL
- ESQLMF
- GLS8BITFSYS
- GL_DATE
- GL_DATETIME

如果计划使用 Unicode，请设置 GL_USEGLU 环境变量以提高对 UTF-8 编码的支持。

在 UNIX 上设置环境变量

用以下某种方法设置 UNIX™ 环境变量：

- 在 onconfig 文件中设置本地环境变量

  在通过 -FILE 选项启动以下实用程序时，这些实用程序将使用这些设置：oninit、onclean、gload、glogdump、gadmin、glogadmin、gspaces、gstat、gtape和 gunload。在运行实用程序时，通过 -FILE 选项指定 onconfig 文件的路径。使用该选项，可轻松在嵌入式环境中的远程计算机上运行实用程序。

- 在命令行上的系统提示符处

  如果在系统提示符处设置某个环境变量，必须在下一次登录到系统时重新指定该变量。

- 在环境配置文件（例如 $GBASEDBTDIR/etc/gbasedbt.rc 或.gbasedbt）中

  环境配置文件是一种公共或专用文件，您可以在该文件中为每个数据库服务器用户设置环境变量。使用环境配置文件将减少必须在命令行或外壳程序文件中设置的环境变量数目。

- 在 .profile 或 .login 文件中

  如果在 .login、.cshrc 或 .profile 文件中设置某个环境变量，每次登录到系统时都将自动指定该变量。有关这些文件的信息，请参阅您的操作系统手册。

  要覆盖自动设置的环境变量，请使用专用环境变量文件 ~/.gbasedbt，或单独为环境变量指定新的值。

要检查环境变量的有效性，请使用 chkenv 实用程序。

以下示例显示包含 **miami** 数据库服务器的环境变量的设置文件。LD_LIBRARY_PATH 设置为数据库服务器和 GBase 8s ESQL/C 库文件的位置。

```
setenv GBASEDBTDIR /ix/gbasedbt93
setenv GBASEDBTSQLHOSTS /ix/sqlhosts.unified
setenv ONCONFIG s.miami
setenv GBASEDBTSERVER miami


# setup to use J/Foundation
setenv JVPHOME /ix/gbasedbt93/extend/krakatoa
```

```
setenv CLASSPATH
$JVPHOME/krakatoa.jar:$JVPHOME/jdbc.jar:/usr/java/lib/classes.zip


# Include jar paths for Java; include /usr/ccs/bin for C compiler:
setenv PATH $GBASEDBTDIR/bin:$GBASEDBTDIR/extend/krakatoa/krakatoa.jar:
    $GBASEDBTDIR/extend/krakatoa/jdbc.jar:/usr/ccs/bin:$PATH


setenv LD_LIBRARY_PATH $GBASEDBTDIR/lib:$GBASEDBTDIR/lib/esql:/usr/lib
```

2.1.5 配置连接

连接信息允许客户机应用程序连接到网络上的任何 GBase 8s 数据库服务器。即使客户机应用程序和数据库服务器在同一计算机或节点上，您也必须准备连接信息。

某特定数据库服务器的连接数据包括数据库服务器名称、客户机可用来连接到该服务器的连接类型、运行数据库服务器的计算机或节点的主机名以及通过其可知道该数据库服务器的服务名称。

在启动数据库服务器之前，不必在 sqlhosts 信息中指定所有可能的网络连接。 但是，要使新连接可用，必须使数据库服务器脱机，然后使其重新联机。

UNIX： 在配置连接时，还要考虑设置 LISTEN_TIMEOUT 和 MAX_INCOMPLETE_CONNECTIONS 配置参数。通过使用这些参数，处理连接的侦听器 VP 更不容易因连接过多而发生问题，从而能够降低受到恶意的拒绝服务 (DOS) 攻击的风险。

sqlhosts 文件

在 UNIX™ 上，sqlhosts 文件包含连接信息。sqlhosts 文件包含在 UNIX 操作系统上运行的数据库服务器和客户机以及在 Windows™ 操作系统上运行的数据库服务器的连接信息。

文件的缺省位置是 $GBASEDBTDIR/etc/sqlhosts。

文件的缺省位置是：

UNIX：

$GBASEDBTDIR/etc/sqlhosts

Windows：

%GBASEDBTDIR%\etc\sqlhosts.%GBASEDBTSERVER%

如果将信息存储在其他位置，那么必须设置 GBASEDBTSQLHOSTS 环境变量。

如果将多个数据库服务器设置为使用分布式查询，请使用下面的某种方法来存储所有数据库的 sqlhosts 信息：

- 在一个 GBASEDBTSQLHOSTS 所指向的 sqlhosts 文件中
- 存储在多个 sqlhosts 文件中，每个文件位于每个数据库服务器目录下

网络配置文件

除了 sqlhosts 文件之外，因特网协议网络连接还需要 /etc/hosts 和 /etc/services 文件中的条目。

网络安全性文件

GBase 8s 数据库服务器遵循 UNIX 的有关建立连接的安全性要求。因此，UNIX 系统管理员可能需要修改 /etc/passwd、/etc/hosts、~/.rhosts 和其他相关文件。

网络配置和网络安全性文件将在操作系统手册中描述。

2.1.6 配置数据库服务器

通过设置配置参数，可以为数据处理环境定制数据库服务器。数据库服务器的配置参数存储在配置文件中。

创建 GBase 8s 数据库服务器实例时，将自动使用基于 config.std 文件的缺省值来创建对应的配置文件。新的配置文件名为 onconfig.gbasedbtservername，缺省情况下会在 %GBASEDBTDIR%\etc\ 目录中创建此文件。如果将配置文件移至其他目录，请将 ONCONFIG 环境变量设置为此文件所在位置。

可以编辑配置文件来修改配置参数值，以提高实例的性能和其他特性。如果在配置文件中省略某个参数值，数据库服务器将使用 onconfig.std 文件中的缺省值，或基于其他参数值来计算相应值。

onconfig.std 文件是用于创建 onconfig.gbasedbtservername 文件的模板。 不要修改或删除 onconfig.std 文件。

通过编辑 onconfig 文件，可以修改配置参数值。 数据库服务器下次关闭并重新启动之后，更改将生效。但是，不便重新启动数据库服务器时，也可以通过其他方式来更改某些配置参数的值：

- 数据库服务器正在运行时，使用 gadmin -wf 命令可永久性更新许多配置参数。
- 使用 gadmin -wm 命令可更新当前会话的许多配置参数值。

可以成组导出、导入和修改配置参数：

- 使用 gadmin －we 命令可将当前配置的快照导出到文件。然后，生成的快照可以进行归档，用作配置文件，或导入到另一个正在运行的实例。

- 使用 gadmin –wi 命令可从先前导出的文件导入可调整的配置参数。文件中不可动态调整的配置参数将被忽略。

您还可以通过 SQL 管理 API 命令来修改、重置、导出和导入配置文件：

- 将 modify config 自变量用于 admin() 或 task() 函数可更改配置参数的值。
- 将 export config 和 import config 自变量用于 admin() 或 task() 函数可导出或导入包含一个或多个可动态调整的配置参数的文件。
- 将 reset config 或 reset config all 自变量用于 admin() 或 task() 函数可将一个配置参数或全部配置参数的值还原为 onconfig 文件中的相应值。

通过一个配置参数 AUTO_TUNE，可以启用或禁用 onconfig 文件中不包含其值的所有自动调整配置参数。自动调整配置参数为 AUTO_AIOVPS、AUTO_CKPTS、AUTO_LRU_TUNING、AUTO_READAHEAD、 AUTO_REPREPARE 和 AUTO_STAT_MODE。

此外，还可以从 onconfig 文件中除去任何自动调整配置参数，然后使用 AUTO_TUNE 配置参数来设置已除去的配置参数的缺省值。例如，如果 AUTO_AIOVPS 和 AUTO_CKPTS 配置参数在配置文件中不存在，那么数据库服务器会自动将 AUTO_AIOVPS 和 AUTO_CKPTS 的值设置为 AUTO_TUNE 的值。 可以动态修改 AUTO_TUNE 配置参数。

联机日志会标识动态调整的配置参数。

可以使用 OpenAdmin Tool (OAT) 来监视配置。

准备 onconfig 配置文件

GBASEDBTDIR 的 etc 子目录中的 onconfig.std 模板文件包含许多配置参数的初始值。您可以复制模板并保存副本，这样就可以根据您的特定配置来定制副本。

这些模板文件包含许多配置参数的初始值。

重要：请勿修改或删除模板文件。数据库服务器将这些文件作为模板而不是作为功能性配置文件提供。

可以更改的功能配置文件是 onconfig 文件（而不是 onconfig.std 模板文件）。可以使用文本编辑器来更改 onconfig 文件中的配置参数。

如果在 onconfig 文件的副本中省略参数，那么数据库服务器将在服务器启动时使用 onconfig.std 文件中的值代替缺少的参数。

提示： 如果不希望直接处理所有 onconfig 文件参数，genoncfg 实用程序可加快根据您的硬件以及根据数据库服务器的预期用途对配置文件进行定制的过程。

在 UNIX™ 上创建 onconfig 文件

当您首次安装 GBase 8s 软件时，将创建并初始化数据库服务器的新实例。安装脚本将自动为您创建 onconfig.demo 文件。

要使用文本编辑器准备 onconfig 文件，请执行以下操作：

1. 复制和重命名 $GBASEDBTDIR/etc/onconfig.std 文件，并将其存储到etc 子目

录中。

2. 使用文本编辑器编辑 onconfig 配置文件。
3. 将 ONCONFIG 环境变量设置为新的 onconfig 文件的名称。
4. 如果它是新的实例，那么初始化数据库服务器。 否则，关闭并重新启动数据库服务器。

查看有关配置参数的信息

可以查看配置文件的内容，配置参数及其当前值的列表，以及有关可使用gstat 命令调整的配置参数的信息。您还可以使用 OpenAdmin Tool (OAT)查看配置文件的内容。您可以使用 gstat 命令或 OpenAdmin Tool (OAT)查看配置文件的内容。

要查看配置文件的内容,请运行 gstat -c 命令。如果在服务器运行期间更改了配置参数，而且未关闭并重新启动数据库服务器，那么生效的配置将不同于 gstat -c 选项所显示的内容。

要查看配置参数及其当前值的列表，请运行 gstat -g cfg 命令。

2.1.7 启动和管理数据库服务器

在安装并配置了数据库服务器之后，必须执行以下一个或多个任务：

- 准备连接到应用程序。
- 启动数据库服务器并初始化磁盘空间。
- 创建存储空间。
- 设置备份并复原系统。
- 执行管理任务。

启动数据库服务器

使用 oninit 实用程序可启动数据库服务器。可以通过不同的方式启动数据库服务器。缺省方式是联机方式，它允许多个用户连接到数据库服务器。

先决条件：

- UNIX™、Linux™ 或 Mac OS X：您必须以 root 或 gbasedbt 用户身份登录。
- 数据库服务器的磁盘空间已初始化。

要启动数据库服务器：

UNIX、Linux 或 Mac OS X：运行 oninit 命令。可以在 oninit 命令中包含选项来定制启动。

准备自动启动

准备操作系统注册表或脚本以自动启动和停止数据库服务器。

准备 UNIX 启动和关闭脚本

可以修改 UNIX™ 启动脚本以便在计算机进入多用户方式时自动初始化数据库服务器。也可以修改 UNIX 关闭脚本，以便无论 UNIX 何时关机数据库服务器都以受控方式关闭。

ISA 提供了用于启动和关闭的 UNIX 脚本，您可以在 $GBASEDBTDIR/etc/ids-example.rc 中定制该脚本。

准备 UNIX 启动脚本

可以准备 UNIX™ 启动脚本来启动数据库服务器。

要准备 UNIX 启动脚本，请将 UNIX 和数据库服务器实用程序命令添加到 UNIX 启动脚本，以便该脚本执行以下步骤。

要准备 UNIX 启动脚本，请执行以下操作：

1. 将 GBASEDBTDIR 环境变量设置为数据库服务器安装目录的完整路径名。
2. 将 PATH 环境变量设置为包含 $GBASEDBTDIR/bin 目录。
3. 将 ONCONFIG 环境变量设置为适当的配置文件。
4. 设置 GBASEDBTSERVER 环境变量，使 sysmaster 数据库可以更新（或在需要时创建）。
5. 运行 oninit，启动数据库服务器，并使该服务器保持联机方式。

   oninit 实用程序具有 -w 选项，在返回至 shell 提示符并返回代码 0 之前，该选项迫使服务器等待，直到成功初始化为止。有关 oninit 实用程序的信息，请参阅《GBase 8s 管理员参考》中 oninit 实用程序的内容。

   如果您计划初始化数据库服务器的多个版本（多处驻留），那么必须重置 ONCONFIG 和 GBASEDBTSERVER，并为数据库服务器的每个实例重新运行 oninit。

如果数据库服务器的不同版本安装在不同的目录中，您必须复位 GBASEDBTDIR 并为每个不同版本重复前面的步骤。

准备 UNIX 关闭脚本

可以准备在 UNIX™ 每次关闭时都以受控方式关闭数据库服务器。

要在无论 UNIX 何时关闭时都以受控方式关闭数据库服务器，请将 UNIX 和数据库服务器实用程序命令添加到 UNIX 关闭脚本，以便该脚本执行以下步骤。

要准备 UNIX 关闭脚本，请执行以下操作：

1. 将 GBASEDBTDIR 环境变量设置为数据库服务器安装目录的完整路径名。
2. 将 PATH 环境变量设置为包含 $GBASEDBTDIR/bin 目录。
3. 将 ONCONFIG 环境变量设置为适当的配置文件。

4. 运行 gadmin -ky，启动立即关闭并使数据库服务器脱机。

   如果正在运行数据库服务器的多个版本（多处驻留），那么必须重置ONCONFIG 并为每个实例重新运行 gadmin -ky。

如果数据库服务器的不同版本安装在不同的目录中，您必须复位 GBASEDBTDIR 并为每个版本重复前面的步骤。

在 UNIX 关闭脚本中，当所有客户机应用程序已经完成它们的事务并退出之后，数据库服务器关闭命令就会运行。

准备连接到应用程序

当数据库服务器联机时，您可以连接客户机应用程序并开始创建数据库。在您可以访问数据库中的信息之前，客户机应用程序必须连接到数据库服务器环境。要连接到数据库服务器或断开连接，您可以从以下客户机程序发出 SQL 语句

- DB-Access
- SQL 编辑器
- GBase 8s ESQL/C
- GBase 8s ODBC Driver
- GBase 8s JDBC Driver

有关创建数据库的信息，请参阅《GBase 8s SQL 指南：教程》。有关如何使用客户机应用程序的信息，请参阅《GBase 8s DB-Access 用户指南》、《GBase 8s ESQL/C 程序员手册》、《GBase 8s ODBC Driver 程序员手册》或《GBase 8s JDBC Driver 程序员手册》。

创建存储空间和块

您负责规划和实现存储空间和块的配置。如何在磁盘上分布数据将影响数据库服务器的性能。

块与已经指定给数据库服务器的逻辑卷、逻辑单元或常规文件相同。 单个块的最大大小是 4 TB。

一个实例中最多可以有 32766 个块。可以将这些块全部放入一个存储空间中，也可以将其分散放入多个存储空间中。

逻辑存储空间由一个或多个块组成。

提示： 要利用每块 4 TB 的上限，应为每个磁盘驱动器指定一个块。这种分布数据的方法将提高性能。

初始化数据库服务器之后，可以创建存储空间（如数据库空间、BLOB 空间或智能大对象空间）。使用 gspaces 实用程序可创建存储空间和块。

初始化数据库服务器之后，可以创建存储空间（如数据库空间、BLOB 空间或智能大对象空间）。使用 gspaces 实用程序或 ISA 创建存储空间和块。

如果您要使用以下功能，必须创建智能大对象空间：

- J/Foundation（保存 Java™ JAR 文件）
- Enterprise Replication（容纳假脱机行数据）
- 智能大对象（BLOB 和 CLOB 数据类型）

有关存储空间和其他物理单元（如表空间和扩展数据块）的描述，请参阅数据存储。有关存储空间的分配和管理的说明，请参阅管理磁盘空间。

支持大块

要支持最大为 4 TB 以及大于 2047 块的大块和大偏移量，请运行 gadmin -BC 1。

您可以在 gadmin -BC 1 方式下测试数据。 如果您确信已正确转换了数据，那么可以运行 gadmin -BC 2，从而使服务器以“仅大块”方式运行。

在运行 gadmin -BC 2 后，将不再支持还原。在启用了对大块的支持后，它就不能禁用了。

设置备份系统和存储

可以使用 ON-Bar 实用程序或 gtape 实用程序来备份和复原数据。

确定要使用哪个实用程序，然后准备备份数据：

- 如果您将 gtape 用作备份工具，您必须在可以备份与复原数据前设置存储设备（磁带机）。Gtape 实用程序不需要存储管理器。
- 如果您将 ON-Bar 用作备份工具，您必须先设置存储管理器和存储设备才能备份与复原数据。

  存储管理器是管理包含备份的存储设备和介质的应用程序。存储管理器将处理所有的介质标号、安装请求以及存储卷。ISM 可以一次将数据备份到多达四个的存储设备中。ISM 将数据存储在简单磁带机、光盘设备以及文件系统上。如果要使用更先进的存储设备，一次备份到四个以上的存储设备或通过网络进行备份，那么可以从其他供应商处购买存储管理器。

有关设置和使用 ON-Bar 或 gtape 的信息，请参阅《GBase 8s 备份与复原指南》。

2.1.8 自动终止空闲连接

可以通过启用 idle_user_timeout 调度程序任务来自动终止与已空闲指定时间的客户机的会话。

必须以用户 gbasedbt 或其他授权用户身份连接 sysadmin 数据库。

要启用 idle_user_timeout 任务，请运行以下语句：

```
UPDATE ph_task
    SET tk_enable = 't'
    WHERE tk_name = 'idle_user_timeout';
```

缺省情况下，idle_user_timeout 任务将终止空闲时间超过 60 分钟的用户会话。无法终止用户 gbasedbt 的会话。idle_user_timeout 任务会在两小时之后开始检查是否有空闲会话，这是该任务的缺省频率。

提示： 数据库服务器计算机上的系统时间更改之后，用户会话已空闲的时间量将不再准确。例如，如果用户会话上次工作的时间为下午 3:14，而系统时钟在下午 3:15 向前拨了一个小时，那么对于数据库服务器而言，用户会话的空闲时间已经超过了一小时。

要更改空闲超时周期，请更新任务的运行频率和该阈值的值。允许的最短空闲超时周期为 5 分钟。例如，要将超时周期更改为 5 分钟，请运行以下语句：

```
UPDATE ph_task
   SET tk_frequency = INTERVAL (5) MINUTE TO MINUTE
   WHERE tk_name = 'idle_user_timeout';


UPDATE ph_threshold
    SET value = '5'
    WHERE task_name = 'idle_user_timeout';
```

2.1.9 配置会话属性

可在连接或访问时更改数据库服务器会话的属性，而不更改会话所运行的应用程序。如果不能修改应用程序的源代码来设置环境选项（或环境变量）或包含与会话相关的 SQL 语句（例如，由于 SQL 语句包含从供应商处获得的代码）时，该操作很有帮助。

要更改会话的属性，可为各种数据库设计定制 sysdbopen()和 sysdbclose() 过程以支持特定用户或 PUBLIC 组的应用程序。Sysdbopen() 和 sysdbclose() 过程可包含数据库服务器在数据库打开或关闭时为用户或 PUBLIC 组执行的一系列 SET、SET ENVIRONMENT、SQL 或 SPL 语句。

例如，对于 user1，可定义包含 SET PDQPRIORITY、SET ISOLATION LEVEL、SET LOCK MODE、SET ROLE 或 SET EXPLAIN ON 语句的过程，无论何时 user1 使用 DATABASE 或 CONNECT TO 语句打开数据库时，这些过程都将执行。

Sysdbopen() 过程中由 SET ENVIRONMENT 语句指定的会话环境变量 PDQPRIORITY 和 OPTCOMPIND 的任何设置都将在整个会话期间保持。对于常规过程非持久的 SET PDQPRIORITY 和 SET ENVIRONMENT OPTCOMPIND 语句，在 sysdbopen() 过程包含它们时将保持。

当作为过程所有者的用户从数据库断开连接时，将运行 user.sysdbclose() 过程（或者此时将运行 PUBLIC.sysdbclose()，前提是此过程存在且当前用户不具有任何 sysdbclose() 过程）。

在定制 sysdbopen() 和 sysdbclose() 过程中，如果在不符合 ANSI 标准的数据库中调用了例程，那么 GBase 8s 将不会忽略 UDR 所有者的名称。

有关更多信息，请参阅配置会话属性和《GBase 8s SQL 指南：语法》。

配置会话属性

只有 DBA 或 **gbasedbt** 用户可以在 SQL 的 ALTER PROCEDURE、ALTER ROUTINE、CREATE PROCEDURE、CREATE PROCEDURE FROM、CREATE ROUTINE FROM、DROP PROCEDURE 或 DROP ROUTINE 语句中创建或更改sysdbopen()或 sysdbclose()。

可在连接或访问时设置更改会话属性的 sysdbopen() 过程，而不更改会话运行的应用程序。如果不能修改应用程序的源代码来设置环境选项（或环境变量）或包含与会话相关的 SQL 语句（例如，由于 SQL 语句包含从供应商处获得的代码）时，该操作很有帮助。

要设置 sysdbopen() 和 sysdbclose() 过程以配置会话属性，请执行以下操作：

1. 将 **IFX_NODBPROC** 环境变量设置为任何值（包括 0）以使数据库服务器绕过并阻止 sysdbopen() 或 sysdbclose() 过程的执行。
2. 编写 CREATE PROCEDURE 或 CREATE PROCEDURE FROM 语句以定义特定用户或 PUBLIC 组的过程。
3. 测试过程（例如，通过使用 EXECUTE PROCEDURE 语句中的 sysdbclose()）。
4. 取消设置 **IFX_NODBPROC** 环境变量以使数据库服务器能够运行 sysdbopen() 或 sysdbclose() 过程。

示例

以下过程设置特定用户的角色和 PDQ 优先级：

```
create procedure oltp_user.sysdbopen()
    set role to oltp;
    set pdqpriority 5;
end procedure;
```

以下过程设置 PUBLIC 组的角色和 PDQ 优先级：

```
create procedure public.sysdbopen()
    set role to others;
    set pdqpriority 1;
end procedure
```

有关更多信息，请参阅《GBase 8s SQL 指南：语法》中有关 sysdbopen() 和sysdbclose() 的信息。

2.1.10 执行例程管理任务

根据贵组织的需求，您可以负责执行以下各段中描述的定期任务。并不是所有这些任务都适合每个安装。例如，如果您的数据库服务器每天运行 24 小时、每周运行 7 天，那么您就不可以使数据库服务器进入脱机方式，这样数据库服务器的运行方式将不能定期更改。

更改数据库服务器方式

数据库服务器管理员负责通过更改方式来启动和关闭数据库服务器。数据库服务器运行方式说明了如何更改数据库服务器方式。

备份数据和逻辑日志文件

要确保能够在发生故障时恢复数据库，请经常备份存储空间和逻辑日志。您也可以使用 archecker 实用程序验证 ON-Bar 备份。

每隔多少时间备份一次存储空间将取决于更新数据的频率以及数据的重要性如何。备份调度可以包括每周进行一次完全（0 级）备份、每天进行增量（1 级）备份以及每小时进行 2 级备份。还必须在执行管理任务（如添加数据库空间、删除逻辑日志文件或启用镜像）后执行 0 级备份。

每个逻辑日志文件一满就立即备份该文件。可以手动或自动备份这些文件。有关使用 ON-Bar 和 gtape 的信息，请参阅《GBase 8s 备份与复原指南》。

监视活动

GBase 8s 数据库服务器的设计使您能够监视数据库服务器的每个方面。监视数据库服务器活动提供了对于可用信息的描述、有关获取信息的指示信息以及有关使用信息的建议。

检查一致性

对数据的一致性执行不定期的检查。有关这些任务的描述，请参 阅一致性检查。

2.1.11 执行其他管理任务

这些主题涵盖了您将在生产数据库服务器上执行的各种管理任务。

磁盘镜像

当您使用磁盘镜像时，数据库服务器会将数据写入两个位置。镜像将消除由存储设备故障所造成的数据丢失。如果由于任何原因镜像数据变得不可用，那么将立即透明地向用户提供数据的镜像。有关镜像的信息，请参阅镜像。有关与镜像相关的任务的指示信息，请参阅使用镜像。

重要： 为物理日志、根数据库空间以及包含逻辑日志文件的关键数据库空间建立镜像。

管理数据库日志记录状态

可指定是否每个数据库在缺省情况下都使用事务日志记录；数据库的缺省日志记录方式是已缓冲还是未缓冲；以及日志记录方式是否符合 ANSI 标准。

可以在日志记录数据库中创建以下表的类型：

- STANDARD
- TEMP
- RAW

有关更多信息，请参阅临时表和日志记录。有关如何更改日志记录选项的信息，请参阅管理数据库日志记录方式。

管理逻辑日志

数据库服务器包含多个称为逻辑日志的文件，这些文件记录数据事务和管理信息（如检查点记录以及块的添加和删除）。

典型的逻辑日志管理任务包括备份逻辑日志文件，添加、释放逻辑日志文件和调整其大小，以及指定高水位标志。

数据库服务器在联机时动态地分配逻辑日志文件用以防止长事务使数据库服务器挂起。

有关更多信息，请参阅逻辑日志。有关创建和修改逻辑日志配置的指示信息，请参阅管理逻辑日志文件。有关备份逻辑日志的信息，请参阅《GBase 8s 备份与复原指南》。

管理物理日志

您可以更改物理日志的大小和位置。有关物理日志的更多信息，请参阅物理日志记录、检查点和快速恢复和管理物理日志。

当数据库服务器启动时，它会检查物理日志是否为空，因为物理日志为空的话就意味着该服务器以受控方式关闭。如果物理日志不为空，那么数据库服务器将自动执行称为快速恢复的操作。 系统发生故障后，可能会使一个或多个事务保持未落实状态，而快速恢复会自动将数据库复原到物理和逻辑一致性状态。

管理共享内存

管理共享内存包含以下任务：

- 更改缓冲区的大小和数量（通过更改逻辑日志或物理日志缓冲区的大小或更改共享内存缓冲池中缓冲区的数量）
- 更改共享内存参数值（如有必要）
- 更改强制的驻留（开或关，临时或用于此会话）
- 调整检查点间隔
- 将段添加到共享内存的虚拟部分
- 使用 SQL 语句高速缓存以减少内存的使用以及查询的准备时间

有关数据库服务器如何使用共享内存的信息，请参阅共享内存。有关其他信息，请参阅管理共享内存。

管理虚拟处理器

虚拟处理器 (VP) 的配置和管理对数据库服务器的性能具有直接的影响。数据库服务器的 VP 的最佳数量和组合取决于硬件和数据库服务器支持的应用程序类型。 有关虚拟处理器的说明，请参阅虚拟处理器和线程。有关其他信息，请参阅管理虚拟处理器。

管理并行数据库查询

您可以控制数据库使用的资源来并行执行决策支持查询。 必须协调决策支持查询与联机事务处理 (OLTP) 查询之间的需求。必须考虑的资源包括共享内存、线程、临时表空间及扫描带宽。

数据复制

数据复制是在多个不同站点上提供数据库对象的过程。数据复制配置由一个主服务器和一个或多个诸如 HDR 辅助服务器的辅助服务器、一台或多台 SD 辅助服务器以及一个或多个 RS 辅助服务器组成。此外，GBase 8s 支持 GBase 8s Enterprise Replication(ER)。可在同一数据库服务器上结合数据复制和 Enterprise Replication。

数据复制环境

带有 HDR 辅助服务器的主服务器支持将整个数据库同步复制到辅助数据库服务器，并且在发生灾难性计算机故障时提供热待机。

带有 SD 辅助服务器的主服务器提供对与主服务器分享的磁盘阵列上的数据的只读访问。带有 RS 辅助服务器的主服务器支持数据库在主服务器上的异步复制。带有 HDR 辅助服务器、RS 辅助服务器和 SD 辅助服务器的主服务器可以在数据复制环境中存在。

Enterprise Replication

Enterprise Replication 支持在按地理分布的数据库服务器上执行异步数据复制，因此您既可以复制整个数据库，又可以复制数据库和表的子集。Enterprise Replication 对用户定义的数据类型提供有限的支持。

审计

如果想要使用数据库服务器审计，那么必须指定存储审计记录的位置、如何处理错误条件等等。如果您怀疑用户正在滥用他们的特权，那么您可能还希望更改审计用户的方法。

分布式查询

可以使用数据库服务器跨同一数据库服务器实例的多个数据库服务器查询和更新多个数据库（跨数据库分布式查询），以及跨多个服务器实例查询和更新多个数据库（跨服务器分布式查询）。这类查询称为分布式查询。 此术语不限为 SELECT 语句，而更多时候是指在本地数据库外返回或引用对象的任何 DML 操作或例程的执行。在跨服务器分布式查询中，参与的数据库服务器可以位于单个主机、同一网络或某个网关上。

有关分布式查询的更多信息，请参阅《 GBase 8s SQL 指南：教程》。

全局事务

全局事务是涉及多个数据库服务器的一种事务。GBase 8s 数据库服务器支持以下两种类型的全局事务：

- TP/XA（带事务管理器）
- 两阶段落实

GBase 8s 使用两阶段落实协议来确保分布式查询跨多个数据库服务器统一落实或回滚。 有关更多信息，请参阅多阶段落实协议。

事务管理器

事务管理器将管理终端和数据恢复。TP/XA 库支持在 X/Open 环境中从供应商处获得的事务管理器和 GBase 8s 数据库之间进行通信。有关更多信息，请参阅事务管理器。

2.1.12 监视数据库服务器活动

可以使用多种工具来监视数据库服务器活动。大多数工具可用于 UNIX™ 平台。如本主题中的表所示。

可以从以下源收集有关数据库服务器活动的信息。

信息源	UNIX
事件警报	X
消息日志	X
ON-Monitor	X
gcheck 实用程序	X
onperf 实用程序	X
gstat 实用程序	X
SMI 表	X
系统控制台	X

以下主题说明了其中每个源。

事件警报

要报告需要您立即注意的情况，数据库服务器会使用事件警报功能。要使用事件警报功能，请将 ALARMPROGRAM 配置参数设置为执行必要的管理操作的可执行文件的完整路径名。

有关更多信息，请参阅《 GBase 8s 管理员参考 》中有关事件警报的附录和有关配置参数的章节。

Server Administrator (ISA)

ISA 是基于浏览器的工具，可为整个系列的 GBase 8s 数据库服务器提供系统管理。通过 ISA，几乎可以访问所有 GBase 8s 数据库服务器命令行函数。

有关 ISA 的更多信息，请参阅 ISA 联机帮助和屏幕指示信息。

消息日志

数据库服务器消息日志为操作系统文件。这些包含在数据库服务器消息日志中的消息通常不需要立即操作。

要报告需要您立即注意的情况，数据库服务器会使用事件警报。有关更多信息，请参阅事件警报。

指定消息日志消息的目标

要指定消息日志的路径名，请设置 MSGPATH 配置参数。对 MSGPATH 所作的更改会在关闭并重新启动数据库服务器后生效。

监视消息日志

每天必须监视一到两次消息日志，以确保处理正常进行，并且是在按预期记录事件。使用 gstat -m 命令以获取消息日志的名称以及 20 条最近的条目。 使用文本编辑器读取完整的消息日志。使用操作系统命令（如 UNIX™ 命令 tail -f）可查看出现的消息。

监视消息日志大小，因为数据库服务器将向此文件追加新的条目。按需编辑日志，或将其备份到磁带并将其删除。

如果数据库服务器遇到故障，消息日志将作为审计跟踪重新跟踪那些今后将发展成为预料不到的问题的事件。数据库服务器通常会在消息日志中提供问题的确切性质以及建议的更正操作。

您可以读取数据库服务器消息日志，了解有关数据库服务器处理的每分钟的报告以便在问题变大前捕捉事件。但是，您无需执行此类监视。

ON-Monitor (UNIX™)

ON-Monitor 提供了监视数据库服务器的许多方面的简单方法。大多数监视功能可以在状态菜单中获取。

gcheck 实用程序

gcheck 实用程序显示有关数据库磁盘配置和使用情况的信息，如用于表的页数、保留页的内容以及表中扩展数据块的数量。

gstat 实用程序

gstat 实用程序提供从命令行监视数据库服务器共享内存的方法。gstat 实用程序从共享内存读取数据，并报告针对命令执行瞬间的精确的统计信息。也就是说，gstat 提供在处理期间动态更改的信息，包括缓冲区、锁、索引和用户的更改。

SMI 表

系统监视接口 (SMI) 表是由数据库服务器管理的特殊表，这些表包含有关数据库服务器状态的动态信息。可对这些表使用 SELECT 语句来确定您可能想要了解的有关数据库服务器的几乎所有信息。有关 SMI 表的描述，请参阅《GBase 8s 管理员参考》中有关 sysmaster 数据库的主题。

系统控制台

数据库服务器将通过系统控制台发送对数据库服务器管理员有用的消息。要指定控制台消息的目标路径名，请设置 CONSOLE 配置参数。

对 CONSOLE 所作的更改会在关闭并重新启动数据库服务器后生效。

UNIX 操作系统工具

数据库服务器依靠主机的操作系统提供对系统资源（例如，CPU、内存和各种未缓冲的磁盘 I/O 接口和文件）的访问。每个操作系统都有其自己的一套实用程序用来报告使用系统资源的状况。不同的操作系统可能有具有相同名称的监视实用程序，但它们的选项和信息显示不同。

下表显示典型的 UNIX™ 操作系统资源监视实用程序。有关如何监视操作系统资源的信息，请查询系统管理指南。

UNIX 实用程序	描述
vmstat	显示虚拟内存统计信息
iostat	显示 I/O 利用率统计信息
sar	显示各种资源统计信息
ps	显示活动进程信息
cron	通过系统调度程序（每隔一定时间运行命令或程序）来捕获系统资源的状态。 还可以使用其他可由操作系统提供的调度工具。

2.2 客户机/服务器通信

以下主题说明了要配置客户机/服务器通信所必须了解的概念和术语。

2.2.1 客户机/服务器体系结构

GBase 8s 产品符合客户机/服务器软件设计模型。应用程序或客户机可以位于托管数据库服务器的计算机上，也可以位于其他计算机上。客户机应用程序将发出请求以获得数据库服务器的服务和数据。数据库服务器通过提供客户机请求的服务和数据来作出响应。

将网络协议连同网络编程接口一起使用可连接客户机和数据库服务器并在它们之间传输数据。

网络协议

网络协议是一组规则，规定了如何在应用程序间传输数据以及在此上下文中如何在客户机与数据库服务器之间传输数据。

协议的规则在网络驱动程序中实施。 在数据从客户机发送到数据库服务器以及从数据库服务器发送到客户机时，网络驱动程序包含的代码会将数据格式化。

客户机和数据库服务器通过网络编程接口获得对网络驱动程序的访问权。网络编程接口包含的系统调用或库例程提供了对网络通信设施的访问权。 UNIX™ 的网络编程接口的示例是 TLI（传输层接口）。

网络协议的作用在于它能够启用客户机/服务器通信，尽管客户机和数据库服务器位于具有不同体系结构和操作系统的不同计算机上。

您可以配置数据库服务器来支持多个协议，但是仅当某些客户机使用 TCP/IP 时才考虑该选项。

网络编程接口

网络编程接口是一种应用程序编程接口 (API)，它包含一组通信例程或系统调用。应用程序可以调用这些例程以便与位于相同或不同计算机上的其他应用程序通信。在该说明的上下文中，客户机和数据库服务器是调用 TLI 或套接字 API 中的例程的应用程序。客户机和数据库服务器都使用网络编程接口根据通信协议发送和接收数据。

如果要使客户机/服务器通信成功，那么客户机和数据库服务器的环境都必须用相同的协议配置。然而，一些网络协议可以通过多个网络编程接口来访问。例如，根据操作系统平台上可用的编程接口，可通过 TLI 或套接字来访问 TCP/IP。

数据库服务器连接

客户机应用程序使用 CONNECT 或 DATABASE SQL 语句建立与数据库服务器的连接。例如，应用程序可能包含以下 CONNECT 语句以连接到名为 my_server 的数据库服务器：

```
CONNECT TO '@my_server'
```

提示：数据库服务器的内部通信设施称为关联服务设施 (ASF)。如果看到一条包含对 ASF 的引用的错误消息，那么说明连接有问题。

支持多路复用连接

有些应用程序会代表一个用户多次连接到同一台数据库服务器。多路复用连接在数据库服务器和客户机之间使用单个网络连接以便处理来自客户机的多个数据库连接。客户机应用程序可以建立多个到数据库服务器的连接以代表单个用户访问多个数据库。如果连接不是多路复用的，那么每个数据库连接将建立到数据库服务器的单独网络连接。每个额外的网络连接都将使用额外的计算机内存和处理器时间，甚至对于不活动的连接亦如此。多路复

用连接使数据库服务器可以创建多个数据库连接，而不会耗尽额外网络连接所需的额外计算机资源。

要配置数据库服务器以支持多路复用连接，请执行以下操作：

1. 使用 DBSERVERALIASES 配置参数定义别名。

   例如，指定：

   ```
   DBSERVERALIASES ifx_mux
   ```

2. 为别名添加 sqlhosts 文件条目，其中将 onsqlmux 用作 nettype 条目。hostname 和 servicename 必须具有条目，但这些条目会被忽略。可将连字符 (-) 用作条目。

   例如：

   ```
   #dbservername    nettype      hostname    servicename    options
   ifx_mux          onsqlmux     -           -
   ```

3. 通过在客户机用于数据库服务器连接的 sqlhosts 条目中指定m=1，从而启用选定连接类型的多路复用。

   例如：

   ```
   #dbservername    nettype      hostname    servicename    options
   menlo            ontlitcp     valley      jfkl           m=1
   ```

以下示例显示了 onconfig 文件和 sqlhosts 文件条目。

onconfig 文件：

```
DBSERVERNAME web_tli

DBSERVERALIASES web_mux
```

sqlhosts 文件：

```
#dbservername    nettype      hostname    servicename    options
web_tli          ontlitcp     node5       svc5           m=1
web_mux          onsqlmux     -           -
```

无需更改数据库服务器使用的 sqlhosts 信息。客户机程序无需执行任何特殊的 SQL 调用，即可启用连接多路复用。当 onconfig 文件和 sqlhosts 条目进行了适当配置并且数据库服务器启动时，将自动启用连接多路复用。

多路复用连接不支持：

- 多线程客户机连接
- 共享内存连接
- 与下级数据库服务器的连接（例如，用于分布式查询或数据复制）

如果在应用程序尝试建立连接时出现以上任一情况，数据库服务器就将建立标准连接。数据库服务器不返回 SQL 错误。

多路复用连接期间，不支持 GBase 8s ESQL/C sqlbreak() 函数。

2.2.2 数据库服务器支持的连接

数据库服务器支持与客户机应用程序的以下连接类型。

连接类型	UNIX™	本地	网络
套接字	X	X	X
TLI (TCP/IP)	X	X	X
共享内存	X	X	
安全套接字层 (Secure Sockets Layer, SSL)	X		X
流管道	X	X	
命名管道		X	

安全套接字层 (SSL) 连接对网络中两点之间的数据通信使用加密。

配置连接时，请考虑设置 LISTEN_TIMEOUT 和 MAX_INCOMPLETE_CONNECTION 配置参数。通过使用这些参数，处理连接的侦听器 VP 更不容易因连接过多而发生问题，这样您就能够降低受到恶意的拒绝服务 (DOS) 攻击的风险。

仅限 UNIX： 在许多 UNIX 平台上，数据库服务器都支持多个网络编程接口。机器说明显示了数据库服务器支持用于您操作系统的接口/协议组合。

要设置客户机连接，请执行以下操作：

1. 在 onconfig 文件中指定连通性和连接配置参数。
2. 在平台上的连接文件中设置适当的条目。
3. 请在 UNIX 启动脚本
4. 添加 sqlhosts 条目以定义数据库服务器的数据库服务器组。

2.2.3 本地连接

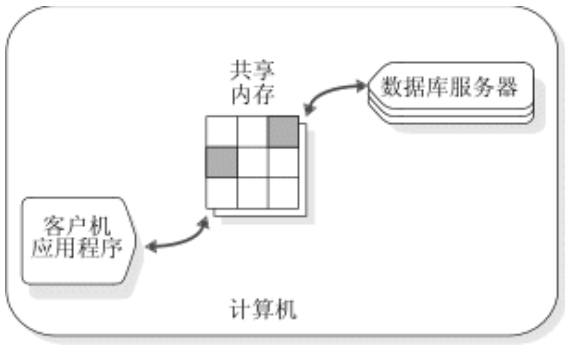
本地连接是同一台计算机上客户机和数据库服务器之间的连接。以下主题描述了不同的本地连接类型。

共享内存连接 (UNIX™)

共享内存连接使用共享内存的一块区域作为客户机和数据库服务器相互通信所通过的通道。客户机不能与数据库服务器建立多个共享内存连接。

下图说明了共享内存连接。

图： 通过共享内存连接进行的客户机应用程序和数据库服务器通信

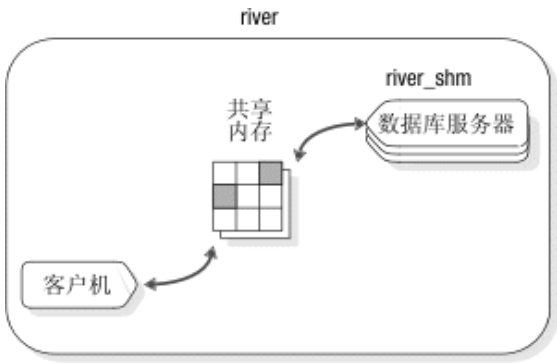


共享内存提供对数据库服务器的快速访问，但是它会引起一些安全性方面的风险。错误或恶意的应用程序可能删除或查看其自己的或其他本地用户的消息缓冲区。 如果客户机应用程序执行显式内存寻址或过度索引数据数组，那么共享内存通信也容易受到编程错误的影响。如果使用的是网络通信或流管道，那么这些错误不会影响数据库服务器。

共享内存连接的示例

下图显示名为 **river** 的计算机上的共享内存连接。

图： 客户机应用程序与名为 river_shm 的数据库服务器之间的共享内存连接。



此安装的 onconfig 文件包含以下行：

```
DBSERVERNAME river_shm
```

此安装的 sqlhosts 文件包含以下行：

```
#dbservername      nettype        hostname        servicename         options
 river_shm          onipcshm        river             rivershm
```

客户机应用程序将连接到使用以下语句的此数据库服务器：

```
CONNECT TO '@river_shm'
```

对于共享内存连接，无需网络配置文件中的任何条目。对 sqlhosts 文件的 hostname 和 servicename 字段使用任意值。

流管道连接（UNIX 和 Linux™）

流管道是 UNIX™ 进程间通信 (IPC) 设施，该设施允许同一计算机上的进程可以互相通信。

流管道连接有下列优势：

- 与共享内存连接不同，流管道不会引起被其他显式地访问共享内存的相同部分的程序覆盖或读取的安全性风险。
- 与共享内存连接不同，流管道连接允许同处一台计算机的数据库服务器之间有分布式事务。

流管道连接有下列劣势：

- 在一些计算机上，流管道连接可能比共享内存连接慢。
- 流管道并不可用于所有平台。
- 当您将共享内存或流管道用于客户机/服务器通信时，hostname 条目被忽略。

本地回送连接

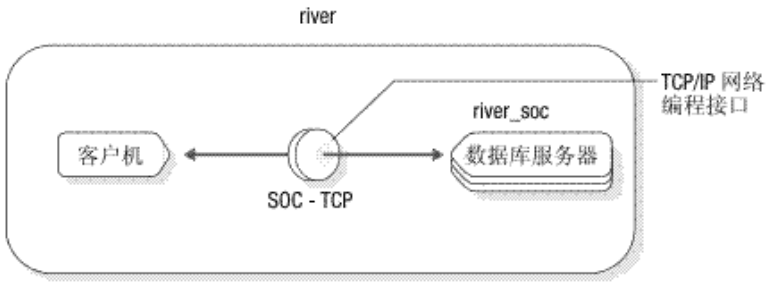
同一台计算机上客户机应用程序和数据库服务器之间的网络连接称为本地回送连接。 所用的网络工具是相同的，就像客户机应用程序和数据库服务器位于不同的计算机上一样。您可以进行本地回送连接，条件是已装备了您的计算机来处理网络事务。 本地回送连接不如共享内存连接快，但它们不会引起共享内存的安全性风险。

在本地回送连接中，数据看上去是从客户机应用程序传出到网络，然后再次传回数据库服务器。内部连接进程将直接在客户机和数据库服务器之间发送信息，而不会在网络上向外传输该信息。

本地回送连接的示例

下图显示了使用套接字和 TCP/IP 的本地回送连接。

图： 名为 river 的计算机上客户机与名为 river_soc 的数据库服务器之间的本地回送连接。



此安装的 sqlhosts 文件包含以下行：

```
#dbservername      nettype         hostname        servicename         options
river_soc          onsoctcp        river           riverol
```

如果网络连接使用 TLI 来代替套接字，那么仅本例中的 nettype 条目将更改。在那种情况下，nettype 条目是 ontlitcp 而不是 onsoctcp。

此安装的 onconfig 文件包含以下行：

```
DBSERVERNAME river_soc
```

此示例假定 river 的条目在 hosts 文件中，而 riverol 的条目在 services 文件中。

2.2.4 通信支持服务

通信支持服务包括与连接相关的服务，例如：

- **认证**，这是验证用户或应用程序的身份的过程。最普通形式的认证是要求用户输入密码以获得对计算机或应用程序的访问权。
- **消息完整性**，用于确保通信消息在到达其目的地时完整无缺且未变更。
- **消息机密性**，用于在传输期间保护消息以免受到未经授权的查看，通常是通过加密和解密来实现这一点。

通信支持服务还可以包括其他处理，如数据压缩或基于流量的记帐。

数据库服务器通过名为“通信支持模块”(CSM) 的插件软件模块来提供额外的安全性相关通信支持服务。数据库服务器将在您未指定通信支持模块时使用该缺省认证策略。

2.2.5 连接文件

连接文件包含支持客户机/服务器通信的信息，并支持数据库服务器与其他数据库服务器通信。

这些连接配置文件可以分成三组：

- 网络配置文件
- 网络安全性文件
- sqlhosts 文件

网络配置文件

这些主题标识并说明网络配置文件在 TCP/IP 网络上的使用。

TCP/IP 连接文件

当配置数据库服务器以使用 TCP/IP 网络协议时，您可使用 hosts 和 services 文件中的信息来准备 sqlhosts 信息。

对于连接正在网络上运行 GBase 8s 客户机/服务器产品的计算机的每个网络控制器卡，hosts 文件都需要一个单独的条目。文件中的每个条目都包含 IP 地址（或以太网卡地址）

和主机名。您还可以包含主机别名。尽管主机名的长度在 hosts 文件中没有限制，但数据库服务器仍将主机名限制为 256 个字节。

以下示例具有两个条目。

```
#address          hostname    alias
98.555.43.21      odyssey
12.34.56.555        illiad      sales
```

services 文件包含对于在 TCP/IP 上可用的每个服务的条目。每个条目都是一个单独的行，包含以下信息：

- 服务名

  GBase 8s 产品使用该名称来确定用于建立客户机/服务器连接的端口号和协议。服务名称限制为 128 个字节。

- 端口号和连接协议，由正斜杠分隔

  端口号是计算机端口，而 TCP/IP 的协议是 tcp。

  操作系统会在端口号上施加限制。用户 gbasedbt 必须使用等于或大于 1024的端口号。只允许 root 用户使用小于 1024 的端口号。

- 主机别名（可选）

  服务名称和端口号为随机。然而，它们在文件的上下文中必须唯一，并且在运行 GBase 8s 客户机/服务器产品的所有计算机上必须完全相同。以下示例具有一个条目：

  ```
  #servicename     port/protocol
  server2          1526/tcp
  ```

  该条目导致 server2 充当 TCP 端口 1526 的服务名称。然后，数据库服务器可以使用此端口来处理连接请求。

重要： 对于与其他数据库服务器通信的数据库服务器，必须为 DBSERVERNAME 配置参数定义 TCP/IP 连接或 IPCSTR（进程间通信流管道）连接。 您还可以使用正确的连接协议为协调程序与下级服务器之间的连接定义至少一个 DBSERVERALIASES 配置参数设置。对于跨服务器事务，每个参与服务器必须支持与协调程序之间进行的 TCP/IP 或 IPCSTR 连接，即使两个数据库服务器实例在同一工作站上。

通常，为每个与数据库服务器名称关联的连接类型都包含单独的 NETTYPE 参数。您可在 DBSERVERNAME 和 DBSERVERALIASES 配置参数中列出数据库服务器名称条目。通过 sqlhosts 文件或注册表中的条目，可将连接类型与数据库服务器名称相关联。

hosts 和 services 文件必须可用于运行 GBase 8s 客户机/服务器产品的每台计算机。

UNIX：

- hosts 和 services 文件位于 /etc 目录中。

- 在使用 NIS 的系统上，hosts 和 services 文件都将保留在 NIS 服务器上。位于本地计算机上的 hosts 和 services 文件可能不会使用，也可能不是最新的。要查看 NIS 文件的内容，请在命令行上输入以下命令：

```
ypcat hosts

ypcat services
```

TCP/IP 连接文件 (UNIX™)

hosts 和 services 文件位于 /etc 目录中。文件必须可用于运行 GBase 8s 客户机/服务器产品的每台计算机。

在使用 NIS 的系统上，hosts 和 services 文件都将保留在 NIS 服务器上。位于本地计算机上的 hosts 和 services 文件可能不会使用，也可能不是最新的。要查看 NIS 文件的内容，请在命令行上输入以下命令：

```
ypcat hosts

ypcat services
```

打开 TCP/IP 连接时客户机和服务器的操作

当打开 TCP/IP 连接时，在客户机方可以读取到以下信息：

- GBASEDBTSERVER 环境变量。
- hosts 文件信息（GBASEDBTSQLHOSTS 环境变量、$GBASEDBTDIR/etc/sqlhosts 文件）和 services 文件信息
- 其他环境变量
- 资源文件

以下信息是在服务器方读取的：

- DBSERVERNAME 配置参数
- DBSERVERALIASES 配置参数
- 服务器环境变量和配置参数，包括管理 TCP/IP 连接的任何 NETTYPE 配置参数设置。

多个 TCP/IP 端口

要利用多块以太网卡：

- 在 services 文件中为数据库服务器将使用的每个端口建立一个条目，如以下示例所示：

```
#servicename    port/protocol    alias

soc1              21/tcp

soc2              22/tcp
```

单个 IP 地址的每个端口必须唯一。独立的以太网卡可以使用唯一或共享的端口号。您可能希望在连接到同一个数据库服务器的几块以太网卡上使用同一个端口号。（在此方案中，服务名称相同。）

- 将每个条目（每个以太网卡一个）放入具有独立 IP 地址的 hosts 文件中，如下例所示：

```
#address                hostname      alias
192.147.104.19          svc8
192.147.104.20          svc81
```

- 在 onconfig 文件中，为其中一个以太网卡设置 DBSERVERNAME 配置参数，为另一个以太网卡设置 DBSERVERALIASES 配置参数。以下行显示 onconfig 文件中的样本条目：

```
DBSERVERNAME chicago1
DBSERVERALIASES chicago2
```

- 针对每张以太网卡添加一个 sqlhosts 条目。即，为 DBSERVERNAME 建立一个条目，并为 DBSERVERALIASES 建立另一个条目。

```
#dbservername   nettype      hostname      servicename      options
chicago1        onsoctcp       svc8          soc1
chicago2        onsoctcp       svc81         soc2
```

此配置完成后，应用程序将通过指定给数据库服务器名称（由 **GBASEDBTSERVER** 环境变量提供）的以太网卡进行通信。

网络安全性文件

GBase 8s 产品遵循由网络安全性文件中包含的信息管理的标准安全性过程。对于与远程计算机上数据库服务器连接的客户机应用程序，客户机应用程序的用户在远程计算机上必须具有有效的用户标识。

可信主机信息

/etc/hosts.equiv 和 .rhosts 文件为 rlogin、rsh、rcp 和 rcmd 提供了远程认证数据库。这些文件指定了可信的远程主机和用户。允许可信用户在不提供密码的情况下访问本地系统。hosts.equiv 文件应用于整个系统。各个用户在其主目录中维护自己的.rhosts 文件。

数据库服务器可以配置为使用 /etc/hosts.equiv 或 .rhosts 文件进行远程认证。要使用可信主机信息进行认证，应该在 sqlhosts 信息中指定 s=1 或 s=3 选项。如果未指定 s 选项，s=3 是缺省值。

可以将数据库服务器配置为使用其他文件进行远程认证。REMOTE_SERVER_CFG 配置参数可以指定要使用的备用文件，而不是 hosts.equiv 文件。 在以下情况下，此备用文件是必需的：

- 针对数据库服务器需要的可信主机不同于为操作系统列出的可信主机
- 安装时的安全策略不允许使用 hosts.equiv
- 您是非 root 服务器实例的用户，并且需要控制哪些主机可信

可信主机文件是指 hosts.equiv 文件或 REMOTE_SERVER_CFG 配置参数指定的文件。

如果客户机应用程序提供了无效的帐户名称和密码，那么即使可信主机信息包含客户端计算机的条目，数据库服务器也将拒绝连接。请仅将可信主机信息用于不提供用户帐户或密码的客户机应用程序。

usename 字段是可选的。包含用户名可将连接限制到指定的用户。

要避免额外的 DNS 查找，请在可信主机文件中指定包含和不包含域名的主机名。例如，如果可信主机名为 **argo**，并且位于域 **example.com** 中，那么可信主机文件将指定以下主机：

```
#trustedhost          #username
argo                   gbasedbt
argo.example.com    gbasedbt
```

在一些网络中，远程主机用于连接到特定计算机的主机名可能与计算机用于表示其自身的主机名不同。例如，包含标准域名 (FQDN) **xyz.gbasedbt.com** 的网络主机可能通过本地主机名 **viking** 引用自身。如果发生这种情况，可信主机信息必须指定这两种主机名格式，如以下示例中所示：

```
#trustedhost
xyz.gbasedbt.com
viking
```

如果使用的是系统 hosts.equiv 文件并使用了 rlogind 守护程序，那么可以在客户端计算机上执行以下语句来确定客户机是否可信：

```
rlogin hostname
```

如果在没有收到密码提示的情况下成功登录，说明该客户机是可信计算机。当 REMOTE_SERVER_CFG 配置参数指定备用文件名时，这种用于确定客户机是否可信的方法将失效。

通过在 sqlhosts 文件中配置专用端口，可以提高 Enterprise Replication 连接和高可用性连接的安全性。

可信用户信息

在用户的 .rhosts 文件中，用户可以列出可从中作为可信用户进行连接的主机。.rhosts 文件位于数据库服务器所在的计算机上的用户主目录中。要启用可信用户认证，请在 sqlhosts 条目的选项中指定 s=2 或 s=3。如果未指定 s 选项，s=3 是缺省值。

无法使用用户的 .rhosts 文件的原因可能有多种。例如，非 root 安装可能对特定用户的 .rhosts 文件不具有读访问权。 通过设置 REMOTE_USERS_CFG 配置参数，可以指定备用文件名。如果设置此参数，数据库服务器只有一个可信用户文件用于所有用户。

.rhosts 文件的每一行是用户可以从中进行连接的一个主机。必须指定包含和不包含域名的服务器名称，以避免执行额外的 DNS 查找。 例如：

```
#trustedusers
xxx.example.com
xxx

yyy.example.com
yyy
```

REMOTE_USERS_CFG 配置参数指定的文件必须是各个 .rhosts 文件的组合。 文件中的每个单行条目都具有以下格式：

```
hostname username
```

例如，假设针对用户 John 和 Fred 存在以下两个 .rhosts 文件：

~john/.rhosts

```
#trustedhosts
xxx.example.com
xxx

yyy.example.com
yyy
~fred/.rhosts
#trustedhosts
xxx.example.com
xxx

zzz.example.com
zzz
```

John 不信任 **zzz.example.com** 或 **zzz**，而 **Fred** 不信任 **yyy.example.com** 或 **yyy**。

.rhosts 文件可以合并到具有以下格式的一个文件中：

```
#trustedhost        username
xxx.example.com     john
xxx                 john

yyy.example.com     john
yyy                 john

xxx.example.com     fred
xxx                 fred

zzz.example.com     fred
zzz                 fred
```

netrc 信息

netrc 信息是指定标识数据的可选信息。没有访问数据库服务器的权限的用户或不是数据库服务器所信任的计算机上的用户可以使用此文件提供可信的名称和密码。在远程计算机上拥有不同用户帐户和密码的用户也可以提供此信息。

UNIX™： netrc 信息位于用户主目录中的 .netrc 文件中。可使用任何标准的文本编辑器准备 .netrc 文件。netrc 条目的格式为：

```
machine machine_name login user_name password user_password
```

如果您没有在应用程序中为远程服务器显式地提供用户密码（即，通过 CONNECT 语句的 USER 子句或 DB-Access 中提示的用户名和密码），那么客户机应用程序将用户名和密码锁定在 netrc 信息中。 如果用户在应用程序中显式指定了密码，或者如果数据库服务器不是远程的，那么将不会参考 netrc 信息。

无论数据库服务器是否使用缺省的认证策略或通信支持模块，它都将使用 netrc 信息。

有关此文件的特定内容的信息，请参阅您的操作系统文档。

用户模仿

对于某些客户机查询或操作，数据库服务器必须模仿客户机，为客户机运行进程或程序。为了模仿客户机，数据库服务器必须接收每个客户机连接的密码。客户机可以通过 CONNECT 语句或 netrc 信息提供用户标识和密码。

以下示例显示了您可以如何提供密码以模仿客户机。

netrc

```
machine trngpc3 login bruce password im4golf
```

CONNECT 语句

```
CONNECT TO ol_trngpc3 USER bruce USING "im4golf"
```

sqlhosts 文件和 SQLHOSTS 注册表键

GBase 8s 客户机/服务器连接信息（sqlhosts 信息）包含允许客户机应用程序找到并连接到网络上任何 GBase 8s 数据库服务器的信息。

sqlhosts 文件 (UNIX™)

在 UNIX 上，缺省情况下，sqlhosts 文件位于 $GBASEDBTDIR/etc 目录中。

sqlhosts 文件中的每个条目包含一个数据库服务器或数据库服务器组的sqlhosts 信息。其中每个字段的附加语法规则将在以下各节中提供，它们将描述 sqlhosts 文件中的条目。 可使用任何标准的文本编辑器在sqlhosts 文件中输入信息。

使用文本编辑器创建 sqlhosts 文件 (UNIX™)

缺省情况下，sqlhosts 文件位于 $GBASEDBTDIR/etc 目录中。或者，您可以将 **GBASEDBTSQLHOSTS** 环境变量设置为包含sqlhosts 信息的文件的完整路径名和文件名。每个主管数据库服务器或客户机的计算机都必须有一个 sqlhosts 文件。

打开任何标准文本编辑器以创建 sqlhosts 文件。

注：

- 使用空格（空格、制表符或两者）分隔各个字段。
- 不要在字段内包含任何空格或制表符。
- 要在 sqlhosts 文件中放置注释，请以注释符 (#) 开始每一行。您还可以将若干行留空以增加可读性。

样本 sqlhosts 文件

以下代码块显示样本 sqlhosts 文件。

#dbservername	nettype	hostname	servicename	options
menlo	onipcshm	valley	menlo	
newyork	ontlitcp	hill	dynsrvr2	s=2,b=5120
payroll	onsoctcp	dewar	py1	
asia	group	-	-	e=asia.3
asia.1	ontlitcp	node6	svc8	g=asia
asia.2	onsoctcp	node0	svc1	g=asia
portland	drsocssl	dewar	portland_serv	

2.2.6 sqlhosts 信息

托管数据库服务器或客户机的每台计算机都必须包含连接信息。信息存储在 UNIX™ 操作系统上的 sqlhosts文件中。连接信息存储在 sqlhosts文件中。

sqlhosts 信息包含每个数据库服务器的连接信息。sqlhosts 信息还包含对于组的定义。当您启动数据库服务器、客户机应用程序连接到数据库服务器、或数据库服务器连接到另一个数据库服务器时，数据库服务器将查询连接信息。

在 sqlhosts 文件中，每一行包含一个数据库服务器的连接信息，或者一个组的定义。

- 每个数据库服务器的连接信息包括四个必填信息字段和一个可选字段。
- 组定义仅包括三个字段中的信息。

在注册表中，数据库服务器名称将指定给 SQLHOSTS 注册表键中的键，而其他字段是该键的值。

下表概括了用于 SQLHOSTS 信息的字段。

sqlhosts 文件中的字段名称	SQLHOSTS 注册表键中的字段名称	连接信息的描述	组信息的描述
dbservername	Database server name key 或 database server group key	数据库服务器名	数据库服务器组名称
nettype	PROTOCOL	连接类型	关键字 group
hostname	HOST	数据库服务器的主机	无信息。在此字段中使用短划线作为占位符。
servicename	SERVICE	端口号的别名	无信息。在此字段中使用短划线作为占位符。
options	OPTIONS	描述或限制连接的选项	组选项

sqlhosts.std 文件中的 IANA 标准服务名称和端口号

因特网指定号码权限 (IANA) 为 GBase 8s 数据库服务器指定了以下服务名称和端口号：

端口/服务	IANA 代码	描述
sqlexec	9088/tcp	GBase 8s SQL 接口
sqlexec-ssl	9089/tcp	GBase 8s SQL 接口 - 已加密

这些服务名称在 GBase 8s 的 sqlhosts.std 文件中创建。无需更改已安装的 GBase 8s 系统，因为这些系统将继续使用现有端口号和服务名称。（另外，不能保证其他某个系统尚未使用指定给 GBase 8s 的服务名称或端口号。）

具有以下标准的策略的组织可以使用这些服务名称和端口号，假如这些组织想要数据库服务器与 IANA 标准保持一致的话。如果安装在同一工作站上的另一应用程序已经使用了其中某个服务名称或端口号，那么可以要求不合规应用程序的发布者注册指定的 IANA 端口号，以避免冲突。应用程序不合规时，可以使用非标准端口来运行 GBase 8s。

有关更多信息，请参阅 IANA 组织的 Web 站点。

sqlhosts 连接信息

sqlhosts 文件中的字段或 SQLHOSTS 注册表键描述连接信息。

语法

元素	用途	限制
dbservername	指定要为其指定连接信息的数据库服务器的名称。 如果通过组关键字而非连接类型进行指定，那么指定一个组以将多个	该名称必须以小写字母开头，并且可以包含小写字母、数字和下划线 (_) 符号。字段长度限制为 128 个字节。

元素	用途	限制
	相关数据库服务器条目作为一个逻辑条目进行处理。您可以使用组来建立或更改客户机/服务器连接，或者简化到数据库服务器的连接重定向。	数据库服务器必须存在。其名称必须由 onconfig 文件中的 DBSERVERNAME 或 DBSERVERALIASES 配置参数指定。 无法将一个数据库服务器组嵌套在另一个数据库服务器组中。 数据库服务器可以是一个组的成员。
connection_type	描述数据库服务器和客户机应用程序或其他数据库服务器之间建立的连接的类型。	
hostname	指定数据库服务器所在的计算机。	字段长度限制为 256 个字节。 如果指定组关键字，那么必须为空值 (-)。
servicename	指定端口号的别名。服务名称字段的解释取决于连接类型字段中的连接类型。	字段长度限制为 128 个字节。 如果指定组关键字，那么必须为空值 (-)。

dbservername 字段

跨所有关联网络的每台数据库服务器必须有一个唯一的数据库服务器名称。

如果 sqlhosts 文件具有使用相同 dbservername 的多个条目，那么只使用第一个条目。

连接类型字段

连接类型字段在 sqlhosts 文件中称为 nettype，在 SQLHOSTS 注册表键中称为 PROTOCOL。

下表概括了不同操作系统上数据库服务器连接的可能连接类型值。

表 1. 连接类型摘要

UNIX™ 的值	描述	连接类型
onipcshm	共享内存通信。如果用于非 root 安装（这种情况下服务器和客户机位于不同位置），那么 sqlhosts 文件中需要 cfd 选项。	IPC
onipcstr	流管道通信。如果用于非 root 安装（这种情况下服务器和客户机位于不同位置），那么 sqlhosts 文件中需要 cfd 选项。	IPC
	命名管道通信	IPC
ontlitcp	TCP/IP 协议的 TLI	网络
onsocssl	安全套接字层 (SSL) 协议	网络
onsoctcp	使用 TCP/IP 协议的套接字	网络

UNIX™ 的值	描述	连接类型
onsocimc	用于与 GBase 8s MaxConnect 通信的 TCP/IP 协议的套接字	网络
ontliimc	用于与 GBase 8s MaxConnect 通信的 TCP/IP 协议的 TLI	网络
onsqlmux	多路复用连接	网络

注： 以“on”开头的连接类型值可以使用“ol”来代替“on”。例如，onipcshm 或 olipcshm 指定共享内存连接（如果在 sqlhosts 信息中使用）。

主机名字段

主机名在 sqlhosts 文件中的 hostname 字段以及 HOST 注册表键中输入。

如果连接类型是 onsqlmux，hostname 字段不能为空，但其中输入的任何具体值都将被忽略。

下面说明了客户机应用程序如何派生主机名字段中使用的值。

通过 TCP/IP 进行的网络通信

使用 TCP/IP 连接协议时，host name 字段是 hosts 文件的一个键，用于提供计算机的网络地址。hostname 字段中使用的名称必须与 hosts 文件中的名称相对应。在大多数情况下，hosts 文件中的主机名与计算机的名称相同。

在某些情况下，您可能希望在主机名字段中使用实际的因特网 IP 地址。

UNIX：共享内存和流管道通信

当您将共享内存或流管道用于客户机/服务器通信时，hostname 字段必须包含数据库服务器所在计算机的实际主机名。

多路复用连接

将 onsqlmux 用作连接类型时，hostname 字段必须有一个条目，但该条目将被忽略。短划线 (-) 可用作条目。

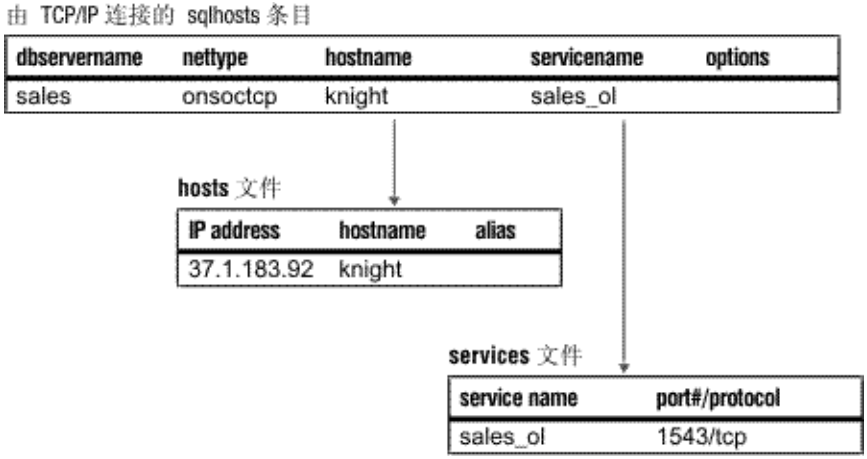
服务名称字段

通过 TCP/IP 进行的网络通信

服务名称字段在 UNIX 操作系统上称为 servicename。使用 TCP/IP 连接协议时，服务名称条目必须与 services 文件中的名称相对应。services文件中的端口号告诉网络软件如何在指定的主机上查找数据库服务器。

下图显示 sqlhosts 信息和 hosts 文件之间的关系，以及 sqlhosts 信息到 services 文件的关系。

图: sqlhosts 信息到 hosts 和 services 文件的关系



在某些情况下，可在服务名称字段中使用实际的 TCP 侦听端口号。

UNIX：共享内存和流管道通信

对于共享内存连接 (onipcshm) 或流管道连接 (onipcstr)，数据库服务器在内部使用 *servicename* 条目中的值来创建支持连接的文件。对于 onipcshm 和 onipcstr 这两种连接，*servicename* 可以是数据库服务器所在主机环境中唯一的任意字母短组。

提示： 对于流管道连接，请将 *dbservername* 用作 *servicename*。

多路复用连接

对于多路复用连接 (onsqlmux)，hostname 字段必须有一个条目，但该条目将被忽略。可将连字符 (-) 用作条目。

- **sqlhosts 文件和 SQLHOSTS 注册表键选项**
  您可以在 sqlhosts 文件或 SQLHOSTS 注册表键中包含服务器选项和组选项。

请参阅 sqlhosts 文件和 SQLHOSTS 注册表键选项

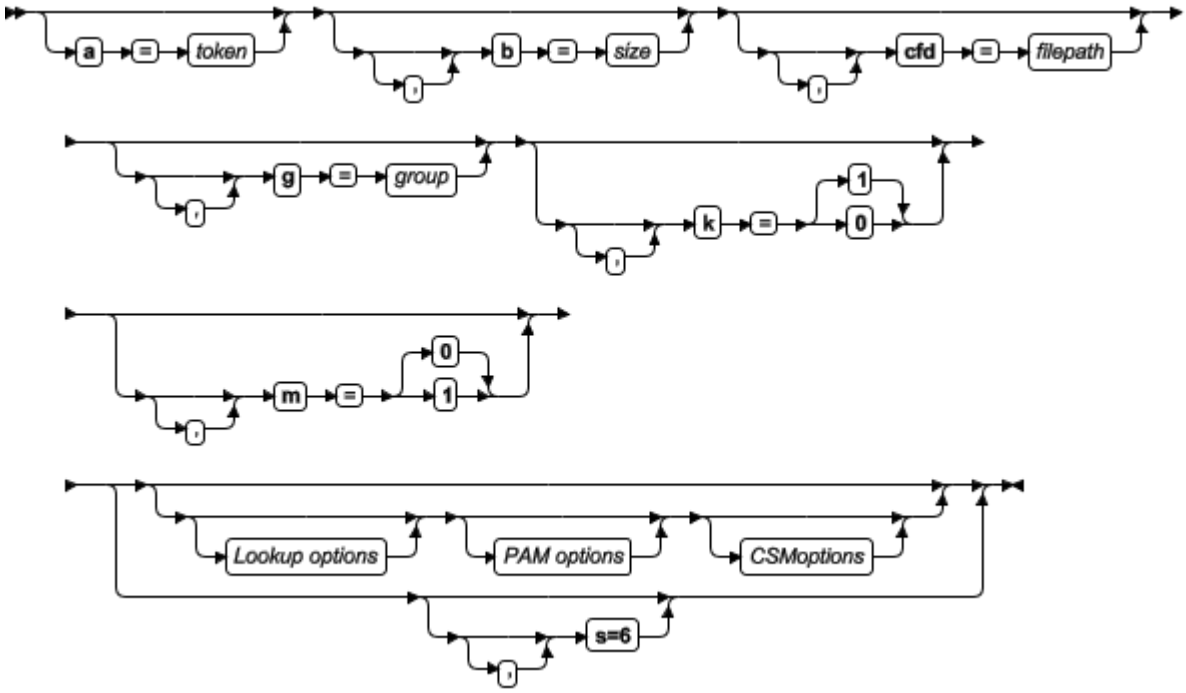
sqlhosts 文件和 SQLHOSTS 注册表键选项

您可以在 sqlhosts 文件或 SQLHOSTS 注册表键中包含服务器选项和组选项。

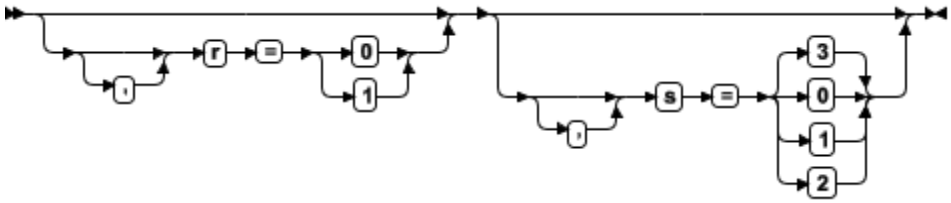
以下语法分段显示服务器选项。在服务器选项之后的部分中描述了组选项的语法分段。

重要： 选项必须以逗号分隔，但在每个 sqlhosts 条目中列出的第一个选项前面不能有逗号。

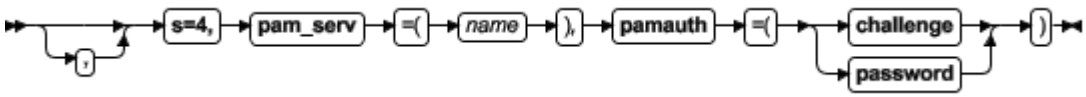
Server options



Lookup options



PAM options



CSM options

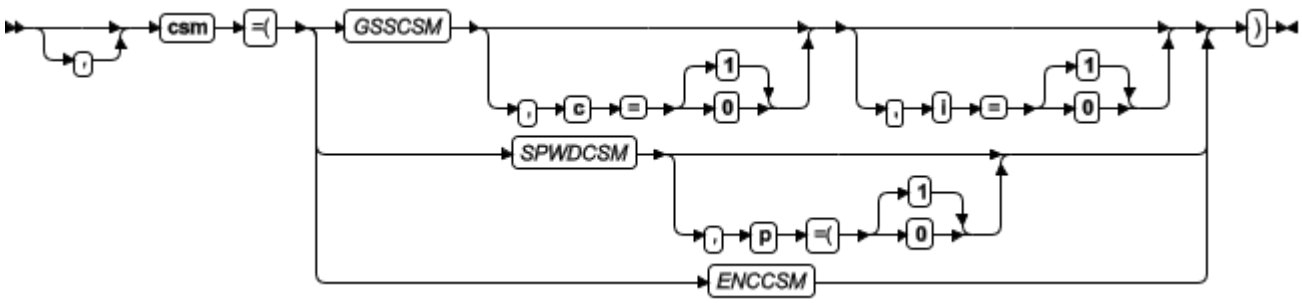


表 1. sqlhosts 文件和 SQLHOSTS 注册表键中的 Server 选项。

元素	用途	限制
a	存储连接到 GBase 8s Warehouse Accelerator 所需的认证令牌。此条目由 GBase 8s 在 GBase 8s Warehouse Accelerator 连接设置期间创建。	**重要：** 不要手动更改此选项。
b	指定用于 TCP/IP 连接的通信缓冲区空间大小（以字节为单位）。	支持的最大缓冲区大小为 32 KB。

元素	用途	限制
c	为支持单点登录 (SSO) 的类属安全性服务 CSM 启用机密性服务。 与 SSO 认证的用户之间传输的数据已加密，只能供使用授权凭证登录的用户查看。	• c=1：启用服务（缺省值） • c=0：禁用服务
cfd	指示共享内存和流管道连接中所用通信文件的存储位置。	cfd 路径的长度限制为 70 个字节。相对路径字节长度包括 $GBASEDBTDIR。
csm	针对使用通信支持模块 (CSM) 的每个数据库服务器描述 CSM。	必须在 concsm.cfg 文件中指定 CSM 条目。
ENCCSM	加密通信支持模块的名称。	必须在 concsm.cfg 文件中指定 ENCCSM。 不能将 ENCCSM 用于 • Enterprise Replication 和高可用性集群 • 多路复用连接 • 简单密码 CSM (SPWDCSM)
g	指定数据库服务器所属的组的名称。	该组必须已定义。
GSSCSM	用于单点登录 (SSO) 认证的类属安全性服务通信支持模块的名称。	必须在 concsm.cfg 文件中指定 GSSCSM。不能将其用于 Enterprise Replication 和高可用性集群。
i	为支持单点登录 (SSO) 的类属安全性服务 CSM 启用完整性服务。	• i=1：启用服务（缺省值） • i=0：禁用服务
k	启用网络服务以定期检查客户机与服务器之间的连接是否仍处于活动状态。 如果发现连接已中断，网络服务将释放资源。	仅可用于 TCP/IP 连接。
m	使数据库服务器可以创建多个数据库连接，而不会耗尽更多网络连接所需的额外计算机资源。	• 不支持多线程客户机连接、共享内存连接以及与下级数据库服务器的连接。 • 不支持 GBase 8s ESQL/C sqlbreak() 函数。 • 不能与 CSM 结合使用。
p	启用和禁用提供密码加密的简单密码 CSM。	• p=0：无需密码（缺省值） • p=1：需要密码
r	启用对操作系统安全文件查找的控制来	数据库服务器将忽略 r 设置。

元素	用途	限制
	控制客户机（用户）获取数据库服务器访问权的方式。s 选项标识数据库服务器端设置，而 r 选项则标识客户机端设置。	
s	启用对操作系统安全文件查找的控制来控制客户机（用户）获取数据库服务器访问权的方式。s 选项标识数据库服务器端设置，而 r 选项则标识客户机端设置。	客户机将忽略 s 设置。
pam_serv	提供数据库要使用的 PAM 服务的名称。	必须与 s=4 选项结合使用。
pamauth	描述 PAM 服务所用的授权方法。	必须与 s=4 选项结合使用。
SPWDCSM	简单密码通信支持模块的名称	必须在 concsm.cfg 文件中指定 SPWDCSM。 不能将 SPWDCSM 用于 • Enterprise Replication 和高可用性集群 • 多路复用连接 • 加密 CSM (ENCCSM)

以下语法分段显示 sqlhosts 文件中的 Group 选项。

组选项

表 2. sqlhosts 文件和 SQLHOSTS 注册表键中的 Group 选项。

元素	用途	限制
c	控制连接重定向。 指示客户机应用程序选择数据库服务器组中的数据库服务器或别名的顺序。	
e	指定标记数据库服务器组末尾的数据库服务器名称。	
i	为数据库服务器组指定标识号。	此标识必须是从 1 到 32767 的一个正整数，而且在您的网络环境中必须唯一。i 选项对于 Enterprise Replication 是必需的。

用途

在更改 sqlhosts 条目中的选项值时，这些更改将影响客户机应用程序建立的下一个连接。服务器将自动识别所做的任何更改。

数据库服务器将 options 条目作为一连串列进行求值。options 条目中的逗号或空格表示一个列的结束。客户机和数据库服务器将检查每一行以确定该选项是否受支持。

您可以在每个条目中组合多个选项，而且可以按任何顺序包含这些选项。options 条目的最大长度为 256 个字节。

注意： 不受支持或不正确的选项不会触发通知。

缓冲区选项 (b)

b 选项仅应用于使用 TCP/IP 连接协议的连接。其他类型的连接将忽略 b 选项。

可调整缓冲区大小，以便更有效地使用系统和网络资源；然而，如果设置过高的缓冲区大小，那么用户会因为不能分配任何内存而接收到连接拒绝错误。例如，如果您在拥有 1000 个用户的系统上设置 b=16000，那么系统可能会需要 16 兆字节的内存用于通信缓冲区。此设置可能会耗尽计算机的内存资源。使用 TCP/IP 的数据库服务器的缺省缓冲区大小是 4096 字节。

如果您的网络包括多种不同类型的计算机，更改通信缓冲区的大小时要小心。

提示： 使用通信缓冲区的缺省大小。如果您选择将缓冲区大小设置为其他值，请将客户机端通信缓冲区以及数据库服务器端通信缓冲区设置为相同的大小。

组连接重定向选项 (c)

c 选项仅针对分配给服务器组的服务器有效。

请将 c 选项用于：

- 平衡多个数据库服务器实例上的负载。
- 设置高可用性数据复制 (HDR)，以便在发生故障的情况下转移到备份数据库服务器。

表 3. 连接重定向选项的设置。

设置	结果
c=0	这是缺省设置。 客户机应用程序连接至 sqlhosts 信息中的服务器组中列出的第一个数据库服务器实例。如果客户机无法连接到第一个实例，它将尝试连接第二个实例，依此类推。
c=1	客户机应用程序会选择一个随机的起点来连接到服务器组中的某个数据库服务器实例。

通信文件目录选项 (cfd)

可使用通信文件目录选项，以将共享内存或流管道连接通信文件存储到新位置中。如果服务器和客户机位于不同位置中，那么必须为 GBase 8s 的非 root 安装指定通信文件目录选项；如果服务器和客户机位于同一位置中，指定此选项将提高系统性能。

cfd 选项可定义绝对路径或相对于 $GBASEDBTDIR 的路径以用于存储通信文件：

cfd=/*location* 定义绝对路径

- cfd=location 定义相对于 $GBASEDBTDIR 的路径
- cfd 路径的长度限制为 70 个字节。相对路径字节长度包括 $GBASEDBTDIR。

GBase 8s 的非 root 安装无权写入 /GBASEDBTTMP 目录，因此如果在 sqlhosts信息中未将任何通信文件目录指定为选项，那么共享内存和流管道连接通信文件将写入 $GBASEDBTDIR/etc 目录。

重要： 必须为 GBase 8s 的非 root 安装（其中服务器和客户机位于不同位置）定义此选项，否则连接将失败。

通信支持模块选项 (csm)

CSM 选项的格式为 csm=(*name*,*options*)

name 的值必须与 concsm.cfg 文件中的某个 name 条目相匹配。

sqlhosts 文件中定义的 GSM 选项会覆盖在 concsm.cfg 文件中指定的选项。 不能在 sqlhosts 信息中指定 CSM 加密选项。

如果不指定 csm 选项，那么数据库服务器会将缺省认证策略用于该数据库服务器。

注： 不推荐也无需将 s=7 选项用于单点登录 (SSO) CSM。

组末尾选项 (e)

如果没有为某个组指定 e 选项，但是所有 sqlhosts 条目都指定组或组成员，那么网络必须扫描整个文件。可以使用 e 选项指定服务器组的末尾，从而提高系统性能。网络层将扫描 sqlhosts 文件，直至读取到 e 选项指定的条目为止。

如果没有为组指定任何组末尾选项，那么组成员假定是连续的。达到不属于组的条目或处于文件的末尾时（以其中首先发生的任一种情况为准），即可确定该组的末尾。

在以下示例中，e 选项指定了条目 lx3，所以网络层不会扫描条目 lx4。

#dbservername	nettype	hostname	servicename	options
g_x1	group	-	-	i=10,e=lx3
lx1	onsoctcp	apollo11	9810	g=g_x1
lx2	onsoctcp	apollo12	9820	g=g_x1
lx3	onsoctcp	apollo13	9830	g=g_x1
lx4	onsoctcp	apollo14	9840	

保持活动选项 (k)

此选项启用网络服务以定期检查客户机与服务器之间的连接是否仍处于活动状态。 如果连接的接收端未在操作系统的参数所指定的时间内响应，那么网络服务将立即检测到断开的连接并释放资源。

表 4. 保持活动选项的设置

设置	结果
k=0	禁用此服务
k=1	启用此服务（缺省值）

多路复用选项 (m)

此选项使数据库服务器可以创建与客户机应用程序的多个数据库连接，而不会耗尽更多网络连接所需的额外计算机资源。启用此服务之后，必须重新启动服务器。

表 5. 多路复用选项的设置

设置	结果
m=0	禁用此服务（缺省值）
m=1	启用此服务

PAM 选项（pam_serv 和 pam_auth）

数据库服务器提供了用于为会话认证使用 PAM 的接口。 要配置此接口，请提供 PAM 服务名称和认证方法。认证可以是连接密码或要求用户回答问题的用户提问。GBase 8s PAM 认证将调用 pam_authenticate() 和 pam_acct_mgmt() 函数。

表 6. PAM 服务的设置

选项	描述	设置
pam_serv	数据库服务器正在使用的 PAM 服务的名称。	PAM 服务通常位于 /usr/lib/security 目录中，而参数在 /etc/pam.conf 文件中列出。 在 Linux™ 中，/etc/pam.conf 文件可替换为目录 /etc/pam.d，每个 PAM 服务在该目录内有一个文件。如果存在 /etc/pam.d，Linux 将忽略 /etc/pam.conf。
pamauth	PAM 服务使用的认证方法。 如果使用此认证方式，那么必须将应用程序设计为正确响应提问提示之后才能连接到数据库服务器。	pamauth=password 对认证使用连接请求密码 pamauth=challenge 认证要求用户正确回答问题或提示

hosts.equiv 和 rhosts 查找选项 (s)

通过这些安全性选项，您可以明确地启用或禁用对这两个文件或其中任何一个文件的使用。

表 7. hosts.equiv 和 rhosts 查找的设置。

设置	结果
s=0	禁用 hosts.equiv 查找。 禁用 rhosts 查找。

设置	结果
	仅接受带密码的入局连接。不能用于分布式数据库操作。
s=1	启用 hosts.equiv 查找。 禁用 rhosts 查找。
s=2	禁用 hosts.equiv 查找。 启用 rhosts 查找。 不能用于分布式数据库操作。
s=3	启用 hosts.equiv 查找。 启用 rhosts 查找。 （缺省值）

可信主机和可信用户查找选项 (s)

通过这些安全性选项，您可以明确地启用或禁用对这两个文件或其中任何一个文件的使用。

表 8. 可信主机和可信用户查找的设置。

设置	结果
s=0	在 hosts.equiv 中或 REMOTE_SERVER_CFG 配置参数指定的文件中禁用可信主机查找。 在 rhosts 文件中或 REMOTE_USERS_CFG 配置参数指定的文件中禁用可信用户查找。 仅接受带密码的入局连接。不能用于分布式数据库操作。
s=1	在 hosts.equiv 中或 REMOTE_SERVER_CFG 配置参数指定的文件中启用可信主机查找。 在 rhosts 文件中或 REMOTE_USERS_CFG 配置参数指定的文件中禁用可信用户查找。
s=2	在 hosts.equiv 中或 REMOTE_SERVER_CFG 配置参数指定的文件中禁用可信主机查找。 在 rhosts 文件中或 REMOTE_USERS_CFG 配置参数指定的文件中启用可信用户查找。 不能用于分布式数据库操作。
s=3	在 hosts.equiv 中或 REMOTE_SERVER_CFG 配置参数指定的文件中启用可信主机查找。 在 rhosts 文件中或 REMOTE_USERS_CFG 配置参数指定的文件中启用可信用户查找。 （缺省值）

集群的安全连接选项 (s=6)

sqlhosts 信息中的 s=6 选项确保集群服务器之间的连接可信。sqlhosts 信息中列出的安全端口只能用于集群通信。客户机应用程序不能连接到安全端口。

表 9. 集群的安全连接选项。

设置	结果
s=6	配置 Enterprise Replication 和高可用性连接安全性。不能与其他任何 s 选项结合使用。

netrc 查找选项 (r)

可使用 r 选项启用或禁用 netrc 查找。

表 10. netrc 查找选项的设置。

设置	结果
r=0	禁用 netrc 查找。
r=1	启用 netrc 查找（缺省值）

组信息

可在 sqlhosts 文件或 SQLHOSTS 注册表键中定义服务器组。创建服务器组时，可以将多个相关数据库服务器条目视为一个逻辑实体，用于建立或更改客户机/服务器连接。也可以使用组简化到数据库服务器的连接的重定向。

要将服务器组用于 Enterprise Replication，请使用 i 和 g 选项。所有参与复制的数据库服务器都必须是数据库服务器组的成员。 企业中的每个数据库服务器都必须有作为服务器组的唯一的标识。请确保在每个参与复制的数据库服务器上都正确设置了 sqlhosts 信息。

要将数据库服务器组用于高可用性数据复制 (HDR)，可使用 HDR 中的 c、e 和 g 选项。HDR 需要两个完全相同的系统。

可以在以下环境变量中或 SQL CONNECT 命令中使用数据库服务器组的名称来代替数据库服务器名称：

- 客户机应用程序的 GBASEDBTSERVER 环境变量的值可以是数据库服务器组的名称。但是，不能将数据库服务器组的名称用作数据库服务器或数据库服务器实用程序的 GBASEDBTSERVER 环境变量的值。
- DBPATH 环境变量的值可以包含数据库服务器组的名称作为数据库服务器名称。

在 sqlhosts 文件中创建数据库服务器组 (UNIX™)

通过向 sqlhosts 文件添加条目，可以定义数据库服务器组和组成员。

要在 sqlhosts 文件中创建数据库服务器组：

1. 添加一个条目来定义数据库服务器组：
   dbservername
   组的名称。该名称必须以小写字母开头，并且可以包含小写字母、数字和下划线 (_)

符号。

nettype
单词 group。

hostname
短划线 (-) 字符 (ASCII 45)，用于指示此字段值为空。

servicename
短划线 (-) 字符 (ASCII 45)，用于指示此字段值为空。

options
c、e 或 i 选项（根据情况而定）。

2. 为属于组的数据库服务器添加一个或多个条目。包括 g=*group* 选项。

示例

以下示例显示了 sqlhosts 文件中名为 **asia** 的数据库服务器组定义以及 **asia** 组成员：

```
#dbservername    nettype      hostname     servicename     options
asia             group        -            -               e=asia.4
asia.1           ontlitcp     node6        svc8            g=asia
asia.2           onsoctcp     node0        svc1            g=asia
asia.3           onsoctcp     node10       svc10           g=asia
asia.4           onsoctcp     node1        svc9            g=asia
```

TCP/IP 连接的备用方法

以下主题描述了绕过 TCP/IP 连接的端口和 IP 地址查找的某些方法。

TCP/IP 连接的 IP 地址

对于 TCP/IP 连接（TLI 和套接字），您可以使用 **hostname** 字段中的实际 IP 地址来代替主机名或 hosts 文件中找到的别名。以下示例显示 hosts 文件中的样本 IP 地址和主机。

```
#address        hostname    alias
555.12.12.12     smoke
98.555.43.21     odyssey
12.34.56.555     knight      sales
```

如果对该表中的 **knight** 使用 IP 地址，以下两个 sqlhosts 条目是等效的：

```
#dbservername    nettype      hostname      servicename     options
sales            ontlitcp     12.34.56.789  sales_ol
#dbservername    nettype      hostname      servicename     options
sales            ontlitcp     knight        sales_ol
```

使用 IP 地址在某些情况下可能加速连接时间。 但是，由于计算机通常由其主机名标识，因此在主机名字段中使用 IP 地址不便于标识与条目相关联的计算机。

UNIX： 您可以在 hosts 文件的网络地址字段中查找 IP 地址，或者您可以使用 UNIX ™ **arp** 或 **ypmatch** 命令查找 IP 地址。

TCP/IP 连接的通配符寻址

同时满足以下两个条件时，可以在 hosts 文件的 **hostname** 字段中使用通配符寻址：

- 您使用的是 TCP/IP 连接。
- 数据库服务器所在的计算机使用多块网络接口卡 (NIC)。

如果满足上述条件，您可以在数据库服务器使用的 **hostname** 字段中使用星号 (*) 作为*通配符*。当您在 **hostname** 字段中输入一个通配符时，数据库服务器可以在其主计算机上接受任何有效 IP 地址的连接。

每个 IP 地址都与唯一的主机名相关联。计算机使用多个 NIC 时（如下表中所示），hosts 文件必须针对每个接口卡具有一个条目。例如，使用两个 NIC 的 texas 计算机的 **hosts** 文件中可能包含以下条目。

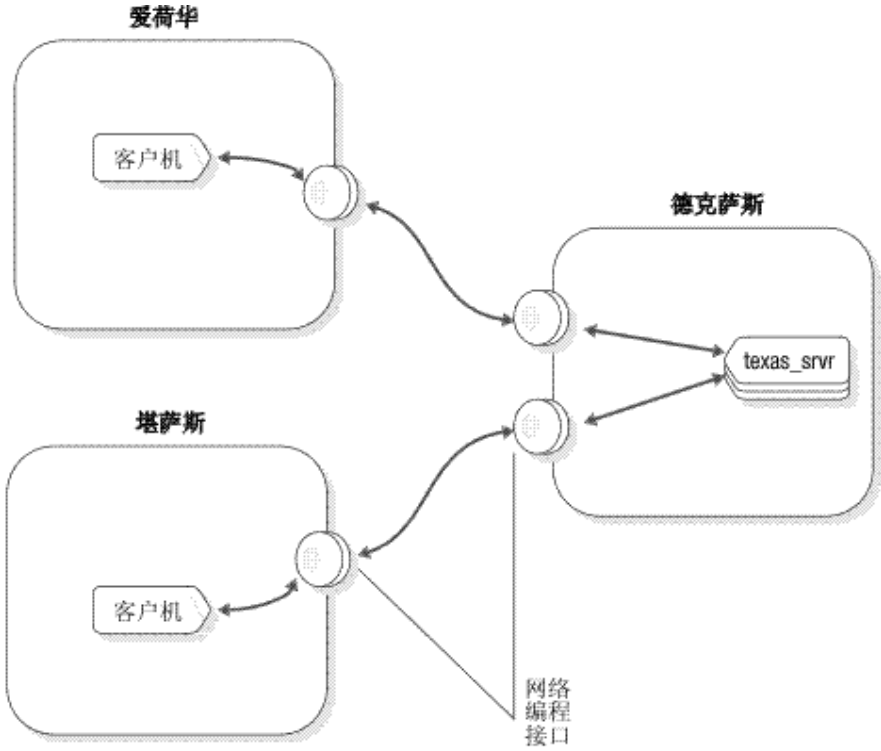
```
#address          hostname     alias
123.45.67.81       texas1
123.45.67.82       texas2
```

如果客户机应用程序和数据库服务器共享 sqlhosts 信息，那么可以在 **hostname** 字段中同时指定通配符和主机名或 IP 地址（例如，*texas1 或 *123.45.67.81）。 客户机应用程序忽略通配符并使用主机名（或 IP 地址）来建立连接，并且数据库服务器使用通配符从任何 IP 地址接受连接。

通配符格式允许数据库服务器的侦听线程等待每块有效的网络接口卡上使用相同服务端口号的客户机连接。但是，等待多个 IP 地址的连接可能比等待特定主机名或 IP 地址的连接所需处理器时间更长。

下图显示具有两块网络接口卡的计算机 **texas** 上的数据库服务器。这两个客户机站点使用不同的网卡与数据库服务器通信。

图: 使用多块网络接口卡



以下示例显示了 **texas_srvr** 数据库服务器可能的 sqlhosts 连接信息。

#dbservername	nettype	hostname	servicename	options
texas_srvr	ontlitcp	*texas1	pd1_on	
#dbservername	nettype	hostname	servicename	options
texas_srvr	ontlitcp	*123.45.67.81	pd1_on	
#dbservername	nettype	hostname	servicename	options
texas_srvr	ontlitcp	*texas2	pd1_on	
#dbservername	nettype	hostname	servicename	options
texas_srvr	ontlitcp	*123.45.67.82	pd1_on	
#dbservername	nettype	hostname	servicename	options
texas_srvr	ontlitcp	*	pd1_on	

如果连接信息与前面任意一行的内容一致，那么 **texas_srvr** 数据库服务器就可以接受来自其中任意一块网卡的客户机连接。数据库服务器将在 **hostname** 字段中查找通配符并忽略显式的主机名。

提示： 为了清晰简便地维护，在主机名字段中使用通配符时，请包含主机名（即，使用 *host，而不是 *）。

客户机应用程序使用的连接信息必须包含显式的主机名或 IP 地址。**iowa** 上的客户机应用程序可以使用下列主机名中的任何一个：texas1、*texas1、123.45.67.81 或 *123.45.67.81。如果 hostname 字段中有通配符 (*)，客户机应用程序会将其忽略。

kansas 上的客户机应用程序可以使用下列主机名中的任何一个：texas2、*texas2、123.45.67.82 或 *123.45.67.82。

TCP/IP 连接的端口号

对于 TCP/IP 网络协议，可以在服务名称字段中使用实际的 TCP 侦听端口号。

例如，如果在 **services** 文件中，sales 数据库服务器的端口号是 1543，那么可以在 sqlhosts 文件中写入一个条目，如下所示：

```
#dbservername    nettype      hostname     servicename     options
sales            ontlitcp     knight       1543
```

在有些情况下，使用实际端口号可能会节省建立连接的时间。但是，对于 hostname 字段中的 IP 地址，使用实际端口号可能不便于对连接信息的管理。

2.2.7 GBase 8s 对 IPv6 地址的支持

在所有平台上，GBase 8s 均可识别因特网协议 V6 (IPv6) 地址（长度为 128 位）和因特网协议 V4 (IPv4) 地址（长度为 32 位）。

自 GBase 8s V8.8 和 Client SDK 2.90.xC4 开始，数据库服务器会在启动时检查底层操作系统中是否支持 IPv6。如果支持 IPv6，那么将使用 IPv6。如果底层的操作系统不支持 IPv6，那么将使用 IPv4 地址。GBase 8s 和 Client SDK 从名称服务检索 IP 地址。

在同时带有 IPv4 和 IPv6 地址的主机上运行的 GBase 8s 可以按与在多宿主主机上运行的服务器相同的方式进行处理。可使用以下某种方法在同时带有IPv4 和 IPv6 地址的主机上配置 GBase 8s：

- 创建别名（使用 DBSERVERALIASES 配置参数），向其中某个别名指定 IPv6 地址而向另一个别名指定 IPv4 地址。
- 通过在 sqlhosts 文件中使用带通配符的主机名，指示 GBase 8s 在主机上配置的所有 IP 地址上进行侦听。

  例如：

```
#dbservername    nettype      hostname     servicename     options
olserver1        onsoctcp     *myhost      onservice1
```

如果已对主机配置了 IPv6 地址，那么 SQLHOSTS 文件中的 hostname 条目将映射到某个 IPv6 地址。如果未对主机配置 IPv6 地址，那么 hostname 条目将映射到 IPv4 地址。

禁用 IPv6 支持

GBase 8s 还提供了在 IPv4 环境中操作时禁用 IPv6 支持的方法。

要对所有数据库实例和客户机应用程序禁用 IPv6 支持，请执行以下操作：

- 创建空文件 $GBASEDBTDIR/etc/IFX_DISABLE_IPV6。

用户 **gbasedbt** 必须具有该文件的读许可权。该文件不可读写，也无需包含任何数据。

要对单个数据库实例或单个客户机应用程序禁用 IPv6 支持，请执行以下操作：

- 在数据库服务器实例上或在运行应用程序的工作站上，创建名为 IFX_DISABLE_IPV6 的环境变量，并将其值设置为 **yes**，如下所示：

```
IFX_DISABLE_IPV6=yes
```

2.2.8 与连通性相关的 ONCONFIG 参数

onconfig 文件中的某些配置参数用于指定与连通性相关的信息。

重新启动数据库服务器时，重新启动过程使用在这些配置参数中设置的值。

以下配置参数与连通性相关：

- DBSERVERNAME
- DBSERVERALIASES
- LIMITNUMSESSIONS
- NETTYPE
- NS_CACHE
- NUMFDSERVER
- HA_ALIAS

DBSERVERNAME 配置参数中设置的连接信息

客户机应用程序连接到数据库服务器时，必须指定该数据库服务器的名称。与指定的数据库服务器名称关联的 sqlhosts 信息描述了应用程序和数据库服务器之间的连接类型。

例如，要为数据库服务器指定名称 nyc_research，请在 onconfig文件设置 DBSERVERNAME 值：

```
DBSERVERNAME nyc_research
```

客户机应用程序将在下列某个地方指定数据库服务器的名称：

- GBASEDBTSERVER 环境变量中
- 在用于指定数据库环境的 SQL 语句（例如，CONNECT、DATABASE、CREATE TABLE 和 ALTER TABLE）中
- 在 DBPATH 环境变量中

DBSERVERNAME 必须指定数据库服务器名称或某个数据库服务器别名。该名称必须以小写字母开头并且可以包含其他小写字母、数字和下划线。该名称不得包含大写字符、字段定界符（空格或制表符）或换行符。基本 ASCII 代码集的其他字符未必可靠。例如，连接符或减号可能产生问题并且冒号可能不能可靠地使用。@ 字符是保留字符，用以从服务器分隔数据库（例如，dbase@server）。

对于 onimcsoc 或 onsoctcp 协议，可更新 DBSERVERNAME 配置参数，以便在 sqlhosts 信息中包含数据库服务器别名的多个侦听线程的数量，如下所示：

```
DBSERVERNAME name-number_of_multiple_listen_threads
```

您可以将 DBSERVERALIASES 连接配置为 SSL 连接，而且也可以组合使用 SSL 和非 SSL 连接。

DBSERVERALIASES 配置参数中设置的连接信息

通过 DBSERVERALIASES 配置参数，您可以为同一数据库服务器指定额外的数据库服务器名称。

最大的别名数为 32。以下示例显示 onconfig 配置文件中的条目，这些条目将三个数据库服务器名称指定给同一个数据库服务器实例。

```
DBSERVERNAME          sockets_srvr
DBSERVERALIASES       ipx_srvr,shm_srvr
```

因为每个数据库服务器名称具有对应的 sqlhosts 条目，您可以将多个连接类型与一个数据库服务器相关联。

```
shm_srvr        onipcshm      my_host          my_shm
sockets_srvr    onsoctcp      my_host          port1
ipx_srvr        ontlispx      nw_file_server   ipx_srvr
```

通过使用上一个示例中所示的 sqlhosts 文件，客户机应用程序可以使用以下语句连接到使用共享内存通信的数据库服务器：

```
CONNECT TO '@shm_srvr'
```

客户机应用程序可使用以下语句初始化与*同*一数据库服务器的 TCP/IP 套接字连接:

```
CONNECT TO '@sockets_srvr'
```

DBSERVERALIASES 必须以小写字母开头并且可以包含其他小写字母、数字和下划线。DBSERVERALIASES 不得包含大写字符、字段定界符（空格或制表符）或换行符。基本 ASCII 代码集的其他字符未必可靠。例如，连接符或减号可能产生问题并且冒号可能不能可靠地使用。@ 字符是保留字符，用以从服务器分隔数据库（例如，dbase@server）。

在前面的示例中，@shm_srvr 语句连接到该服务器上未识别的数据库；或者，也可以连接到 dbase1@shm_srvr。

对于 **onimcsoc** 或 **onsoctcp** 协议，可更新 DBSERVERALIASES 配置参数，以便在 **sqlhosts** 信息中包含数据库服务器别名的多个侦听线程的数量，如下所示：

```
DBSERVERALIASESname-number,name-number
```

您可以将 DBSERVERALIASES 连接配置为 SSL 连接，而且也可以组合使用 SSL 和非 SSL

连接。

LIMITNUMSESSIONS 配置参数中设置的连接信息

LIMITNUMSESSIONS 配置参数是一个可选参数，用于指定希望连接到 GBase 8s 的最大会话数。 如果指定了最大数，那么也可以指定在会话数接近最大数时是否要让 GBase 8s 向 online.log 文件打印消息。

针对服务器进行的分布式查询计数可能达到限制。

可能需要动态增加或临时关闭 LIMITNUMSESSIONS 配置参数，以便在数据库服务器即将达到限制时，允许管理实用程序运行。可么使用 gadmin -wf 或 gadmin -wm 来动态增加或关闭 LIMITNUMSESSIONS。

如果启用了 LIMITNUMSESSIONS 配置参数，但是由于此限制导致会话受限，那么连接到任何数据库的普通用户线程计数和 DBSA 用户线程计数都可能达到该限制。但是，即使达到了该限制之后，也允许 DBSA 用户连接到服务器。

LIMITNUMSESSIONS 配置参数不应用作遵守许可协议的方法。

示例

以下示例指定您希望连接到数据库服务器的最大会话数是 100，并且在连接的会话数接近 100 时显示警告消息：

LIMITNUMSESSIONS 100,1

NETTYPE 配置参数中设置的连接信息

NETTYPE 配置参数使您调整数据库服务器用于通信的虚拟处理器的数量和类型。 每种类型的网络连接（例如，ipcshm 或 soctcp）可以在配置文件中具有单独的 NETTYPE 条目。

建议： 尽管 NETTYPE 参数不是必需参数，但是如果使用两个或更多连接类型，那么必须设置 NETTYPE。当数据库服务器持续运行了一段时间后，您可以使用 NETTYPE 配置参数来调整数据库服务器以获得更好的性能。

有关 NETTYPE 的更多信息，请参阅网络虚拟处理器。有关 NETTYPE 配置参数的信息，请参阅《GBase 8s 管理员参考》。

NS_CACHE 配置参数中设置的名称服务最大保留时间

NS_CACHE 配置参数定义以下对象中单个条目的最大保留时间：主机名/IP 地址高速缓存、服务高速缓存、用户高速缓存和组高速缓存。如果指定最大保留时间，数据库服务器可从高速缓存获取主机、服务、用户和组数据库服务器的信息。

在为特定高速缓存配置的时间之后或者重新配置该时间时，每个高速缓存条目都将到期。

网络名服务提供者（如 DNS）通常位于远程计算机上。要避免花费时间从网络名服务提供者返回信息，可使用 NS_CACHE 配置参数来指定最大保留时间，用于从某一个内部高速缓存中获取信息。GBase 8s 将在高速缓存中查找信息。如果该处没有此类信息，数据库服务

器将查询操作系统以获取这些信息。

使用 GBase 8s 名称服务高速缓存机制可避免许多在操作系统中进行的查找，从而在可配置的时间内保留并复用检索到的每个信息段。

服务器从高速缓存获取信息的速度比查询操作系统更快。但是，如果将保留时间设置为 0 来禁用一个或多个这类高速缓存，那么数据库服务器将查询操作系统以获取主机、服务、用户或组信息。

作为 DBA，您可能希望在以下情况下修改 NS_CACHE 配置参数的设置：网络名服务提供程序在远程计算机上运行，或 MSC VP 在运行时的处理器使用量很大。

例如，可运行 gstat -g glo 命令，以在输出的 Individual virtual processors 部分中检查 msc VP 的使用情况。在以下输出样本中，usercpu 和 syscpu 列中显示的 msc 处理器使用量非常高。如果怀疑使用量高是因为 DNS 调用花费了过多时间，那么可使用操作系统命令来确定高使用量，然后修改 NS_CACHE 配置参数的设置。

Individual virtual processors:

vp	pid	class	usercpu	syscpu	total	Thread	Eff
1	2036	cpu	76.95	7.14	84.09	99.08	84%
2	2149	adm	0.00	0.00	0.00	0.00	0%
3	2151	LIC	0.00	0.00	0.00	0.00	0%
4	2260	lio	0.00	0.00	0.00	0.03	0%
5	2442	pio	0.00	0.00	0.00	0.00	0%
6	2443	aio	0.00	0.01	0.01	0.11	8%
7	2444	msc	14.18	14.64	28.82	199.91	14%
8	2446	fifo	0.00	0.00	0.00	0.00	0%

您还可能希望在以下情况下指定 NS_CACHE 信息：操作系统没有名称服务 (NS) 高速缓存，或禁用了操作系统 NS 高速缓存。

示例

要将主机和服务连接的最大保留时间定义为 600 秒，并对用户和组数据库服务器连接禁用最大保留限制，请指定：

```
NS_CACHE host=600,service=600,user=0,group=0
```

NUMFDSERVERS 配置参数中设置的连接信息

对于 UNIX™ 上的网络连接，可以使用 NUMFDSERVERS 配置参数指定处理在 GBase 8s 虚拟处理器 (VP) 之间迁移的网络连接所需的最大轮询线程数。

如果 GBase 8s 的新连接和断开连接请求的比率极高，或者发现了网络共享文件 (NSF) 锁定之间存在大量争用，那么指定 NUMFDSERVERS 信息非常有用。

HA_ALIAS 配置参数中设置的连接信息

HA_ALIAS 配置参数是一个可选参数，用于定义辅助服务器的网络别名。在 gadmin -d 命令中指定了辅助服务器时，将使用 HA_ALIAS 配置参数指定的网络别名。

设置数据库服务器的 HA_ALIAS 配置参数后，与其他高可用性集群节点的所有服务器间通信均通过指定的网络别名来进行。

如果高可用性集群中的主服务器发生故障，连接管理器将确定要提升为主服务器的辅助服务器。如果设置了辅助服务器的 HA_ALIAS 配置参数，那么 HA_ALIAS 网络别名用于识别新的主服务器。HA_ALIAS 配置参数只影响 RS 辅助服务器和 SD 辅助服务器类型。

HA_ALIAS 配置参数的值必须是 DBSERVERNAME 或 DBSERVERALIASES 配置参数中指定的其中一个名称值。 网络别名的连接类型必须是 TCP 网络协议。

2.2.9 网络连接的环境变量

GBASEDBTCONTIME（连接时间）和 GBASEDBTCONRETRY（连接重试）环境变量是在客户机尝试连接到数据库服务器时会影响客户机行为的客户机环境变量。使用这些环境变量可使网络流量繁忙所引起的连接错误最小化。

如果客户机应用程序显式连接到共享内存段，那么您可能需要设置 GBASEDBTSHMBASE（共享内存库）。

可使用 GBASEDBTSERVER 环境变量来指定客户机连接到的缺省数据库服务器名称。

2.2.10 客户机/服务器配置的示例

下面几个部分显示多个客户机/服务器连接的正确的 sqlhosts 条目。可以假定已正确准备了网络配置文件 hosts 和 services（即使没有明确提到这些文件）。包括以下示例：

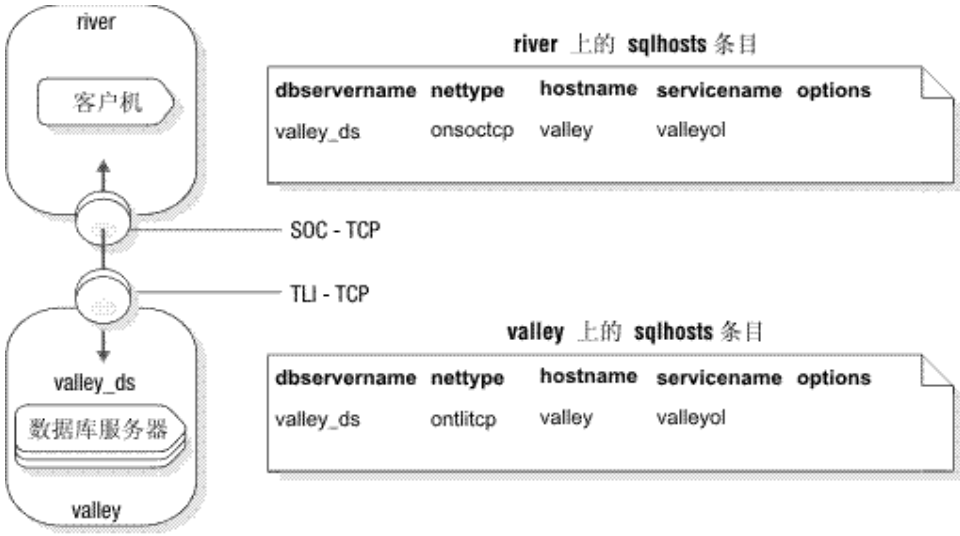
- 使用网络连接
- 使用多种连接类型
- 访问多个数据库服务器

共享内存和本地回送连接的示例可以在共享内存和本地回送连接的说明中找到。

网络连接

下图显示的配置中，客户机应用程序位于主机 **river** 上，而数据库服务器位于主机 **valley** 上。

图: 网络客户机/服务器配置示例



在这两台计算机上定义了 **valley_ds** 数据库服务器的 sqlhosts 条目。

两个计算机在同一个 TCP/IP 网络上，但是主机 **river** 对于其网络编程接口使用套接字，而主机 **valley** 对于其网络编程接口使用 TLI。**nettype** 字段必须反映 sqlhosts 所在计算机使用的网络编程接口类型。在此示例中，主机 **river** 上 **valley_ds** 数据库服务器的 **nettype** 字段是 onsoctcp，而主机 **valley** 上 **valley_ds** 数据库服务器的 **nettype** 字段是 ontlitcp。

多种连接类型

数据库服务器的单个实例可以提供多种类型的连接。下图说明了此类配置。数据库服务器位于主机 **river** 上。由于共享内存速度快，所以客户机 A 通过共享内存连接连接到数据库服务器。客户机 B 必须使用网络连接，因为客户机和服务器在不同的计算机上。

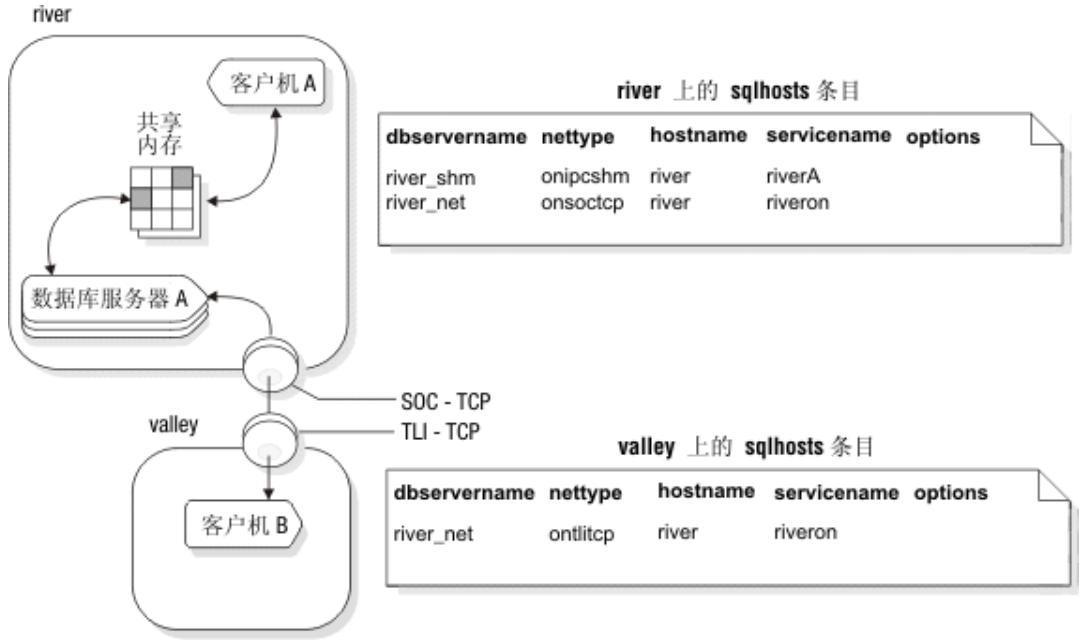
当您希望数据库服务器接受多种类型的连接时，您必须进行下列操作：

- 在 onconfig 文件中添加 DBSERVERNAME 和 DBSERVERALIASES 条目。
- 针对每个数据库服务器/连接类型对，添加 sqlhosts 条目。

对于下图中的配置，数据库服务器有两个数据库服务器名称：**river_net** 和 **river_shm**。onconfig 文件包含以下条目：

```
DBSERVERNAME          river_net
DBSERVERALIASES       river_shm
```

图：使用多种连接类型的 UNIX™ 客户机/服务器配置的示例



客户机应用程序使用的数据库服务器名称可用于确定使用的连接类型。客户机 A 使用以下语句连接到数据库服务器：

```
CONNECT TO '@river_shm'
```

在 sqlhosts 文件中，与名称 **river_shm** 关联的 **nettype** 指定共享内存连接，因此该连接是共享内存连接。

客户机 B 可使用以下语句连接到数据库服务器：

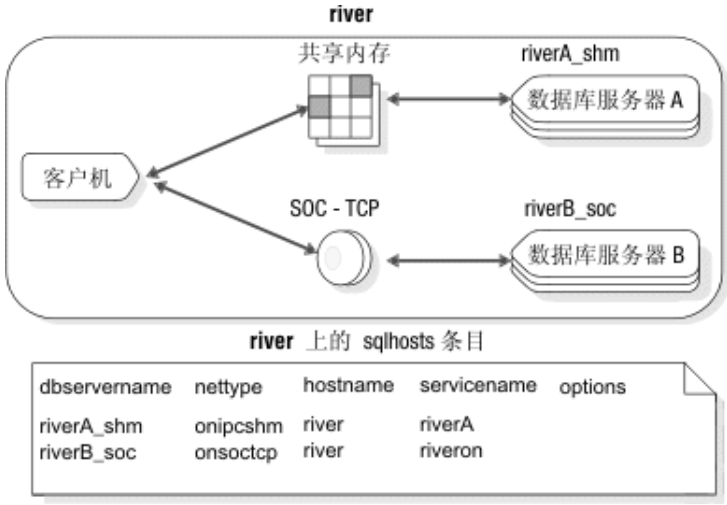
```
CONNECT TO '@river_net'
```

在 sqlhosts 文件中，与 **river_net** 相关联的 **nettype** 值指定网络 (TCP/IP) 连接，因此客户机 B 使用网络连接。

访问多个数据库服务器

下图显示了主机 **river** 上两个数据库服务器的配置。当一台计算机上有多个数据库服务器处于活动状态时，称为*多处驻留*。

图: UNIX™ 上的多个数据库服务器



对于前面示例中的配置，您必须准备两个 onconfig 文件，一个用于数据库服务器 A，另一个用于数据库服务器 B。sqlhosts 文件包含这两个数据库服务器的连接信息。

数据库服务器 A 的 onconfig 文件包含以下行：

```
DBSERVERNAME          riverA_shm
```

数据库服务器 B 的 onconfig 文件包含以下行：

```
DBSERVERNAME          riverB_soc
```

2.2.11 GBase 8s MaxConnect

GBase 8s MaxConnect 是 UNIX™ 上 GBase 8s 数据库服务器环境的网络产品。GBase 8s MaxConnect 管理大量（从数百到数万）客户机/服务器连接。GBase 8s MaxConnect 可多路复用连接，这样客户机连接与数据库连接之比可达到 200:1 或更高。GBase 8s MaxConnect 将系统可伸缩性提高至数千个连接并节省了系统资源，从而缩短响应时间并降低处理器需求。GBase 8s MaxConnect 最适合用于 OLTP 数据传输，不应该用于大型多媒体数据传输。

将 GBase 8s MaxConnect 独立于 GBase 8s 数据库服务器及客户机应用程序进行安装。为了获得最佳性能，请将 GBase 8s MaxConnect 安装在 GBase 8s 客户机所连接的独立计算机上或安装在客户机应用程序服务器上。您可以在下列配置中安装 GBase 8s MaxConnect：

- 在 GBase 8s 客户机所连接的专用服务器上
- 在客户机应用程序服务器上
- 在数据库服务器计算机上

多路复用连接的两个协议（ontliimc 和 onsocimc）可供 GBase 8s MaxConnect 用户使用。您可以在下列两种配置中使用 ontliimc 和 onsocimc 协议：

- 将 GBase 8s MaxConnect 连接到数据库服务器。

  在此配置中，客户机连接使用多路复用并且使用信息包聚集。

- 直接将客户机应用程序连接到数据库服务器而不经过 GBase 8s MaxConnect。

在此配置中，客户机将不能利用连接多路复用或信息包聚集的好处。当客户机应用程序传送简单或智能大对象数据时选择此配置，因为直接连接到数据库服务器是最好的做法。

2.3 数据库服务器初始化

数据库服务器需要磁盘空间初始化和共享内存初始化。

2.3.1 初始化类型

数据库服务器的初始化包含两种相关活动：共享内存初始化和磁盘空间初始化。

*共享内存初始化*或启动服务器将确定数据库服务器共享内存的如下内容：内部表、缓冲区和共享内存通信区。共享内存将在每次启动数据库服务器时初始化。从命令行使用 oninit 实用程序来初始化数据库服务器共享内存，并使数据库服务器联机。

共享内存初始化还会在您重新启动数据库服务器时发生。

区分共享内存初始化与磁盘空间初始化的关键差异：

共享内存初始化对磁盘空间分配或布局没有影响。不会删除任何数据。

*磁盘空间初始化*使用存储在配置文件中的值以在磁盘上创建根数据库空间的初始块。当您初始化磁盘空间时，作为过程的一部分，数据库服务器将自动初始化共享内存。磁盘空间将在第一次启动数据库服务器时初始化。之后，它将只有在冷复原期间或在数据库服务器管理员的请求下才初始化。

警告： 当您初始化磁盘空间时，您将覆盖该磁盘空间上的所有内容。如果您重新初始化现有数据库服务器的磁盘空间，那么较早数据库服务器中的所有数据都将无法访问，并且实际上会将其删除。

2.3.2 初始化磁盘空间

第一次启动数据库服务器或者希望除去所有数据库空间及其关联数据时，初始化根数据库空间的磁盘空间。安装数据库服务器并选择初始化数据库服务器的新实例时，数据库服务器会自动初始化。

警告： 初始化数据库服务器时，将删除数据库服务器磁盘空间中的所有现有数据。

先决条件：

- **UNIX™、Linux™ 或 Mac OS X：**您必须以 **root** 或 **gbasedbt** 用户身份登录。

对数据库服务器已经在使用的根数据库空间进行重新初始化之前：

- 通过执行 0 级备份，备份现有数据。
- 通过运行 gadmin -k 命令，停止数据库服务器。
- 将 FULL_DISK_INIT 配置参数设置为 1。

要初始化数据库服务器：

UNIX、Linux 或 Mac OS X：运行 oninit -iy 命令。

初始化完成后，可以执行 0 级复原。

2.3.3 初始化过程

启动数据库服务器或初始化磁盘空间时，数据库服务器会执行一组步骤。可以在消息日志中查看每个步骤的结果。

磁盘空间初始化总是包含共享内存的初始化。然而，有些通常在共享内存初始化期间发生的活动（如记录配置更改）在磁盘初始化期间是不需要的，因为这些活动与新初始化的磁盘无关。

下表显示了在两种类型的初始化期间完成的主要任务。以下各节对每个步骤进行了说明。

表 1. 初始化步骤

共享内存初始化	**磁盘初始化**
处理配置文件。	处理配置文件。
创建共享内存段。	创建共享内存段。
初始化共享内存结构。	初始化共享内存结构。
	初始化磁盘空间。
启动所有必需的虚拟处理器。	启动所有必需的虚拟处理器。
进行必要的转换。	
启动®快速恢复。	
启动检查点。	启动检查点。
记录配置更改。	
更新 oncfg_servername.servernum 文件。	更新 oncfg_servername.servernum 文件。
更改到静默方式。	更改到静默方式。
删除临时表空间（可选）。	
设置强制的驻留（如果需要）。	设置强制的驻留（如果指定）。
更改到联机方式并将控制权交还用户。	更改到联机方式并将控制权交还用户。
如果 SMI 表不是最新的，那么更新这些表。	创建包含 SMI 表的 **sysmaster** 数据库。
	创建 **sysutils** 数据库。
	创建 **sysuser** 数据库
	创建 **sysadmin** 数据库
监视每个检查点的最大用户连接数。	监视每个检查点的最大用户连接数。

初始化期间使用的配置文件

数据库服务器使用配置参数在初始化和重新启动期间分配共享内存段。 如果您修改了共享内存配置参数，您必须关闭并重新启动数据库服务器才能使更改生效。

ONCONFIG 环境变量、onconfig 文件和 onconfig.std 模板存储在 $GBASEDBTDIR/etc（在 UNIX™ 上）中。缺省配置文件将用作模板而不是功能性配置。

在您初始化或重新启动数据库服务器之前，始终设置 ONCONFIG 环境变量，此变量指定包含配置参数的 onconfig 文件。确保您还具有 onconfig.std 文件。 如果缺少 onconfig.std 文件，服务器将无法初始化。

初始化期间，数据库服务器将在以下文件中查找配置值：

- 如果已设置 ONCONFIG 环境变量，那么数据库服务器将从 onconfig 文件中读取值。

  如果已设置 ONCONFIG 环境变量，但数据库服务器不能访问指定的 onconfig 文件，那么服务器将返回错误消息。

- 如果未设置 ONCONFIG 环境变量，那么数据库服务器将从 onconfig 文件中读取值。

如果在 onconfig 文件中省略了任何配置参数，那么数据库服务器将使用服务器中构建的缺省值。如果在 onconfig 文件中省略了任何配置参数，那么数据库服务器将从 $GBASEDBTDIR/etc/onconfig.std 文件中读取配置值。

重新启动进程将当前配置文件中的值与以前的值（如果有的话）相比较，以前的值存储在根数据库空间保留页面 PAGE_CONFIG 中。如果存在差异，那么当重新启动数据库服务器时，数据库服务器将使用当前 onconfig 配置文件中的值。

创建共享内存部分

数据库服务器可使用配置值来计算数据库服务器常驻共享内存的所需大小。此外，数据库服务器将计算内部值的附加配置要求。将计算和存储开销的空间要求。

要创建共享内存，数据库服务器将从操作系统中获得用于三种不同类型的内存的共享内存空间：

- 常驻部分，用于数据缓冲区和内部表
- 虚拟部分，用于大多数系统和用户会话内存要求
- IPC 通信部分，用于 IPC 通信

仅当您配置 IPC 共享内存连接时，数据库服务器才分配共享内存的该部分。

接着，数据库服务器会将共享内存段连接到其虚拟地址空间然后初始化共享内存结构。 有关共享内存结构的更多信息，请参阅共享内存的虚拟部分。

在完成了初始化并且数据库服务器已经开始运行之后，它可以根据需要创建额外的共享内存段。数据库服务器将以页大小的增量方式创建段。

初始化或重新启动共享内存

在数据库服务器连接到共享内存之后，它将清除未初始化数据的共享内存空间。接着，数据库服务器将设计共享内存头信息，并在共享内存结构中初始化数据。数据库服务器将安排逻辑日志缓冲区所需的空间，初始化这些结构，然后将形成逻辑日志缓冲区的三个单独的缓冲区链接在一起。有关这些结构的更多信息，请参阅《GBase 8s 管理员参考》中的 gstat 实用程序部分。

在数据库服务器重新映射共享内存空间之后，它将新的开始地址以及每个结构的大小注册到新的共享内存头中。

共享内存初始化期间，磁盘结构和磁盘布局将不受影响。数据库服务器将从磁盘中读取必需的地址信息（如逻辑和物理日志的位置），然后使用此信息更新共享内存中的指针。

初始化磁盘空间

该过程仅在磁盘空间初始化期间才执行，而不是在数据库服务器重新启动时执行。初始化共享内存结构之后，数据库服务器将开始初始化磁盘。数据库服务器初始化保留在磁盘的根数据库空间中的所有保留页面，并将页面 0 控制信息写入该磁盘。

FULL_DISK_INIT 配置参数指定当根路径位置（第一个块位置的第一个页面处）有页面 0 时，oninit -i 是否可在实例上运行。使用此配置参数可防止意外重新初始化现有服务器实例的磁盘。

FULL_DISK_INIT 配置参数的缺省设置为 0。如果此配置参数设置为 0，那么仅当根路径位置没有页面 0 时，oninit -i 命令才能运行。

如果根路径位置有页面 0，那么仅当 FULL_DISK_INIT 配置参数设置为 1 时，才会执行初始化。初始化之后，数据库服务器会自动将 FULL_DISK_INIT 配置参数重置为 0。

启动所有必需的虚拟处理器

数据库服务器启动其所需的所有虚拟处理器。onconfig 文件中的参数影响将要启动哪些处理器。例如，NETTYPE 参数可以影响为建立连接而启动的处理器的数量和类型。有关虚拟处理器的更多信息，请参阅虚拟处理器。

进行必要的转换

数据库服务器将检查其内部文件。如果这些文件是来自较早版本的，那么该它会将这些文件更新为当前格式。

启动快速恢复

数据库服务器检查是否需要快速恢复，如果需要则启动快速恢复。有关快速恢复的更多信息，请参阅快速恢复。

快速恢复不会在磁盘空间初始化期间执行，因为还没有任何要恢复的内容。

启动检查点

在快速恢复完成后，数据库服务器将执行一个检查点来验证是否所有已恢复的事务都已清空到磁盘，以便不重复执行这些事务。

作为检查点过程的一部件，数据库服务器将在消息日志中写入检查点完成的消息。有关检查点的更多信息，请参阅检查点。

此时数据库服务器是移动到静默方式还是联机方式，取决于您如何启动初始化或数据库服务器重新启动进程。

记录配置更改

数据库将存储在配置文件中的当前值与存储在根数据库空间保留页面 PAGE_CONFIG 中的以前的值相比较。当存在差异时，数据库服务器会将消息中的两个值（旧的和新的）都记录到消息日志中。

该任务不会在磁盘空间初始化或重新启动时执行。

创建 oncfg_servername.servernum 文件

数据库服务器将创建 oncfg_*servername.servernum*文件，并在每次添加或删除数据库空间、Blob 空间、逻辑日志文件或块时更新该文件。无需以任何方式处理该文件，但是可以发现在 $GBASEDBTDIR/etc 目录（在 UNIX™ 上）中列出了该文件。数据库服务器在用于抢救逻辑日志的整个系统复原期间，使用 oncfg_*servername.servernum* 文件。

有关 oncfg_*servername.servernum* 文件的更多信息，请参阅《GBase 8s 管理员参考》中有关数据库服务器使用的文件的章节。

删除临时表空间

数据库服务器将在所有数据库空间中搜索临时表空间。（如果您使用 **oninit** 的 -p 选项来初始化数据库服务器，那么数据库服务器会跳过此步骤。）这些临时表空间（如果有的话）是用户进程所留下的表空间，这些用户进程过早终止而未能执行适当的清除。数据库服务器将删除任何临时表空间并回收磁盘空间。有关临时表空间的更多信息，请参阅临时表。

该任务在数据库服务器重新启动时执行；它在磁盘空间初始化期间不执行。

设置强制驻留（如果指定）

如果 RESIDENT 配置参数的值是 -1 或是大于 0 的数，那么数据库服务器将尝试强制实施共享内存的驻留状态。如果主机系统不支持强制的驻留，那么初始化过程将继续。驻留未强制执行，所以数据库服务器将向消息日志发送错误消息。有关 RESIDENT 配置参数的更

多信息，请参阅《GBase 8s 管理员参考》。

将控制权交还用户

仅当发生了初始化而非数据库服务器重新启动时，数据库服务器才会将 GBase 8s Database Server initialized - complete disk initialized 消息写入消息日志。数据库服务器还将动态分配虚拟共享内存段。

此时，控制权被交还用户。任何由初始化过程生成的错误消息都将在下列位置显示：

- 命令行
- 数据库服务器消息日志文件，由 MSGPATH 配置参数指定。

  有关 MSGPATH 参数的更多信息，请参阅《GBase 8s 管理员参考》。
- Server Administrator (ISA) 的**摘要**部分

可以使用oninit -w 实用程序，迫使服务器在可配置的超时内返回到命令提示符。oninit -w 实用程序对于故障诊断初始化故障很用的。 有关 oninit 的语法和信息，请参阅《GBase 8s 管理员参考》。

创建 sysmaster 数据库并准备 SMI 表

尽管数据库服务器已将控制权交还给了用户，它还是未完成工作。数据库服务器现在检查系统监视接口（SMI）表。 如果 SMI 表不是当前表，那么数据库服务器将更新这些表。 如果 SMI 表不存在（当初始化磁盘时发生该情况），那么数据库服务器将创建这些表。当数据库服务器构建了 SMI 表之后，它将消息 sysmaster 数据库已成功构建放入消息日志中。数据库服务器还会在转换和还原期间重新创建 sysmaster 数据库。有关 SMI 表的更多信息，请参阅《GBase 8s 管理员参考》中有关 sysmaster 数据库的章节。

如果您在数据库服务器完成构建 SMI 表之前关闭了该数据库服务器，那么构建这些表的进程将停止。这种情况不会损坏数据库服务器。数据库服务器在您下次使数据库服务器联机时构建 SMI 表。但是，如果您不允许这些 SMI 表完成构建，您将无法运行对于那些表的任何查询，并且您也无法使用 ON-Bar 进行备份。

完成创建 SMI 表后，即可使用数据库服务器了。数据库服务器将运行直到您将其停止或出现可能发生的故障。

建议：不要尝试通过停止虚拟处理器或终止另一个数据库服务器进程来停止数据库服务器。 有关更多信息，请参阅启动和停止虚拟处理器。

创建 sysutils 数据库

数据库服务器将在磁盘初始化、转换或还原期间删除并重新创建 sysutils 数据库。ON-Bar 将在 sysutils 数据库中存储备份与复原信息。 请稍候，直到在消息日志中显示消息 sysutils database built successfully。有关更多信息，请参阅《GBase 8s 备份与复原指南》。

创建 sysuser 数据库

sysuser 数据库用于 GBase 8s 服务器与服务器通信中的可插式认证模块 (PAM) 认证。

创建 sysadmin 数据库

sysadmin 数据库提供了 GBase 8s 中的远程管理和调度程序 API 功能部件。

监视最大用户连接数

在每个检查点上，数据库服务器都在消息日志中打印最大的用户连接数：maximum server connections *number*。可以监视自从上次重新启动或磁盘初始化以来已经连接到数据库服务器的用户数。

当客户重新初始化数据库服务器时，显示的数量将复位。

2.3.4 数据库服务器运行方式

可以通过从命令行运行 gstat 实用程序来确定当前的数据库服务器方式。gstat 头将显示该方式。

下表显示了数据库服务器的主要运行方式。

表 1. 运行方式

运行方式	描述	允许访问的用户
脱机方式	数据库服务器未在运行。共享内存未被分配。	只有管理员（用户 **gbasedbt**）可以从此方式更改为其他方式。
静默方式	数据库服务器进程正在运行并且共享内存资源已被分配。 管理员使用此方式执行不需要执行 SQL 和 DDL 语句的维护功能。	只有管理员（用户 **gbasedbt**）可以访问数据库服务器。 其他用户可查看数据库服务器状态信息，但他们不能访问数据库服务器。
管理方式	此方式是一种介于静默方式和联机方式之间的中间方式。 管理员使用此方式执行所有维护任务，其中包括需要执行 SQL 和 DDL 语句的任务。管理员还可以执行在联机方式中可用的所有其他功能。	以下用户可在管理方式下连接到数据库服务器： • **gbasedbt** 用户 • 具有 DBSA 角色的用户 如果您想要属于 DBSA 组成员的用户（除用户 **gbasedbt** 外）以管理方式连接到数据库服务器，请将 ADMIN_USER_MODE_WITH_DBSA 配置参数设置为 1。 • 具有管理方式访问权限的一个或

运行方式	描述	允许访问的用户
		多个用户 用户 gbasedbt 或 DBSA 可通过 gadmin -j 命令、oninit -U 命令或 ADMIN_MODE_USERS 配置参数动态地授予一个或多个特定用户以管理方式连接到数据库服务器的能力。 其他用户可查看数据库服务器状态信息，但他们不能访问数据库服务器。
联机方式	这是数据库服务器的普通运行方式。	任何授权用户都可以与数据库服务器连接并执行所有数据库活动。 用户 gbasedbt 或用户 root 可使用命令行实用程序更改许多服务器 ONCONFIG 参数值。

此外，数据库服务器也可以处于下列方式中的一种：

- *只读方式*由数据复制环境中的辅助数据库服务器使用。应用程序可以查询处于只读方式下的辅助数据库服务器，但该应用程序不能写入只读数据库。
- *恢复方式*是过渡的。它将在数据库服务器从系统归档或系统复原中执行一个或多个快速恢复时发生。恢复将在从脱机方式更改到静默方式时发生。
- *关闭方式*是过渡的。它将在数据库服务器从联机方式转到静默方式或从联机（或静默）方式转到脱机方式时发生。当前用户可以访问系统，但不允许任何新用户访问。

当关闭方式启动后，无法将其取消。

2.3.5 更改数据库服务器运行方式

本节描述如何使用 oninit 和 gadmin 实用程序将一种数据库服务器运行方式更改为另一种方式。它还包含有关使用 ADMIN_MODE_USERS 配置参数以指定哪些用户可以采用管理方式连接服务器的信息。

本节描述如何使用 oninit 和 gadmin 实用程序以及使用 ISA 从一种数据库服务器运行方式更改到另一种方式。它还包含有关使用 ADMIN_MODE_USERS 配置参数以指定哪些用户可以采用管理方式连接服务器的信息。

提示： 更改数据库服务器的方式之后，请运行 gstat 命令以验证当前服务器状态。

允许更改方式的用户

仅适用于 UNIX™

以 root 或 gbasedbt 身份登录的用户以及 DBSA 组的成员可以更改数据库服务器的操作方式。

如果您希望属于 DBSA 组的用户以管理方式连接到数据库服务器，请将 ADMIN_USER_MODE_WITH_DBSA 配置参数设置为 1。如果该参数设置为 0，那么访问将仅限于用户 gbasedbt。如果 $ONCONFIG 中缺少该参数，该参数值将视为 0。

用户 gbasedbt 或 DBSA 可以使用 gadmin 实用程序、oninit 实用程序或 ADMIN_MODE_USERS 配置参数动态授予一个或多个特定用户以管理方式连接数据库服务器的能力。

注： 对于 DBSA 组的成员，必须更改对 $GBASEDBTDIR/bin/oninit 的许可权以允许在标准 GBase 8s 安装中运行 public execute permission - root:gbasedbt:6755。

用于更改方式的 ISA 选项

可以使用 Server Administrator (ISA) 更改数据库服务器方式。有关更多信息，请参阅 ISA 联机帮助。

操作	ISA 选项
初始化数据库服务器。	方式 > 联机（初始化磁盘）或静默（初始化磁盘）
从脱机或管理更改为静默。	方式 > 静默
从任何方式更改为管理方式。	方式 > 单用户
从脱机、静默或管理更改为联机。	方式 > 联机
平稳地从联机更改到静默。	方式 > 静默（平稳）
立即从联机更改到静默。	方式 > 静默（立即）
关闭数据库服务器。	方式 > 脱机

用于更改方式的 ON-Monitor 选项 (UNIX)

在 UNIX™ 上，您可以使用 ON-Monitor 更改数据库服务器方式。有关更多信息，请参阅《GBase 8s 管理员参考》中有关 ON-Monitor 的主题。

操作	菜单选项
初始化数据库服务器。	参数 > 初始化
从脱机更改到静默。	方式 > 启动
从脱机或静默更改到联机。	方式 > 联机
平稳地从联机更改到静默。	方式 > 平稳关闭
立即从联机更改到静默。	方式 > 立即关闭
更改到管理方式。	方式 > 单用户
关闭数据库服务器。	方式 > 采用脱机

用于更改方式的命令行选项

表 1 包含所有方式的描述，并显示当服务器处于每种方式时哪些用户可以访问数据库服务器。这些主题包含有关用于更改方式的命令的信息，以及有关方式更改如何影响用户会话的信息。

另见使用 ADMIN_MODE_USERS 配置参数指定管理方式用户。

从脱机更改到静默方式

当数据库服务器从脱机方式更改到静默方式时，数据库服务器将初始化共享内存。仅管理员可以访问数据库服务器以执行不用执行 SQL 和 DDL 语句的维护功能。

操作系统	操作
UNIX™	运行 oninit -s。

从脱机更改到联机方式

将数据库服务器从脱机转为联机方式时，数据库服务器将初始化共享内存并且可被所有用户会话使用。

操作系统	操作
UNIX™	运行 oninit。

从脱机更改到管理方式

将数据库服务器从脱机转移到管理方式时，您将该服务器移至了仅管理员才能用于执行数据库服务器函数和维护函数（包括涉及执行 SQL 和 DDL 语句的函数）的方式中。

操作系统	操作
UNIX™	运行 oninit -j。

用户 gbasedbt 或 DBSA 可使用 oninit -U 命令来指定管理方式用户的列表，如本例中所示：

```
oninit -U mark,ajay,carol
```

在 oninit -U 列表中指定的用户可以在服务器实例活动的时间段内，或在您运行 gadmin -j -U 命令来更改可以连接到服务器的用户列表之前进行连接。运行具有空格的 gadmin -j -U 命令而非名称可除去列表中的所有用户，如本例中所示：

```
oninit -U " "
```

另见使用 ADMIN_MODE_USERS 配置参数指定管理方式用户。

从静默更改到联机方式

当您将数据库服务器从静默方式转到联机方式时，所有会话都将获得访问权。

如果您已经将数据库服务器从联机方式转到静默方式，并且您现在正在将数据库服务器返回到联机方式，那么较早的处理中被中断的任何用户都必须重新选择他们的数据库并且重新声明他们的游标。

操作系统	操作
UNIX™	运行 gadmin -m。

平稳地从联机更改到静默方式

在不中断当前处理的情况下，将数据库服务器从联机方式平稳地转换成静默方式，以限制对数据库服务器的访问。

在您执行了此任务之后，数据库服务器将设置一个标志，该标志将防止新的会话获得对数据库服务器的访问权。 允许当前的会话完成处理。

启动方式更改之后，无法将其取消。在从联机方式转为静默方式的过程中，数据库服务器将被认为处于关闭方式。

操作系统	操作
UNIX™	运行 gadmin -s 或 gadmin -sy。

立即从联机更改到静默方式

将数据库服务器立即从尽快限制对数据库服务器的访问。进行中的工作可能会丢失。

会提示您确认立即关闭。如果确认，那么数据库服务器将向所有连接到共享内存的会话发送断开连接信号。如果会话没有接收到断开连接信号或者不能自动在 10 秒之内答应，那么数据库服务器会终止该会话。

数据库服务器用户将会接收到错误消息 -459（指示数据库服务器已关闭）或错误消息 -457（指示他们的会话已意外终止）。

数据库服务器将清除数据库服务器已终止的所有会话。活动的事务将回滚。

操作系统	操作
UNIX™	运行 gadmin -u 或 gadmin －uy； 使用 **-y** 选项，就无需确认提示。

从静默或联机更改到管理方式

当您将数据库服务器从静默或联机方式移至管理方式时，您就将该服务器移至了一个仅管理员可以使用的方式中。

如果以联机方式开始，数据库服务器自动断开与所有用户的连接，这些用户是使用非用户 **gbasedbt** 的用户标识连接的，并且这些用户将接收到错误消息。如果连接在事务期间终止，那么数据库服务器将回滚回事务。

如果您希望在没有连接其他任何用户的情况下运行 SQL 和 DDL 命令，请更改为管理方式。

操作系统	操作
UNIX™	运行 gadmin -j。

用户 **gbasedbt** 或 DBSA 可以使用 gadmin -j -U 选项授予单个用户对管理方式下数据库服务器的访问权。

例如，运行以下命令可使三个单独的用户连接到数据库服务器并具有数据库服务器的访问权，除非数据库服务器方式更改为脱机、静默或联机方式：

```
gadmin -j -U mark,ajay,carol
```

连接之后，这些个别用户可运行任何 SQL 或 DDL 命令。当该服务器更改为管理方式时，没有在 gadmin -j -U 命令中标识的用户的所有会话将失去与它们的数据库服务器的连接。

在初始运行 gadmin -j -U 命令之后，可以通过运行 gadmin -j -U 并从命令中名称的新列表除去个别用户名，从而除去个别用户，例如，通过运行：

```
gadmin -j -U mark,carol
```

运行具有空格的 gadmin -j -U 命令而非名称可除去列表中的所有用户，如本例中所示：

```
oninit -U " "
```

另见使用 ADMIN_MODE_USERS 配置参数指定管理方式用户。

从管理更改到联机方式

当您将数据库服务器从管理方式移至联机方式时，所有用户都可以访问该数据库服务器。

操作系统	操作
UNIX™	运行 gadmin -m。

从管理更改到静默方式

当您将数据库服务器从管理方式移至静默方式时，您就将该服务器移至了一个仅管理员可以用于执行维护函数（这些函数不包含执行 SQL 和 DDL 语句）的方式中。

操作系统	操作
UNIX™	运行 gadmin -s。

从任何方式立即更改到脱机方式

可以立即使数据库服务器从任何方式更改到脱机方式。

会提示您确认脱机。如果确认，那么数据库服务器将启动检查点请求并且向所有连接到共享内存的会话发送断开连接信号。如果会话没有接收到断开连接信号或者不能自动在 10 秒之内答应，那么数据库服务器会终止此会话。

数据库服务器用户将会接收到错误消息 -459（指示数据库服务器已关闭）或错误消息 -457（指示他们的会话已意外终止）。

当您对数据库服务器采取脱机方式后，请以静默、管理或联机方式重新启动数据库服务器。当您重新启动数据库服务器时，它将执行快速恢复以确保数据在逻辑上是一致的。

数据库服务器将清除已由数据库服务器终止的所有会话。活动的事务将回滚。

操作系统	操作
UNIX™	运行 gadmin -k 或 gadmin -ky。 **-y** 选项将消除确认立即关闭的自动提示。

如果 gadmin 命令无法关闭数据库服务器，可使用 onclean 实用程序强制执行立即关闭。有关 onclean 实用程序的更多信息，请参阅《GBase 8s 管理员参考》。

使用 ADMIN_MODE_USERS 配置参数指定管理方式用户

ADMIN_MODE_USERS 配置参数可使您指定哪个用户可以管理方式连接数据库服务器。不同于 oninit 和 gadmin 命令可使您指定管理方式下的用户直到服务器更改为脱机、静默或联机方式，ADMIN_MODE_USERS 配置参数无限期地保存管理方式下用户的列表。

要创建 onconfig 文件中保存的管理方式用户列表，请指定用户的逗号分隔列表作为 ADMIN_MODE_USERS 配置参数值，例如，mark,ajay,carol。

要在会话期间覆盖 ADMIN_MODE_USERS，请使用 gadmin -wf 命令，如本例中所示：

```
gadmin -wf ADMIN_MODE_USERS=sharon,kalpana
```

ADMIN_MODE_USERS 配置参数的作用是添加到一列允许访问管理方式下的服务器的人员。在 gadmin 命令行中列出的人员会覆盖在 onconfig 文件中列出的人员。

3 磁盘、内存和进程管理

3.1 虚拟处理器和线程

这些主题描述虚拟处理器，说明线程如何在虚拟处理器中运行，并说明数据库服务器如何使用虚拟处理器和线程来提高性能。

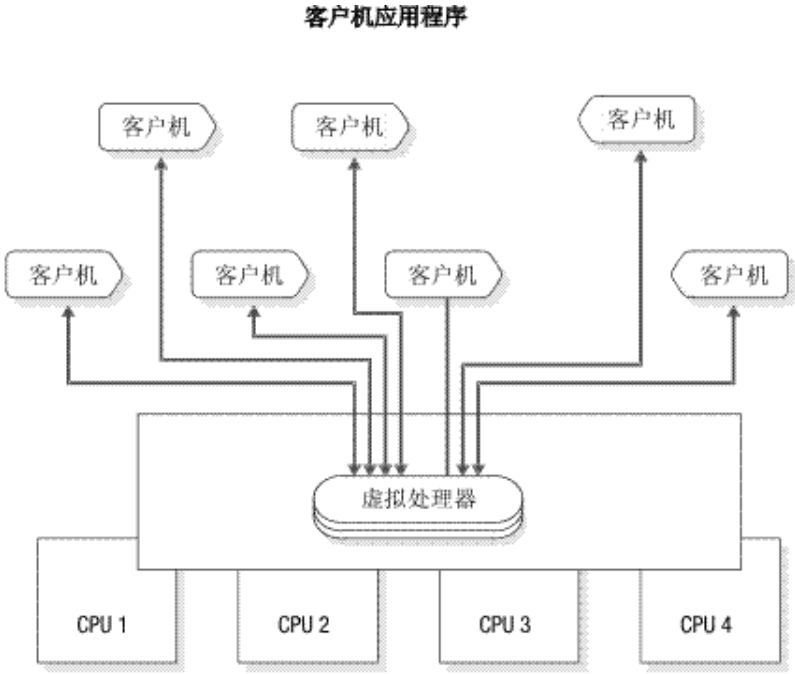
3.1.1 虚拟处理器

数据库服务器进程都被称为*虚拟处理器*，因为它们作用的方式类似于计算机中 CPU 作用的方式。如同 CPU 运行多个操作系统进程为多个用户提供服务一样，数据库服务器虚拟处理器运行多个线程来为多个 SQL 客户机应用程序服务。

虚拟处理器是操作系统调度处理的进程。

下图说明了客户机应用程序与虚拟处理器之间的关系。少量的虚拟处理器可以为大量的客户机应用程序或查询提供服务。

图: 虚拟处理器



线程

线程是虚拟处理器的一个任务，如同虚拟处理器是 CPU 的一个任务一样。虚拟处理器是操作系统调度在 CPU 上执行的任务；数据库服务器线程是虚拟处理器调度进行内部处理的任务。线程有时称为*轻量级进程*，因为它们类似进程，但是对操作系统的要求更少。

数据库服务器虚拟处理器是多线程的，因为它们运行多个并发线程。

线程的性质如下所示。

操作系统	操作
UNIX™	线程是虚拟处理器为处理而内部调度的任务。

重要： 在这些主题中，对*线程*的所有引用都是指数据库服务器创建、调度和删除的线程。

虚拟处理器代表 SQL 客户机应用程序运行线程（*会话*线程），并运行线程以满足内部需求（*内部*线程）。在大多数情况下，对于客户机应用程序的每个连接，数据库服务器都运行一个会话线程。另外，数据库服务器运行内部线程以完成数据库 I/O、日志记录 I/O、页面清除和管理任务。有关数据库服务器为单个客户机运行多个会话线程的情况，请参阅并行处理。

*用户线程*是为来自客户机应用程序的请求提供服务的数据库服务器线程。用户线程包括会话线程（称为 **sqlexec** 线程），这些线程是数据库服务器为了向客户机应用程序提供服务而运行的主线程。

用户线程还包括为来自 gadmin 实用程序的请求提供服务的线程、用于恢复的线程、B 型树扫描程序线程以及页清除程序线程。

要显示活动用户线程，请使用 gstat -u。

虚拟处理器的类型

每一类虚拟处理器都专用于处理某些类型的线程。

下表显示虚拟处理器的类以及它们执行的处理的类型。

您配置的每个类的虚拟处理器数量取决于物理处理器 (CPU)、硬件内存和使用中数据库应用程序的可用性。

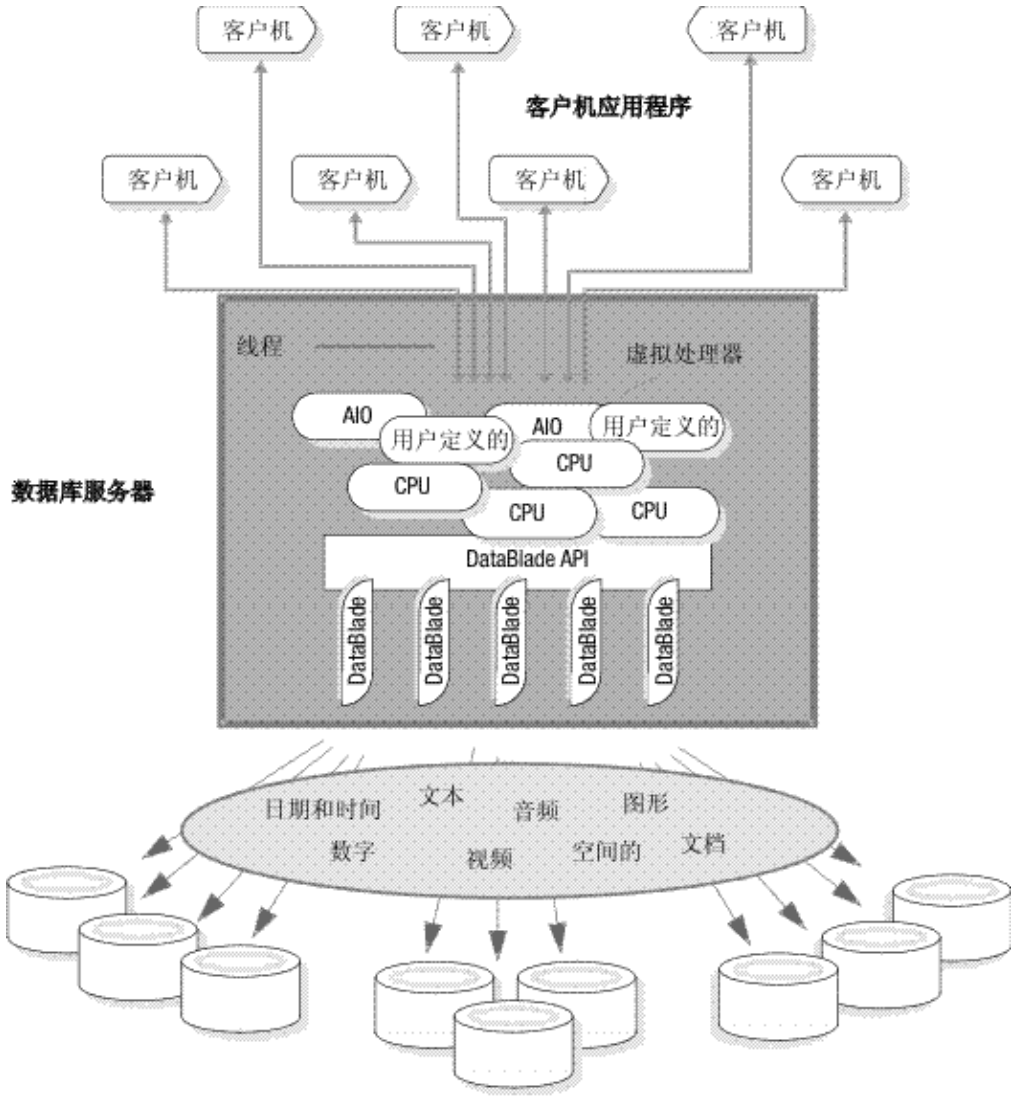
表 1. 虚拟处理器类

虚拟处理器类	分类	用途
ADM	管理	执行管理功能。
ADT	审计	执行审计功能。
AIO	磁盘 I/O	执行非日志记录磁盘 I/O。如果使用了 KAIO，那么 AIO 虚拟处理器将对格式化的磁盘空间执行 I/O。
BTS	Basic Text Search	运行 Basic Text Search 索引操作和查询。
CPU	中央处理	运行所有会话线程以及一些系统线程。 运行可用时的内核异步 I/O (KAIO) 的线程。 可以根据配置运行单个轮询线程。
CSM	通信支持模块	执行通信支持服务操作。
dwavp	数据仓储	在连接到 GBase 8s Warehouse Accelerator 的数据库服务器上，对 GBase 8s Warehouse Accelerator 运行管理功能和过程。
加密	加密	调用加密或解密功能时被数据库服务器使用。
IDSXMLVP	XML 发布	运行 XML 发布函数。
Java™ VP (JVP)	Java UDR	运行 Java UDR。 包含 Java 虚拟机 (JVM)。
LIO	磁盘 I/O	如果逻辑日志文件（内部类）位于格式化的磁盘空间，那么写入这些逻辑日志文件。
MQ	MQ 消息传递	执行 MQ 消息传递事务。
MSC	其他	为需要很大堆栈的系统调用的请求提供服务。
OPT (UNIX™)	Optical	执行对光盘的 I/O。
PIO	磁盘 I/O	如果物理日志文件（内部类）位于格式化的磁盘空间，那么写入该物理日志文件。
SHM	网络	执行共享内存通信。
SOC	网络	使用套接字执行网络通信。
TLI	网络	使用传输层接口 (TLI) 执行网络通信。

虚拟处理器类	分类	用途
WFSVP	Web 要素服务	运行 Web 要素服务例程。
类名	用户定义的	以线程安全方式运行用户定义的例程，以便在例程失败时数据库服务器不受影响。 用 VPCLASS 配置参数指定。 必须指定*类名*。

下图说明了数据库服务器的主要组件以及可扩展性。

图: 数据库服务器



虚拟处理器的优势

与为单台客户机应用程序提供服务的数据库服务器进程相比，数据库服务器虚拟处理器的动态多线程性质将提供下列优势：

- 虚拟处理器可以共享处理。
- 虚拟处理器将保存内存和资源。
- 虚拟处理器可以执行并行处理。

- 当数据库服务器正在运行时，您可以启动其他的虚拟处理器并终止活动的 CPU 虚拟处理器。
- 您可以将虚拟处理器绑定到 CPU。

以下主题将描述这些优势。

共享处理

同一类的虚拟处理器具有相同的代码并且共享对内存中的数据和处理队列的访问。类中任何的虚拟处理器都可以运行任何属于该类的线程。

通常，数据库服务器会尝试将线程保持在同一个虚拟处理器上运行，因为将它移动到不同的虚拟处理器可能需要来自处理器内存的数据在总线上传送。然而，当线程正等待运行时，数据库服务器可以将线程迁移至另一个虚拟处理器，因为平衡处理负载的好处要大于在传送数据时花费的开销。

一类虚拟处理器中的共享处理是自动发生的并且对数据库用户是透明的。

节省内存和资源

相较于一个客户机进程对应一个服务器进程的体系结构，数据库服务器能够用少量的服务器进程处理大量客户机。对于每台客户机，数据库服务器是通过运行线程而不是进程来实现这点的。

因为线程共享分配给虚拟处理器的资源，所以多线程允许对操作系统资源进行更有效的使用。虚拟处理器运行的所有线程对虚拟处理器内存、通信端口和文件具有相同的访问权。虚拟处理器会按线程协调对资源的访问。但是，每个单独的进程都有一组不同的资源，并且在多个进程需要访问相同资源时，操作系统必须对访问进行协调。

通常，虚拟处理器从一个线程切换至另一个线程比操作系统从一个进程切换至另一个进程更快。当操作系统在两个进程间切换时，它必须使一个进程停止在处理器上运行，保存其当前的处理状态（或上下文），然后启动另一个进程。两个进程都必须进入然后退出操作系统内核，并且可能需要替换部分物理内存的内容。但是，线程会共享相同的虚拟内存和文件描述符。在虚拟处理器从一个线程切换到另一个线程时，就相当于从一个执行路径切换到另一个执行路径。虚拟处理器是一个进程，该进程在 CPU 上持续运行而不会中断。有关虚拟处理器如何从一个线程切换到另一个线程的描述，请参阅上下文切换。

并行处理

在以下情况中，CPU 类的虚拟处理器可以为单台客户机以并行方式运行多个会话线程：

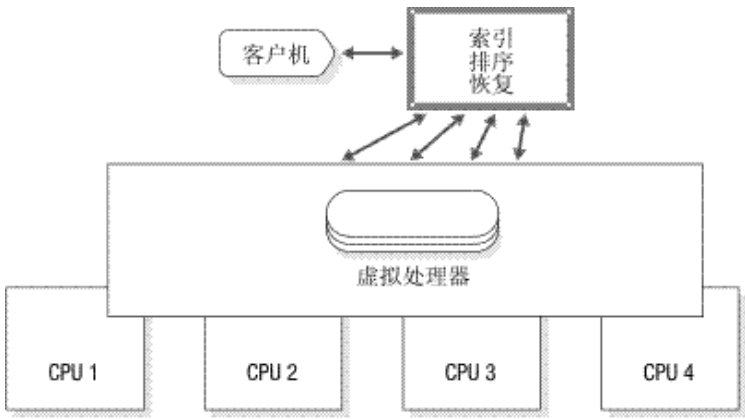
- 索引构建
- 排序
- 恢复
- 扫描
- 连接

- 聚集
- 分组
- 用户定义的例程 (UDR) 执行

有关并行 UDR 执行的更多信息，请参阅《GBase 8s 用户定义的例程与数据类型开发者指南》。

下图说明了并行处理。当客户机启动索引构建、排序或逻辑恢复时，数据库服务器将使用尽可能多的计算机资源创建多个线程以并行处理任务。当一个线程正在等待 I/O，另一个线程可能正在工作。

图: 并行处理



在联机方式下添加和删除虚拟处理器

可以添加虚拟处理器以满足数据库服务器运行时对服务的日益增长的需求。例如，如果类的虚拟处理器变成了计算绑定或 I/O 绑定（意思是可以处理 CPU 工作或 I/O 请求的虚拟处理器的当前数量满足不了处理 CPU 工作或 I/O 请求数日益累积的需求），那么您可以为该类启动额外的虚拟处理器来进一步分布处理装入。

可以在数据库服务器运行时为这些类中的任何一个添加虚拟处理器。有关更多信息，请参阅在联机方式下添加虚拟处理器。

当数据库服务器正在运行时，您可以删除 CPU 的虚拟处理器或用户定义类。有关更多信息，请参阅设置虚拟处理器配置参数。

将虚拟处理器绑定到 CPU

可使用某些多处理器系统将进程绑定到特定 CPU。此功能称为*处理器专用*。

在数据库服务器支持处理器专用的多处理器计算机上，您可以将 CPU 虚拟处理器绑定到该计算机中特定的 CPU。当您将 CPU 虚拟处理器绑定到 CPU 时，虚拟处理器将互斥运行在该 CPU 上。此操作将提高虚拟处理器的性能，因为它减少了操作系统必须在进程间进行的切换量。将 CPU 虚拟处理器绑定到特定的 CPU 还可以使您隔离计算机上特定处理器的数据库工作，以将剩余的处理器释放给其他的工作。仅 CPU 虚拟处理器可以绑定到 CPU。

有关如何将 CPU 虚拟处理器指定到硬件处理器的信息，请参阅处理器亲缘关系。

3.1.2 虚拟处理器如何为线程提供服务

在给定的时间中，虚拟处理器只可运行一个线程。虚拟处理器通过在多个线程之间切换来并发地为这些线程提供服务。虚拟处理器运行线程直到其中止。当一个线程中止时，虚拟处理器将切换到下一个已准备好运行的线程。虚拟处理器将继续此过程，并且最后在原始的线程准备好继续时返回到该线程。有些线程完成了它们的工作，虚拟处理器会启动新线程来处理新的工作。由于虚拟处理器持续在线程之间切换，它可以使得 CPU 持续处理。发生处理时的速度使虚拟处理器看上去就像同时在处理多个任务，而实际上也的确如此。

运行多个并发线程需要调度和同步以防一个线程干扰另一个线程的工作。虚拟处理器可以使用下列结构和方法协调多个线程的并发处理：

- 控制结构
- 上下文切换
- 堆栈
- 队列
- 互斥

这些主题描述虚拟处理器如何使用这些结构和方法。

控制结构

当客户机连接到数据库服务器时，数据库服务器创建*会话*结构（称为*会话控制块*）以保存有关连接和用户的信息。会话在客户机连接到数据库服务器时开始并在连接终止时结束。

接下来，数据库服务器将创建线程结构（该线程结构被称为会话的*线程控制块* (TCB)），并启动主线程（**sqlexec**）来处理客户机请求。当线程*中止*，即当它暂停并允许另一个线程运行时，虚拟处理器将保存有关线程控制块中线程状态的信息。此信息包括系统注册的进程内容、程序计数器（下一个要执行的指令的地址）以及堆栈指针。此信息组成线程的内容。

在大多数情况下，数据库服务器将在每个会话中运行一个主线程。然而，在它执行并行处理的情况下，它将为单台客户机创建多个会话线程以及多个相应的线程控制块。

上下文切换

通过*上下文切换*，虚拟处理器将从运行一个线程切换到运行另一个线程。当固定的时间量（时间片）期满时，数据库服务器不会像操作系统对进程那样预占一个正在运行的线程。而是，线程会在下列点中的一个点上让步：

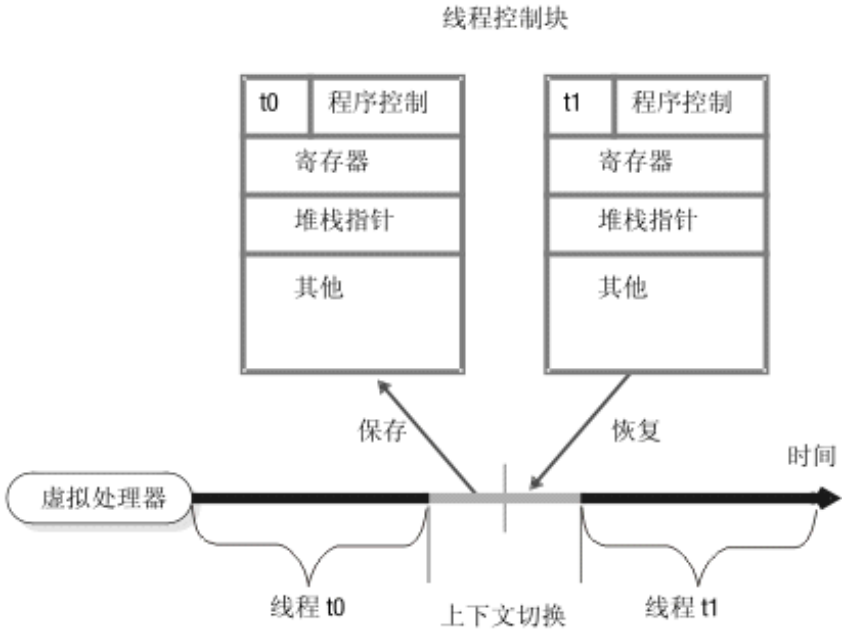
- 代码中预先确定的点
- 当线程在满足某个条件前不再可以执行时

当完成任务所需的处理量会引起其他线程等待一段长度过长的时间时，线程在预先确定的点中止。用于这些长时间运行的任务的代码包括处理中的策略点上对让步函数的调用。当线程执行这些任务之一时，它会在遇到让步函数调用时让步。然后，就绪队列中的其他线程将有机会运行。当下一次轮到原来的线程时，该线程将在对让步函数进行调用后立即重新在点上执行代码。对中止函数的预先确定的调用将允许数据库服务器在对性能最有利的点上中断线程。

当线程无法再继续它的任务时，它也会中止直到出现某种条件。例如，当线程等待磁盘 I/O 完成时、当等待来自客户机的数据时或当等待一个锁定或其他资源时，线程中止。

当一个线程中止时，虚拟处理器在线程控制块中保存其上下文。 然后，虚拟处理器将从一队就绪的线程中选择要运行的新线程、从该新线程的线程控制块中装入该新线程的上下文，然后在程序计数器中的新地址处开始执行。下图说明了虚拟处理器如何完成上下文切换。

图: 上下文切换：虚拟处理器如何从一个线程切换到另一个线程



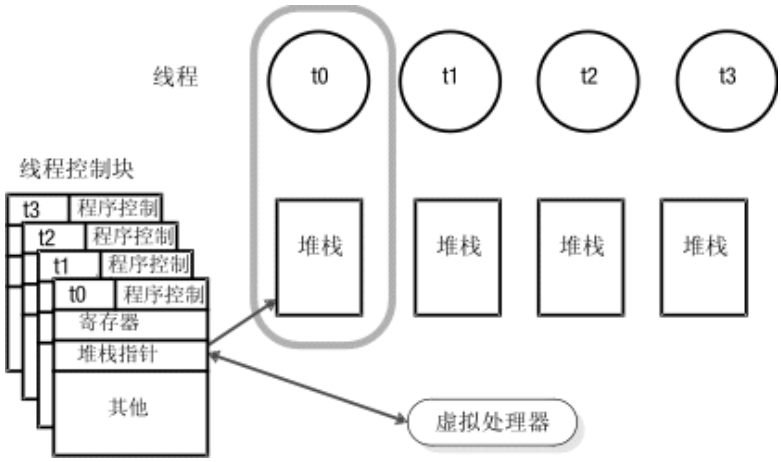
堆栈

数据库服务器将在共享内存的虚拟部分中分配一个区域以便为线程所执行的函数存储非共享数据。此区域称为*堆栈*。有关如何设置堆栈大小的信息，请参阅堆栈。

堆栈使虚拟处理器能够保护线程中非共享数据免遭并发执行相同代码的其他线程的覆盖。例如，如果几台客户机应用程序并发地执行 SELECT 语句，那么每台客户机的会话线程将在代码中执行许多同样的函数。如果线程没有专用堆栈，那么一个线程可能会覆盖属于函数中另一个线程的本地数据。

当虚拟处理器切换到新的线程时，它会为该线程从线程控制块中装入堆栈指针。堆栈指针将存储堆栈的开始地址。 然后，虚拟处理器可以将偏移量指定到在堆栈中存取数据的开始地址。该图说明了虚拟处理器如何使用堆栈来为会话线程分离非共享数据。

图: 虚拟处理器为每个用户分离非共享数据



队列

数据库服务器使用三种类型的队列来调度多个并发运行的线程的处理。

相同类的虚拟处理器将共享队列。有些时候，这将使线程能够在需要时从类中的一个虚拟处理器迁移到另一个虚拟处理器。

就绪队列

就绪队列将在当前的（正在运行的）线程中止时容纳已准备好运行的线程。 当一个线程让步时，虚拟处理器将从就绪队列中选取下一个具有适当优先级的线程。在该队列中，虚拟处理器处理在先进先出 (FIFO) 基础上具有相同优先级的线程。

在多处理器计算机上，如果您注意到线程正在一类的虚拟处理器的就绪队列中堆积（表示工作堆积的速度比虚拟处理器可以处理它的速度块），那么您可以启动该类的附加虚拟处理器以分发处理负载。有关如何监视就绪队列的信息，请参阅监视虚拟处理器。有关如何在数据库服务器处于联机方式时添加虚拟处理器的信息，请参阅在联机方式下添加虚拟处理器。

睡眠队列

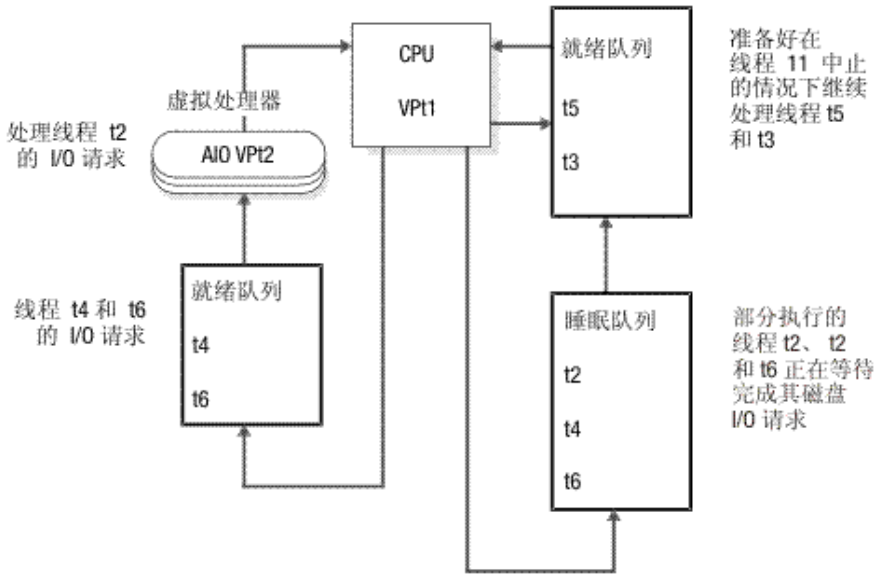
睡眠队列将容纳特定的时间中没有工作要做的线程的上下文。使线程在指定的时间段或永久睡眠。

虚拟处理器的管理类 (ADM) 运行系统计时器和特殊的实用程序线程。此类中的虚拟处理器将自动创建并运行。没有配置参数会影响此类虚拟处理器。

ADM 虚拟处理器唤醒已休眠了指定时间段的线程。运行在 ADM 虚拟处理器中的线程以一秒为时间间隔来检查正在休眠的线程。 如果休眠的线程已休眠了指定时间段，那么 ADM 虚拟处理器将其移动到适当的就绪队列。正在睡眠指定时间段的线程也可以显式地由另一个线程唤醒。

永久睡眠的线程会在其有更多的工作要做时被唤醒。例如，当 CPU 虚拟处理器上运行的线程必须访问磁盘时，该线程会发出 I/O 请求，将其自身置于 CPU 虚拟处理器的休眠队列中，然后停止。当 I/O 线程通知 CPU 虚拟处理器 I/O 已经完成时，CPU 虚拟处理器调度原来的线程通过将该线程从休眠队列移动到就绪队列来继续处理。下图说明了数据库服务器线程如何排队执行数据库 I/O。

图: 数据库服务器线程如何排队执行数据库 I/O



等待队列

等待队列保存了必须等待特定事件才能继续运行的线程。例如，等待队列将协调线程对共享数据的访问。当用户线程尝试获取逻辑日志锁存器但发现锁存器由另一个用户所持有时，遭拒绝的线程会将其自身放在逻辑日志等待队列中。当拥有锁的线程准备好释放锁存器时，它会检查等待的线程，如果线程正在等待，它将唤醒等待队列中的下一个线程。

互斥

互斥（互相排斥），又称为锁存器，是数据库服务器用于同步多个线程对共享资源的访问的一种锁存机制。互斥与信号相似，一些操作系统使用后者来控制多个进程对共享数据的访问。然而，互斥比信号允许更高的并行度。

互斥是与共享资源（如缓冲区）关联的变量。线程必须在可以访问资源前获取资源的互斥。其他线程将不能访问该资源直到所有者将其释放。在互斥可用后，线程通过将其状态设置为使用状态而获得互斥。互斥的同步确保一次只有一个线程写入共享内存的区域。

有关监视互斥的信息，请参阅监视共享内存概要文件和锁存器。

3.1.3 虚拟处理器类

给定类的虚拟处理器只可以运行该类的线程。这些主题将描述每一类虚拟处理器执行的线程类型或处理类型。 还将说明如何确定必须为每一类运行的虚拟处理器的数量。

CPU 虚拟处理器

CPU 虚拟处理器运行所有的会话线程（这些线程处理来自 SQL 客户机应用程序的请求）和某些内部线程。内部线程执行数据库服务器的内部服务。 例如，侦听来自客户机应用程序的连接请求的线程就是一个内部线程。

每个 CPU 虚拟处理器可以具有与之相关联的专用内存高速缓存。每个专用内存高速缓存块由 1 到 32 个内存页组成，每个内存页大小为 4096 个字节。数据库服务器使用专用内存高速缓存来提高对内存块的访问时间。使用 VP_MEMORY_CACHE_KB 配置参数可启用专用内存高速缓存，并指定内存高速缓存的信息。

确定所需 CPU 虚拟处理器数

CPU 虚拟处理器的正确数量是保持繁忙（但并未繁忙到无法跟上进入的请求）的所有 CPU 虚拟处理器的数量。分配的 CPU 虚拟处理器数不得超过计算机中的硬件处理器数。

数据库服务器启动时，除非启用了 SINGLE_CPU_VP 配置参数，否则会自动将 CPU 虚拟处理器的数量增加到数据库服务器计算机上 CPU 处理器数量的一半。但是，可以根据您的系统调整 CPU VP 的数量。

当数据库服务器正在运行时，要评估 CPU 虚拟处理器的性能，请在一组时间周期中以固定的时间间隔重复以下命令：

```
gstat -g glo
```

如果将累积的 *usercpu* 和 *syscpu* 时间加在一起，将接近测试周期的 100% 实际耗用时间，如果您让一个可用的 CPU 运行它，请添加另一个 CPU 虚拟处理器。

可使用 VPCLASS 配置参数指定以下所有信息：

- 类要最初启动的虚拟处理器数
- 类要运行的最大虚拟处理器数
- CPU 类虚拟处理器的处理器专用
- 禁用优先级迟滞（如果适用）

使用 VPCLASS 配置参数可指定此信息。（如果已从非常旧的版本升级到 GBase 8s 的当前版本，请注意，必须使用 VPCLASS 配置参数，而不能使用已终止的 AFF_SPROC、AFFNPROCS、NOAGE、NUMCPUVPS 和 NUMAIOVPS 配置参数。此外，不能将 VPCLASS 参数与已终止的参数结合使用。例如，如果 onconfig 文件包含 NUMCPUVPS 参数，并且还包含 VPCLASS **cpu** 参数，那么您将接收到错误消息。）

在多处理器计算机上运行

如果您要在多处理器计算机上运行多个 CPU 虚拟处理器，请将 onconfig 文件中的 MULTIPROCESSOR 参数设置为 1。在将 MULTIPROCESSOR 设置为 1 时，数据库服务

器将按适合多处理器计算机的方式执行锁定。有关设置多处理器方式的信息，请参阅《GBase 8s 管理员参考》中有关配置参数的章节。

在单处理器计算机上运行

如果您正在单个处理器计算机上运行数据库服务器，请将 MULTIPROCESSOR 配置参数设置为 0。 要运行只具有一个 CPU 虚拟处理器的数据库服务器，请将 SINGLE_CPU_VP 参数设置为 1。

将 MULTIPROCESSOR 设置为 0 可使数据库服务器绕过多处理器计算机上多个处理器所需的锁定。有关 MULTIPROCESSOR 配置参数的信息，请参阅《GBase 8s 管理员参考》。

将 SINGLE_CPU_VP 设置为 1 允许数据库服务器绕过某些互斥调用，这些互斥调用通常在运行多个 CPU 虚拟处理器时建立。有关设置 SINGLE_CPU_VP 参数的信息，请参阅《GBase 8s 管理员参考》。

重要： 将 *VPCLASS num* 设置为 1 并将 SINGLE_CPU_VP 设置为 0 不会减少互斥调用的数量，即使数据库服务器仅启动一个 CPU 虚拟处理器。必须将 SINGLE_CPU_VP 设置为 1 以减少当您运行单个 CPU 虚拟处理器时执行的锁存器的量。

将 SINGLE_CPU_VP 参数设置为 1 会对数据库服务器施加两个重要限制，如下所示：

- 只允许一个 CPU 虚拟处理器。

  当数据库服务器是联机方式时，不能添加 CPU 虚拟处理器。

- 不允许任何用户定义的类。（但是，用户仍然可以定义直接运行在 CPU VP 上的例程。）

有关更多信息，请参阅在联机方式下添加虚拟处理器。

在联机方式下添加和删除 CPU 虚拟处理器

当数据库服务器是联机方式时，您可以添加或删除 CPU 类虚拟处理器。有关如何进行这些操作的指示信息，请参阅在联机方式下添加虚拟处理器和删除 CPU 和用户定义的虚拟处理器。

阻止优先级迟滞

有些操作系统将在累积处理时间时降低长时间运行的进程的优先级。操作系统的这种功能称为优先级迟滞。优先级迟滞可以导致数据库服务器进程的性能随着时间的流逝而降低。然而，在有些情况下，可使用操作系统禁用此功能并且将长时间运行的进程保持在高优先级中运行。

要确定优先级迟滞是否在您的计算机上可用，请检查在本指南『简介』中描述的安装随附的机器说明文件。

如果可通过操作系统禁用优先级迟滞，那么可以在 VPCLASS 配置参数中为优先级条目指定 noage 来禁用该功能。有关更多信息，请参阅《GBase 8s 管理员参考》。

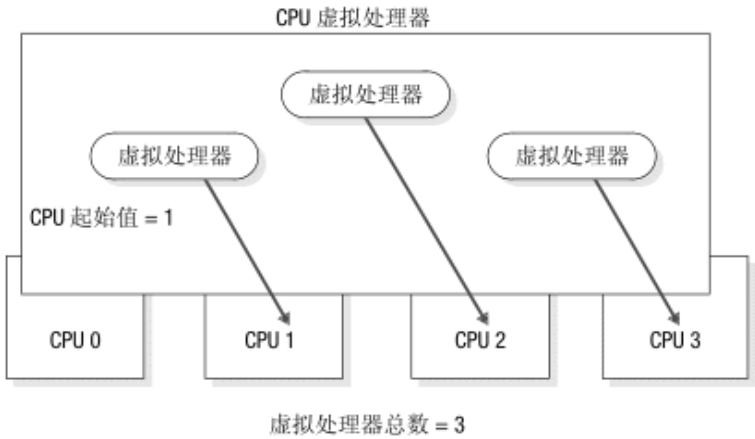
处理器亲缘关系

数据库服务器支持将 CPU 虚拟处理器自动绑定到支持*处理器亲缘关系*的多处理器计算机上的处理器。您的数据库服务器分发手册将包含机器说明文件，该文件包含有关您的数据库服务器是否支持此功能的信息。当您将 CPU 虚拟处理器指定给特定 CPU 时，虚拟处理器只运行在该 CPU 上，但是其他进程也可以运行在该 CPU 上。

使用带有 *aff* 选项的 VPCLASS 配置参数可在支持它的多处理器计算机上实现处理器专用。

下图说明了处理器亲缘关系的概念。

图: 处理器亲缘关系



仅限 UNIX： 要查看 UNIX™ 平台上是否支持处理器亲缘关系，请参阅机器说明文件。

使用 VPCLASS 配置参数设置处理器亲缘关系

要使用 VPCLASS 配置参数设置处理器亲缘关系，可以指定要为其指定虚拟处理器的单个处理器或某一范围的处理器。指定某一范围处理器时，还可以指定范围内的递增值，用于指示将范围中的哪些 CPU 指定给虚拟处理器。例如，可以指定将虚拟处理器分配给范围 0-6 中的每隔一个 CPU，从 CPU 0 开始。

```
VPCLASS CPU,num=4,aff=(0-6/2)
```

虚拟处理器分配给 CPU 0、2、4、6。

如果指定 VPCLASS CPU,num=4,aff=(1-10/3)，将虚拟处理器分配给范围 1-10 中的每隔两个 CPU，从 CPU 1 开始。虚拟处理器分配给 CPU 1、4、7、10。

指定多个值或范围时，这些值和范围不必是递增的，也无需按照任何特定顺序指定。例如，可以指定 aff=(8,12,7-9,0-6/2)。

数据库服务器将 CPU 虚拟处理器分配给循环模式中的 CPU，从您在 aff 选项中指定的第一个处理器编号开始。如果指定的 CPU 虚拟处理器数多于实际的 CPU 数量，那么数据库服务器继续从首个 CPU 开始分配 CPU 虚拟处理器。例如，假设您指定了以下 VPCLASS 设置：

```
VPCLASS cpu,num=8,aff=(4-7)
```

数据库服务器会进行以下分配：

- CPU 虚拟处理器编号 0 到 CPU 4
- CPU 虚拟处理器编号 1 到 CPU 5
- CPU 虚拟处理器编号 2 到 CPU 6
- CPU 虚拟处理器编号 3 到 CPU 7
- CPU 虚拟处理器编号 4 到 CPU 4
- CPU 虚拟处理器编号 5 到 CPU 5
- CPU 虚拟处理器编号 6 到 CPU 6
- CPU 虚拟处理器编号 7 到 CPU 7

有关更多信息，请参阅《GBase 8s 管理员参考》中的 VPCLASS 配置参数。

用户定义的虚拟处理器类

可以定义特殊的虚拟处理器类以运行用户定义的例程。通常可编写用户定义的例程以支持用户定义的数据类型。 如果您不想要用户定义的例程运行在 CPU 类（这是缺省值）中，那么您可以将它指定给虚拟处理器 (VP) 的用户定义类。用户定义的虚拟处理器类还称为*扩展虚拟处理器*。

这些主题提供了以下有关用户定义的虚拟处理器的信息：

- 何时在用户定义的 VP 而不是 CPU VP 中运行 C 语言 UDR
- 如何将 C 语言 UDR 指定给一个特定的用户定义 VP 类
- 当数据库服务器处于联机方式时，如何添加和删除用户定义的 VP

确定所需的用户定义的虚拟处理器数

可以指定操作系统允许的任意数量的用户定义虚拟处理器。如果运行许多 UDR 或带有 UDR 的并行 PDQ 查询，那么必须配置多个用户定义的虚拟处理器。

用户定义的虚拟处理器

用户定义的虚拟处理器类可保护数据库服务器免遭*行为不良的*用户定义例程的破坏。行为不良的用户定义的例程至少有以下特征之一：

- 不会为其他线程让出控制权
- 进行分块操作系统调用
- 修改全局 VP 状态

行为良好的 C 语言 UDR 不具有这些特征。 仅在 CPU VP 中运行行为良好的 C 语言 UDR。

警告： 在 CPU VP 中执行行为不良的例程会导致对数据库服务器操作的严重干扰，并且可能导致其失败或行为反常。此外，例程本身可能不会产生正确的结果。

要确保安全执行，将任何行为不良的用户定义的例程指定给用户定义的虚拟处理器类。 用户定义的 VP 除去 CPU VP 类上的以下编程限制：

- 需要定期让出处理器
- 需要终止阻塞 I/O 调用

在用户定义的虚拟处理器类中运行的函数无需让出处理器，并且它们可能发出直接的文件系统调用，用于阻止虚拟处理器的进一步处理，直到 I/O 完成。

用户查询的正常处理不受 C 语言 UDR 不良行为的影响，因为这些 UDR 不在 CPU 虚拟处理器中执行。

指定用户定义的虚拟处理器

带有 *vpclass* 选项的 VPCLASS 参数定义用户定义的 VP 类。还可以指定非让步的用户定义的虚拟处理器。有关更多信息，请参阅指定虚拟处理器类和《GBase 8s 管理员参考》中有关配置参数的主题。

将 UDR 指定给用户定义的虚拟处理器类

SQL CREATE FUNCTION 语句注册用户定义的例程。例如，以下 CREATE FUNCTION 语句注册用户定义的例程 GreaterThanEqual()，并指定对于该例程的调用由名为 UDR 的用户定义的 VP 类执行：

```
CREATE FUNCTION GreaterThanEqual(ScottishName, ScottishName)
    RETURNS boolean
    WITH (CLASS = UDR    )
    EXTERNAL NAME    '/usr/lib/objects/udrs.so'
    LANGUAGE C
```

要执行该函数，onconfig 文件必须包含定义 UDR 类的 VPCLASS 参数。否则，对 GreaterThanEqual 函数的调用将失败。

提示： CLASS 例程修饰符可为 VP 类指定任何名称。注册 UDR 时，无需有该类名。然而，尝试运行 UDR 且其指定了用于执行的用户定义 VP 类时，此类必须存在并且必须为其指定虚拟处理器。

要配置 UDR 类，请在 onconfig 文件中包含如下类似行。此行将配置带有两个虚拟处理器并且没有优先级迟滞的 UDR 类。

```
VPCLASS        UDR    ,num=2,noage
```

前面的行将 UDR VP 类定义为生成的 VP 类；即该 VP 类允许 C 语言 UDR 生成必须访问 UDR VP 类的其他线程。有关如何使用 VPCLASS 配置参数的更多信息，请参阅《GBase 8s 管理员参考》。

有关 CREATE FUNCTION 语句的更多信息，请参阅《GBase 8s SQL 指南：语法》。

在联机方式下添加和删除用户定义的虚拟处理器

可以在数据库服务器处于联机状态时添加或删除用户定义的类中的虚拟处理器。有关如何进行这些操作的指示信息，请参阅在联机方式下添加虚拟处理器和删除 CPU 和用户定义的虚拟处理器。

Java 虚拟处理器

Java™ UDR 和 Java 应用程序在专门的虚拟处理器（称为 *Java 虚拟处理器* (JVP)）上运行。JVP 在其代码中嵌入 Java 虚拟机 (JVM)。JVP 与 CPU VP 具有相同的功能，CPU VP 可以处理所有的 SQL 查询。

可以指定操作系统允许的任意数量的 JVP。如果运行许多 Java 或带有 Java UDR 的并行 PDQ 查询，那么必须配置更多 JVP。

使用带有 jvp 关键字的 VPCLASS 配置参数可配置 JVP。有关更多信息，请参阅《GBase 8s 管理员参考》中的配置参数章节。

磁盘 I/O 虚拟处理器

以下虚拟处理器的类执行磁盘 I/O：

- PIO（物理日志 I/O）
- LIO（逻辑日志 I/O）
- AIO（异步 I/O）
- CPU（内核异步 I/O）

*除非*物理日志文件和逻辑日志文件位于原始磁盘空间中，并且数据库服务器已实现 KAIO，否则 PIO 类将执行所有到物理日志文件的 I/O，而 LIO 类执行所有到逻辑日志文件的 I/O。

在不支持 KAIO 的操作系统上，数据库服务器使用虚拟处理器的 AIO 类来执行与物理或逻辑日志记录不相关的数据库 I/O。

当 KAIO 可用于平台时，数据库服务器使用 CPU 类来执行 KAIO。如果数据库服务器实现 KAIO，那么 KAIO 线程对原始的磁盘空间执行所有的 I/O，包括物理日志和逻辑日志的 I/O。

仅限 UNIX： 要了解您的 UNIX™ 平台是否支持 KAIO，请参阅机器说明文件。

有关非日志记录 I/O 的更多信息，请参阅异步的 I/O。

I/O 优先级

一般来说，数据库服务器通过将不同类型的 I/O 指定给虚拟处理器的不同类，并将优先级指定给非日志记录 I/O 队列来划分磁盘 I/O 的优先级。例如，划分优先级可确保高优先级日志 I/O 绝不会排列在对临时文件进行写操作的后面，该写操作具有低优先级。数据库服务器会对于其执行的不同类型的磁盘 I/O 划分优先级，如下表所示。

表 1. 数据库服务器为磁盘 I/O 划分优先级的方法

优先级	I/O 类型	VP 类

优先级	I/O 类型	VP 类
第 1 级	逻辑日志 I/O	CPU 或 LIO
第 2 级	物理日志 I/O	CPU 或 PIO
第 3 级	数据库 I/O	CPU 或 AIO
第 3 级	页清除 I/O	CPU 或 AIO
第 3 级	预读取 I/O	CPU 或 AIO

逻辑日志 I/O

虚拟处理器的 LIO 类对以下情况中的逻辑日志文件执行 I/O：

- 没有实现 KAIO。
- 逻辑日志文件位于熟磁盘空间。

仅当实现了 KAIO 并且逻辑日志文件在原始的磁盘空间中时，数据库服务器才会使用 CPU 虚拟处理器中的 KAIO 线程来对逻辑日志执行 I/O。

逻辑日志文件将存储使数据库服务器能够回滚事务并从系统故障中恢复的数据。对逻辑日志文件的 I/O 是数据库服务器执行的最高优先级的磁盘 I/O。

如果逻辑日志文件存在于非镜像的数据库空间中，那么数据库服务器只运行一个 LIO 虚拟处理器。如果逻辑日志文件存在于镜像的数据库空间中，那么数据库服务器运行两个 LIO 虚拟处理器。此虚拟处理器类没有任何与其关联的参数。

物理日志 I/O

虚拟处理器的 PIO 类对以下情况中的物理日志文件执行 I/O：

- 没有实现 KAIO。
- 物理日志文件存储在已缓冲的文件块中。

仅当实现了 KAIO 并且物理日志文件存在于原始的磁盘空间中时，数据库服务器才会使用 CPU 虚拟处理器中的 KAIO 线程来对物理日志执行 I/O。物理日志文件将存储自上一个检查点以来已更改的数据库空间页的前映像。（有关检查点的更多信息，请参阅检查点。）在恢复开始时，数据库服务器会在处理来自逻辑日志的事务前，使用物理日志文件将前映像复原到自上一个检查点以来已更改的数据库空间页。对物理日志文件的 I/O 是继对逻辑日志文件的 I/O 之后第二高的优先级 I/O。

如果物理日志文件存在于非镜像的数据库空间中，那么数据库服务器只运行一个 PIO 虚拟处理器。 如果物理日志文件存在于镜像的数据库空间中，那么数据库服务器运行两个 PIO 虚拟处理器。此虚拟处理器类没有任何与其关联的参数。

异步的 I/O

数据库服务器异步执行数据库 I/O，这意味着 I/O 将独立于请求 I/O 的线程进行排列和执行。异步地执行 I/O 允许建立请求的线程当 I/O 正在执行时继续工作。

数据库服务器使用以下某个设施异步执行所有的数据库 I/O：

- AIO 虚拟处理器
- 平台（该平台支持 KAIO）上的 KAIO

数据库 I/O 包含 SQL 语句的 I/O、预读 I/O、页清除 I/O 以及检查点 I/O。

内核异步 I/O

当以下条件存在时，数据库服务器使用 KAIO：

- 计算机和操作系统对其支持。
- 已实现性能提高。
- I/O 是对于原始的磁盘空间的。

数据库服务器通过在 COU 虚拟处理器上运行 KAIO 线程来实现 KAIO。KAIO 线程通过建立对操作系统（该操作系统可执行独立于虚拟处理器的 I/O）的系统调用来执行 I/O。KAIO 线程能够对于磁盘 I/O 产生比 AIO 虚拟处理器产生的更好的性能，因为该线程不需要在 CPU 和 AIO 虚拟处理器之间切换。

仅限 UNIX： 当 GBase 8s 具有支持此功能的平台的端口时，GBase 8s 将实现 KAIO。数据库服务器管理员不用配置 KAIO。要查看 KAIO 是否在您的平台上受支持，请参阅机器说明文件。

仅限 Linux： 内核异步 I/O (KAIO) 在缺省情况下启用。 您可以通过在启动服务器的进程环境中指定 KAIOOFF=1 来禁用内核异步 I/O。

在 Linux™ 上，并行 KAIO 请求具有最大数量的系统范围限制。/proc/sys/fs/aio-max-nr 文件包含该值。Linux™ 系统管理员可以增加该值，例如，通过使用该命令：

```
# echo new_value > /proc/sys/fs/aio-max-nr
```

所有操作系统进程的已分配请求的当前数量在 /proc/sys/fs/aio-nr 文件中可见。

在缺省情况下，Datebse Server 分配请求的最大数量的一半，并将它们同样分配给已配置 CPU 虚拟处理器的数量。 您可以使用环境变量 KAIOON 来控制分配给每个 CPU 虚拟处理器的请求数量。在启动 GBase 8s 之前，通过将 KAIOON 设置为必需值来执行此操作。

KAIOON 的最小值是 100。如果 Linux 即将耗尽 KAIO 资源，例如当动态地添加许多 CPU 虚拟处理器时，online.log 文件中将打印警告。如果发生了此情况，那么 Linux 系统管理员必须按照上述方式添加 KAIO 资源。

AIO 虚拟处理器

如果平台不支持 KAIO 或如果 I/O 是缓冲区文件块，那么数据库服务器将通过虚拟处理器的 AIO 类执行数据库 I/O。所有的 AIO 虚拟处理器同等服务其类中的所有 I/O 请求。

数据库服务器将根据块的文件名为每个磁盘块指定一个队列（有时称为 gfd 队列）。数据库服务器根据最小化磁头移动算法预订队列中的 I/O 请求。AIO 虚拟处理器为工作在循环法方式下暂挂的队列提供服务。

所有其他的非阻塞 I/O 排列在 AIO 队列中。

使用带有 aio 关键字的 VPCLASS 参数可指定数据库服务器初次启动的 AIO 虚拟处理器的数量。有关 VPCLASS 的信息，请参阅《GBase 8s 管理员参考》中有关配置参数的章节。

当数据库服务器是联机方式时，您可以启动其他的 AIO 虚拟处理器。有关更多信息，请参阅在联机方式下添加虚拟处理器。

当数据库服务器处于联机方式时，您不能删除 AIO 虚拟处理器。

自动增加和降低 AIO 虚拟处理器的数量

当服务器检测到 AIO VP 的处理速度跟不上 I/O 工作负载时，AUTO_AIOVPS 配置参数启用或禁用数据库服务器自动增加 AIO VP 和 flusher 线程的数目。如果您想要手动调整该数值，请禁用此参数。有关设置减少此参数的详细信息，请参阅《GBase 8s 管理员参考》。

所需的 AIO 虚拟处理器数

分配 AIO 虚拟处理器的目的是分配足够的虚拟处理器从而可以保持 I/O 请求队列的长度比较短；即，队列中具有的 I/O 请求要尽可能少。当 gfd 队列总是很短时，它指示对于磁盘设备的 I/O 正在以与产生请求一样快的速度进行处理。

gstat-g ioq 命令显示了有关 I/O 队列的长度和其他统计信息。可以使用此命令为 AIO 虚拟处理器监视 gfd 队列的长度。

一个 AIO 虚拟处理器可能已足够：

- 如果数据库服务器在您的平台上实现内核异步 I/O (KAIO) 并且所有的数据库空间都是由原始的磁盘空间组成。
- 如果您的文件系统支持用于数据库空间块的页大小的直接 I/O 并且您使用直接 I/O 对每个活动的数据库空间分配两个由缓冲区文件块组成的 AIO 虚拟处理器。
- 如果数据库服务器实现 KAIO，但是您正在使用块的某些缓冲区文件
- 如果系统不支持块的 KAIO。

如果 KAIO *不*在您的平台上实现，请为数据库服务器经常访问的每个磁盘分配两个 AIO 虚拟处理器。

如果您使用熟文件，并且如果您使用 DIRECT_IO 配置参数启用直接 I/O，那么您可能可以减少 AIO 虚拟处理器的数量。

如果数据库服务器实现了 KAIO 并且使用 DIRECT_IO 配置参数启用了直接 I/O，那么 GBase 8s 将尝试使用 KAIO，这样可能就不需要多个 AIO 虚拟处理器。 但是，即使启用了直接 I/O，如果文件系统不支持直接 I/O 或 KAIO，那么仍必须为组成已缓冲文件块或不使用 KAIO 的每个活动的数据库空间分配两个 AIO 虚拟处理器。

临时数据库空间不使用直接 I/O。如果您拥有临时数据库空间，那么可能需要多个 AIO 虚拟处理器。

分配足够的 AIO 虚拟处理器以满足 I/O 请求的最大值。通常来说，分配过多的 AIO 虚拟处理器不会产生不利影响。

当服务器检测到 AIO 虚拟处理器无法满足 I/O 工作负载时，AUTO_AIOVPS 配置参数可使数据库服务器自动增加 AIO 虚拟处理器和 page-cleaner 线程的数量。

网络虚拟处理器

如客户机/服务器通信中所述，客户机可以下列方式连接到数据库服务器：

- 通过网络连接
- 通过管道
- 通过共享内存

网络连接可以由远程计算机上的客户机建立，或由本地计算机上的客户机通过模拟来自远程计算机的连接（称为本地回送连接）建立。

指定网络连接

通常来说，DBSERVERNAME 和 DBSERVERALIASES 参数定义在 sqlhosts 文件或注册表中具有对应条目的 dbservername。sqlhosts 中的每个 dbservername 都有指定接口/协议组合的 nettype 条目。数据库服务器为每个唯一的 nettype 条目运行一个或多个轮询线程。

NETTYPE 配置参数为接口/协议组合提供可选配置信息。可以将其用于为接口/协议组合指定多个轮询线程，也可用于指定运行轮询线程的虚拟处理器类（CPU 或 NET）。

有关 NETTYPE 配置参数的完整描述，请参阅《GBase 8s 管理员参考》。

在 CPU 或网络虚拟处理器上运行轮询线程

轮询线程可以在 CPU 虚拟处理器或网络虚拟处理器上运行。通常来说，特别在单个处理器计算机上，轮询线程在 CPU 虚拟处理器上的运行更加高效。然而，在具有大量远程客户机的多处理器计算机上，这种情况可能并不成立。

NETTYPE 参数具有名为 *vp class* 的可选条目，可用于指定 CPU 或 NET 以分别表示 CPU 或网络虚拟处理器类。

如果没有为与 DBSERVERNAME 变量关联的接口/协议组合（轮询线程）指定虚拟处理器类，该类将缺省为 CPU。数据库服务器假设与 DBSERVERNAME 关联的接口/协议组合是效率最高的主接口/协议组合。

对于其他的接口/协议组合，如果没有指定 vp class，那么缺省值将是 NET。

当数据库服务器处于联机方式时，您无法删除正在运行轮询或侦听线程的 CPU 虚拟处理器。

重要： 必须仔细区分用于网络连接的轮询线程和用于共享内存连接的轮询线程，后者在每个 CPU 虚拟处理器上只能运行一个。TCP 连接必须仅处于网络虚拟处理器中，并且您必须只有维持响应所需的最少的 TCP 连接数。共享内存连接必须仅处于 CPU 虚拟处理器中，并在每个 CPU 虚拟处理器中运行。

指定联网虚拟处理器数

每个轮询线程需要单独的虚拟处理器，因此当您为接口/协议组合指定轮询线程数量时，您将间接地指定连网虚拟处理器的数量并指定这些处理器将由 NET 类运行。如果您为 vp class 指定 CPU 时，您必须分配一个足够大数量的 CPU 虚拟处理器以运行轮询线程。如果数据库服务器没有使 CPU 虚拟处理器运行 CPU 轮询线程，它将启动指定类的网络虚拟处理器运行轮询线程。

对于大多数系统，每个接口/协议组合有一个轮询线程以及一个虚拟处理器就足够了。对于具有 200 或更多网络用户的系统，运行额外的网络虚拟处理器可能提高吞吐量。在这种情况下，必须进行实验以便确定每个接口/协议组合的最佳虚拟处理器数。

为客户机/服务器连接指定侦听和轮询线程

启动数据库服务器时，oninit 进程将为使用 onconfig 文件中的 DBSERVERNAME 和 DBSERVERALIASES 参数指定的每个 dbservername 启动内部线程，这些线程称为*侦听线程*。要为这些数据库服务器名条目中的每一个指定侦听端口，请在 **sqlhosts** 中为其指定 **hostname** 和 service name 条目的唯一组合。例如，下表中显示的 sqlhosts 文件条目或注册表项将使数据库服务器 **soc_ol1** 为主机或网络地址 **myhost** 上的 **port1** 启动侦听线程。

表 1. 每个侦听端口的侦听线程

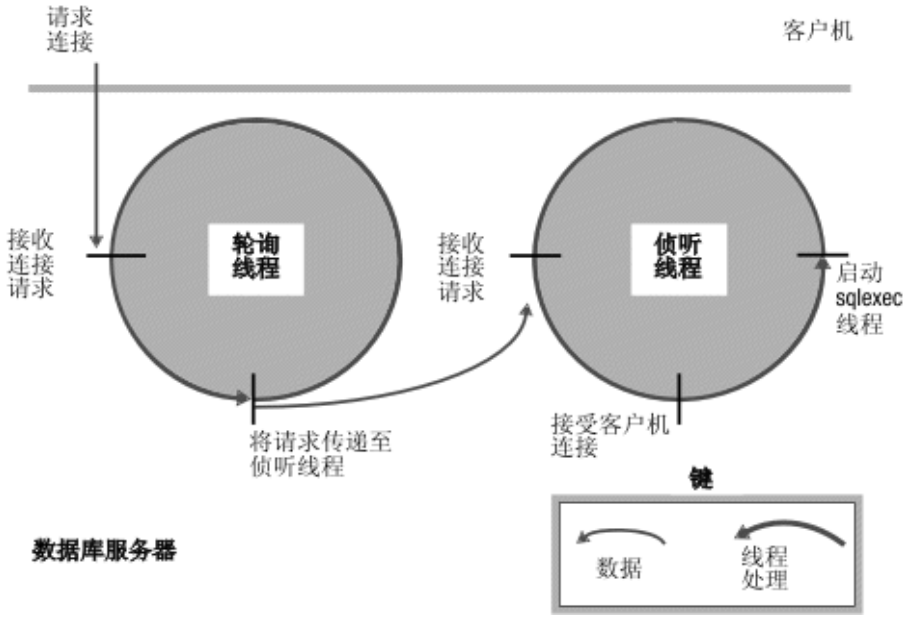
dbservername	**nettype**	**hostname**	**服务名**
soc_ol1	onsoctcp	myhost	port1

侦听线程打开端口并请求指定接口/协议组合的轮询线程之一监视客户机请求的端口。轮询线程为正在使用的连接运行于 CPU 虚拟处理器或网络虚拟处理器中。有关轮询线程数的信息，请参阅指定联网虚拟处理器数。

有关如何指定接口/协议组合的轮询线程是在 CPU 虚拟处理器中运行还是在网络虚拟处理器中运行的信息，请参阅在 CPU 或网络虚拟处理器上运行轮询线程以及《GBase 8s 管理员参考》中的 NETTYPE 配置参数。

当轮询线程从客户机接收到连接请求，它会将请求传递给该端口的侦听线程。侦听线程将认证用户，建立到数据库服务器的连接，然后启动 **sqlexec** 线程，该线程是为客户机执行主处理的会话线程。下图说明了侦听线程和轮询线程在建立与客户机应用程序的连接时的角色。

图: 轮询线程和侦听线程在与客户机连接时的角色

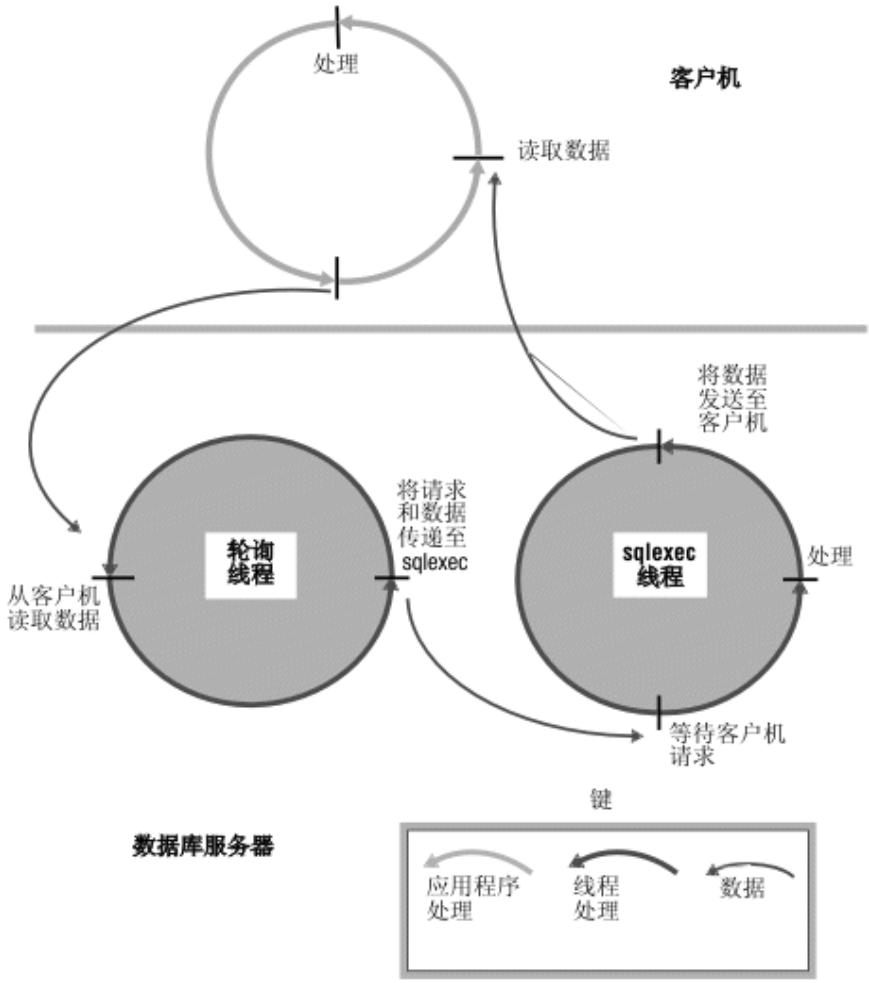


轮询线程等待来自客户机的请求，并将这些请求放在共享内存以便由**sqlexec** 线程处理。 对于网络连接，轮询线程将消息放在共享内存全局池中的队列中。然后，轮询线程将唤醒客户机的 **sqlexec** 线程以处理该请求。只要可能，**sqlexec** 线程都会直接写回客户机而无需轮询线程的帮助。通常，轮询线程从客户机读取数据，然后 **sqlexec** 线程将数据发送到客户机。

仅限 UNIX： 对于共享内存连接，轮询线程将该消息放在共享内存的通信部分中。

下图说明了轮询线程和 **sqlexec** 线程在与客户机应用程序通信时执行的基本任务。

图: 轮询线程和 sqlexec 线程在与客户机应用程序通信时的角色



快速轮询

如果操作系统平台支持快速轮询，那么使用 FASTPOLL 配置参数可启用或禁用网络的快速轮询。如果有大量连接，快速轮询会很有用。例如，如果数据库服务器具有 300 多个并行连接，那么可以启用 FASTPOLL 配置参数以获取更好的性能。可通过将 FASTPOLL 配置参数设置为 1 来启用快速轮询。

如果您的操作系统不支持快速轮询，GBase 8s 将忽略 FASTPOLL 配置参数。

多个侦听线程

可以通过使用多个侦听线程来提高连接请求的服务。

如果数据库服务器无法为带有单个端口和相应侦听线程的给定接口/协议组合提供令人满意的连接请求服务，那么可以通过以下方法改进对连接请求的服务：

- 为其他端口添加侦听线程。
- 向同一个端口添加侦听线程（前提是有 **onimcsoc** 或 **onsoctcp** 协议）
- 再添加一块网络接口卡。
- 通过使用 SQL 管理 API 或 gadmin -P 命令，动态启动、停止或重新启动 SOCTCP 或 TLITCP 网络协议的侦听线程。

如果 onsoctcp 协议的一个端口有多个侦听线程，并且 CPU VP 连接正忙，那么数据库服务器可以接受新连接。

添加侦听线程

启动数据库服务器时，**oninit** 进程将为具有使用 DBSERVERNAME 和 DBSERVERALIASES 配置参数指定的服务器名称和服务器别名的服务器启动侦听线程。可以为其他端口添加侦听线程。

还可以为 **onimcsoc** 或 **onsoctcp** 的一个服务（端口）设置多个侦听线程。

要为附加端口添加侦听线程，您必须首先使用 DBSERVERALIASES 参数来指定每个端口的 dbservername。例如，下图中 DBSERVERALIASES 参数为标识为 **soc_ol1** 的数据库服务器实例另外定义了两个 dbservername：**soc_ol2** 和 **soc_ol3**。

```
DBSERVERNAME      soc_ol1

DBSERVERALIASES   soc_ol2,soc_ol3
```

为数据库服务器定义了其他 dbservername 之后，必须在 sqlhosts 文件或注册表中为每个 dbservername 指定接口/协议组合与端口。每个端口将由 **hostname** 和 **servicename** 条目的唯一组合标识。 例如，下表中显示的 sqlhosts 条目会使数据库服务器为 **onsoctcp** 接口/协议组合启动三个侦听线程，每个定义的端口一个线程。

表 1. 要侦听单个接口/协议组合的多个端口的 sqlhosts 条目

dbservername	**nettype**	**hostname**	**服务名**
soc_ol1	onsoctcp	myhost	port1
soc_ol2	onsoctcp	myhost	port2
soc_ol3	onsoctcp	myhost	port3

如果包含了接口/协议组合的 NETTYPE 参数，它将应用于接口/协议组合的所有连接。 换句话说，如果上表中存在 **onsoctcp** 的 NETTYPE 参数，那么该参数将应用于所有显示的连接。在本例中，数据库服务器为 **onsoctcp** 接口/协议组合运行一个*轮询*线程，直到指定了更多的 NETTYPE 参数。有关 sqlhosts 文件中的条目或注册表中的项的更多信息，请参阅连接文件。

为 onimcsoc 或 onsoctcp 协议的一个端口设置多个侦听线程

要为 **onimcsoc** 或 **onsoctcp** 协议的一个服务(端口)设置多个侦听线程,请执行以下操作：

- DBSERVERNAME <*name*>-<*n*>
- DBSERVERALIASES <*name1*>-<*n*>,<*name2*>

例如：

- 要为 DBSERVERNAME 为 `ifx` 的服务器启用两个侦听线程，请指定：

  ```
  DBSERVERNAME ifx-2
  ```

- 要为 DBSERVERALIASES 为 `ifx_a` 和 `ifx_b` 的服务器启用两个侦听线程,请指定：

```
DBSERVERALIASES ifx_a-2,ifx_b-2
```

添加网络接口卡

可以添加网络接口卡来提高性能或将数据库服务器连接到多个网络。

如果主机的网络接口卡无法提供令人满意的连接请求服务，您可能希望提高性能。

要支持多块网络接口卡，您必须在 **sqlhosts** 中为每块卡指定一个唯一的主机名（网络地址）。

例如，通过使用添加侦听线程中显示的同一个 dbservernames，下表中显示的 sqlhosts 文件条目或注册表项将使数据库服务器为相同接口/协议组合启动三个侦听线程（如添加侦听线程中的条目所执行的操作）。但是，在此情况下，两个线程要侦听一个接口卡 (**myhost1**) 上的端口，而第三个线程要侦听第二个接口卡 (**myhost2**) 上的端口。

表 1. 为 onsoctcp 接口/协议组合的两块网络接口卡提供支持的 sqlhosts 条目示例

dbservername	**nettype**	**hostname**	**服务名**
soc_ol1	onsoctcp	myhost1	port1
soc_ol2	onsoctcp	myhost1	port2
soc_ol3	onsoctcp	myhost2	port1

动态启动、停止或重新启动侦听线程

可以为 SOCTCP 或 TLITCP 网络协议动态启动、停止或停止并启动侦听线程，而不必中断现有连接。 例如，您可能希望停止不响应的侦听线程，然后启动新侦听线程，而同时不希望关闭其他正在正常执行的服务器。

侦听线程必须在服务器的 sqlhosts 文件中定义。如有必要，可在启动、停止或重新启动侦听线程之前，修改 sqlhosts 条目。

要动态启动、停止或重新启动侦听线程，请执行以下操作：

1. 运行以下某个 gadmin -P 命令：
   - gadmin -P start *server_name*
   - gadmin -P stop *server_name*
   - gadmin -P restart *server_name*
2. 此外，如果已直接或远程连接到 **sysadmin** 数据库，那么可运行以下某个命令：
   - 带 start listen 自变量的 admin() 或 task() 命令，格式如下

     ```
     EXECUTE FUNCTION task("start listen", "server_name");
     ```
   - 带 stop listen 自变量的 admin() 或 task() 命令，格式如下

     ```
     EXECUTE FUNCTION task("stop listen" ,"server_name");
     ```
   - 带 restart listen 自变量的 admin() 或 task() 命令，格式如下

     ```
     EXECUTE FUNCTION task("restart listen", "server_name");
     ```

例如，以下任一命令均为名为 **ifx_serv2** 的服务器启动新侦听线程：

```
gadmin -P start ifx_serv2

EXECUTE FUNCTION task("start listen", "ifx_serv2");
```

通信支持模块虚拟处理器

虚拟处理器的通信支持模块 (CSM) 类执行通信支持服务和通信支持模块函数。

除非将通信支持模块设置为 GSSCSM 来支持单点登录，否则数据库服务器会启动与 CPU 虚拟处理器启动数量相同的 CSM 虚拟处理器。如果通信支持模块为 GSSCSM，数据库服务器将仅启动一个 CSM 虚拟处理器。

有关通信支持服务的更多信息，请参阅客户机/服务器通信。

加密虚拟处理器

如果没有在 onconfig 配置文件中定义 VPCLASS 参数的 encrypt 选项，那么在首次调用为列级别加密定义的任何加密或解密功能时，数据库服务器将启动一个 ENCRYPT VP。您可以根据需要定义多个 ENCRYPT VP 以缩短启动数据库服务器所需的时间。

使用带有 encrypt 关键字的 VPCLASS 配置参数可配置加密 VP。例如，要添加 5 个 ENCRYPT VP，请如下所示在 onconfig 文件中添加信息：

```
VPCLASS encrypt,num=5
```

您可使用 gadmin 实用程序修改相同的信息，如下所示：

```
gadmin -p 5 encrypt
```

有关更多信息，请参阅《GBase 8s 管理员参考》中的配置参数和 gadmin 实用程序主题。

光虚拟处理器

虚拟处理器的可选类 (OPT) 仅与 Optical Subsystem 一起使用。如果 STAGEBLOB 配置参数存在，那么 Optical Subsystem 在可选类中启动一个虚拟处理器。

审计虚拟处理器

通过将 onconfig 文件中的 ADTMODE 参数设置为 1 来打开审计方式时，数据库服务器将启动审计类 (ADT) 中的一个虚拟处理器。

综合性虚拟处理器

综合性虚拟处理器将为系统调用的请求（如获取有关当前用户或主机系统名的信息）提供服务，这些系统调用可能需要非常大的堆栈。此虚拟处理器上只可运行一个线程；该线程将通过 128 KB 的堆栈执行。

Basic Text Search 虚拟处理器

要运行 Basic Text Search 查询，需要使用 Basic Text Search 虚拟处理器。

必须先创建 Basic Text Search (BTS) 虚拟处理器，然后才能创建 Basic Text Search 索引并对其运行查询。

创建 Basic Text Search 索引时，会自动添加 Basic Text Search 虚拟处理器。

Basic Text Search 虚拟处理器运行时不会让步；它一次处理一个索引操作。要同时运行多个 Basic Text Search 索引操作和查询，请创建额外的 Basic Text Search 虚拟处理器。

将 VPCLASS 配置参数与 BTS 关键字一起使用以配置 Basic Text Search 虚拟处理器。例如，要添加五个 BTS 虚拟处理器，请将以下行添加到 onconfig，然后重新启动数据库服务器：

```
VPCLASS bts,num=5
```

使用 gadmin -p 命令可动态添加 BTS 虚拟处理器，例如：

```
gadmin -p 5 bts
```

MQ 消息传递虚拟处理器

要使用 MQ 消息传递，需要 MQ 虚拟处理器。

必须先创建 MQ 虚拟处理器，然后才能使用 MQ 消息传递。

执行 MQ 消息传递事务时，将自动创建 MQ 虚拟处理器。

MQ 虚拟处理器运行时不带生成器功能，所以一次处理一个操作。要同时执行多个 MQ 消息传递事务，请创建多个 MQ 虚拟处理器。

在 VPCLASS 配置参数中使用 MQ 关键字可配置 MQ 虚拟处理器。例如，要添加五个 MQ 虚拟处理器，请将以下行添加到 onconfig，然后重新启动数据库服务器：

```
VPCLASS mq,noyield,num=5
```

有关 VPCLASS 配置参数的更多信息，请参阅《GBase 8s 管理员参考》。有关 MQ 消息传递的更多信息，请参阅《GBase 8s Database Extensions 用户指南》。

Web 要素服务虚拟处理器

要为地理空间数据使用 Web 要素服务，需要 Web 要素服务虚拟处理器。

必须先创建 WFS 虚拟处理器，然后才能使用 WFS。

运行 WFS 例程时，将自动创建 WFS 虚拟处理器。

WFS 虚拟处理器运行时不带生成器功能，所以一次处理一个操作。要同时运行多个 WFS 例程，请创建多个 WFS 虚拟处理器。

在 VPCLASS 配置参数中使用 WFSVP 关键字可配置 WFS 虚拟处理器。例如，要添加五个 WFS 虚拟处理器，请将以下行添加到 onconfig，然后重新启动数据库服务器：

```
VPCLASS wfsvp,noyield,num=5
```

有关 VPCLASS 配置参数的更多信息，请参阅《GBase 8s 管理员参考》。有关 WFS 的更多信息，请参阅《GBase 8s Database Extensions 用户指南》。

XML 虚拟处理器

要执行 XML 发布，需要 XML 虚拟处理器。

必须先创建 XML 虚拟处理器，然后才能执行 XML 发布。

运行 XML 函数时，将自动创建 XML 虚拟处理器。

XML 虚拟处理器一次只能运行一个 XML 函数。要同时运行多个 XML 函数，请创建多个 XML 虚拟处理器。

在 VPCLASS 配置参数中使用 IDSXMLVP 关键字可配置 XML 虚拟处理器。例如，要添加五个 XML 虚拟处理器，请将以下行添加到 onconfig，然后重新启动数据库服务器：

```
VPCLASS idsxmlvp,num=5
```

使用 gadmin -p 命令可动态添加 XML 虚拟处理器，例如：

```
gadmin -p 5 idsxmlvp
```

有关 VPCLASS 配置参数和 gadmin 实用程序的更多信息，请参阅《GBase 8s 管理员参考》。有关 XML 发布的更多信息，请参阅《GBase 8s Database Extensions 用户指南》。

3.2 管理虚拟处理器

这些主题将描述如何设置会影响数据库服务器虚拟处理器的配置参数，以及如何启动和停止虚拟处理器。

有关虚拟处理器类的描述以及有关必须为每个类指定多少虚拟处理器的建议，请参阅虚拟处理器和线程。

3.2.1 设置虚拟处理器配置参数

以 **root** 或 **gbasedbt** 用户身份，使用文本编辑器为数据库服务器虚拟处理器设置配置参数。

要实现对配置参数所做的任何更改，您必须关闭并重新启动数据库服务器。有关更多信息，请参阅设置共享内存。

使用文本编辑器设置虚拟处理器参数

您可以随时使用文本编辑器程序来设置 ONCONFIG 参数。可以使用编辑器定位希望更改的参数，输入新的值，然后重新将文件写入磁盘。

下表列出了用于配置虚拟处理器的 ONCONFIG 参数。有关这些参数如何影响虚拟处理器的更多信息，请参阅虚拟处理器类。

表 1. 用于配置虚拟处理器的配置参数

参数	子参数	用途
MULTIPROCESSOR		指定在多处理器计算机上运行
NETTYPE		指定网络协议线程（和虚拟处理器）的参数
SINGLE_CPU_VP		指定运行单个 CPU 虚拟处理器
VPCLASS	adm kio shm adt lio soc aio msc str cpu ntk tli encrypt opt jvp pio	指定虚拟处理器的预定义类名。例如，`jvp` 指定的是 Java™ 虚拟处理器 (JVP)。
VPCLASS	num=*number*	指定数据库服务器启动的指定类的虚拟处理器数
VPCLASS	max=*number*	指定类所允许的最大虚拟处理器数
VPCLASS	noage	指定禁用迟滞优先级
VPCLASS	aff = (*processor_number, start_range- end_range, start_range - end_range / increment*)	指定 CPU 的虚拟处理器指定情况（如果处理器亲缘关系可用）
VPCLASS	*user_defined*	指定用户定义的虚拟处理器
VPCLASS	noyield	指定非中止虚拟处理器
VP_MEMORY_CACHE_KB		加速对内存块的存取。

指定虚拟处理器类

使用 VPCLASS 配置参数可指定虚拟处理器 (VP) 的类、创建用户定义的虚拟处理器，以及指定如下选项：服务器启动的 VP 数、允许用于类的最大 VP 数，以及处理器亲缘关系可用时为 CPU 指定的 VP 数。

可以指定最大长度为 128 字节的 VPCLASS 名称。VPCLASS 名称必须以字母或下划线开头且必须包含字母、数字、下划线或 $ 字符。

禁用优先级迟滞 (UNIX)

使用带有 *noage* 选项的 VPCLASS 参数可禁用允许该功能的平台上的进程优先级迟滞。

有关 UNIX™ 上这些数据库服务器参数的推荐值，请参阅机器说明文件。

3.2.2 启动和停止虚拟处理器

当您启动数据库服务器时，oninit 实用程序将启动您已直接和间接指定了数量和类型的虚拟处理器。主要通过 ONCONFIG 参数配置虚拟处理器，并且对于网络虚拟处理器，那么通过 sqlhosts 文件或注册表中的参数来配置。有关虚拟处理器类的描述，请参阅虚拟处理器类。

可以使用数据库服务器来启动最多 1000 个虚拟处理器。

当数据库服务器处于联机方式后，如果需要，您可以启动其他虚拟处理器以提高性能。有关更多信息，请参阅在联机方式下添加虚拟处理器。

当数据库服务器处于联机方式，那么您可以删除 CPU 和用户定义类的虚拟处理器。有关更多信息，请参阅删除 CPU 和用户定义的虚拟处理器。

要终止数据库服务器并且从而终止所有的虚拟处理器，请使用 gadmin -k 命令。有关使用 gadmin -k 的更多信息，请参阅《GBase 8s 管理员参考》。

在联机方式下添加虚拟处理器

当数据库服务器处于联机方式时，您可以启动以下类的其他虚拟处理器：CPU、AIO、PIO、LIO、SHM、STR、TLI、SOC、JVP 和用户定义。数据库服务器自动为 LIO 和 PIO 类的每一个类启动一个虚拟处理器，除非使用了镜像，如果使用了镜像，那么数据库服务器将启动两个虚拟处理器。

可以使用 gadmin 实用程序的 -p 选项启动这些额外的虚拟处理器。

可以使用 gadmin 实用程序的 -p 选项或者 ISA 启动这些额外的虚拟处理器。

还可以启动用户定义类的附加虚拟处理器来运行用户定义的例程。有关用户定义的虚拟处理器的更多信息，请参阅将 UDR 指定给用户定义的虚拟处理器类。

在联机方式下使用 gadmin 添加虚拟处理器

使用 **gadmin** 命令的 -p 选项可在数据库服务器处于联机方式时添加虚拟处理器。用正数指定希望添加的虚拟处理器数。作为选择，您可以在虚拟处理器数前加上一个加号 (+)。在这个数字后面，请以小写字母指定虚拟处理器的类。例如，以下两个命令中任意一个都可启动 AIO 类中的其他四个虚拟处理器：

```
gadmin -p 4 aio

gadmin -p +4 aio
```

gadmin 实用程序将立即启动附加的虚拟处理器。

一次只可以将虚拟处理器添加到一个类。要为另一个类添加虚拟处理器，您必须再次运行 gadmin。

添加网络虚拟处理器

添加网络虚拟处理器时，可添加轮询线程，其中每个轮询线程都需要其自己的虚拟处理器才能运行。

如果尝试在数据库服务器处于联机方式时为协议添加轮询线程，并且在 NETTYPE 配置参数中指定轮询线程在 CPU 类中运行，那么在没有 CPU 虚拟处理器可用于运行新轮询线程时，数据库服务器不会启动新轮询线程。

在以下示例中，轮询线程总共处理 240 个连接：

```
NETTYPE ipcshm,4,60,CPU # Configure poll thread(s) for nettype
```

对于 ipcshm，轮询线程的数量对应于内存段的数量。例如，如果 NETTYPE 设置为 3,100 并且您只想要一个轮询线程，那么请将该轮询线程设置为 1,300。

删除 CPU 和用户定义的虚拟处理器

当数据库服务器处于联机方式中，那么您可以使用 **gadmin** 实用程序的 -p 选项来删除或终止 CPU 或用户定义类的虚拟处理器。

删除 CPU 虚拟处理器

紧跟 gadmin 命令之后，请指定一个负数，该数值是您想要删除的虚拟处理器的数量，然后以小写字母指定 CPU 类。例如，以下命令删除两个 CPU 虚拟处理器：

```
% gadmin -p -2 cpu
```

如果您尝试删除一个正在运行轮询线程的 CPU 虚拟处理器，那么您将接收以下消息：

```
gadmin: failed when trying to change the number of cpu virtual processor by -number.
```

有关更多信息，请参阅在 CPU 或网络虚拟处理器上运行轮询线程。

删除用户定义的虚拟处理器

在 gadmin 命令的后面，请指定一个负数（该数字是您希望删除的虚拟处理器的数量），然后以小写字母指定用户定义的类。例如，以下命令将删除类 *usr* 的两个虚拟处理器：

```
gadmin -p -2 usr
```

有关如何创建用户定义的虚拟处理器类以及如何将用户定义的例程指定给该类，请参阅用户定义的虚拟处理器类。

3.2.3 监视虚拟处理器

监视虚拟处理器以确定为数据库服务器配置的虚拟处理器数对于当前活动程度是否最佳。

有关 gstat -g 命令输出的示例，请参阅《GBase 8s 管理员参考》中有关 gstat 实用程序的信息。

3.2.4 使用命令行实用程序监视虚拟处理器

可以使用以下 gstat -g 选项来监视虚拟处理器：

- gstat -g ath 命令
- gstat -g glo 命令
- gstat -g ioq 命令
- gstat -g rea 命令

gstat -g ath 命令

gstat -g ath 命令显示了有关系统线程和虚拟处理器类的信息。

gstat -g glo 命令

使用 gstat -g glo 命令可显示有关当前正在运行的每个虚拟处理器的信息，以及有关每个虚拟处理器类的累积统计信息。有关 gstat -g glo 输出的示例，请参阅《GBase 8s 管理员参考》中有关 gstat 实用程序的信息。

gstat -g ioq 命令

使用 gstat -g ioq 选项可确定是否必须分配更多虚拟处理器。命令 gstat -g ioq 显示有关 I/O 队列的长度和其他统计信息。

如果 I/O 队列的长度不断增加，那么 I/O 请求的累积速度将超过 AIO 虚拟处理器处理请求的速度。如果 I/O 队列的长度继续显示 I/O 请求不断累积，那么考虑添加 AIO 虚拟处理器。

有关 gstat -g ioq 输出的示例，请参阅《GBase 8s 管理员参考》中的信息。

gstat -g rea 命令

使用 gstat -g rea 选项监视就绪的队列中的线程数。如果在就绪队列中虚拟处理器类（例如，CPU 类）的线程数量不断增加，那么可能需要将更多此类虚拟处理器添加到配置中。

有关 gstat -g rea 输出的示例，请参阅《GBase 8s 管理员参考》中的信息。

使用 SMI 表监视虚拟处理器

查询 **sysvpprof** 表以获得有关当前正在运行的虚拟处理器的信息。此表包含以下各列。

列	描述
vpid	虚拟处理器的标识号

列	描述
class	虚拟处理器类
usercpu	用户 CPU 已用分钟数
syscpu	系统 CPU 已用分钟数

3.3 共享内存

这些主题描述了数据库服务器共享内存的内容、确定共享内存区域大小的因素，以及数据移入和移出共享内存的方式。有关如何更改确定共享内存分配的数据库服务器配置参数的信息，请参阅管理共享内存。

3.3.1 共享内存

共享内存是允许数据库服务器线程和进程通过共享对内存池的访问权来共享数据的一种操作系统功能。数据库服务器将共享内存用于以下用途：

- 要减少内存使用和磁盘 I/O
- 执行进程间的高速通信

共享内存使数据库服务器能够减少总体内存使用量，因为参与进程（在此情况下即虚拟处理器）不需要保留共享内存中数据的专用副本。

共享内存将减少磁盘 I/O，因为缓冲区（作为公共池受管）将在整个数据库服务器范围内清空，而不是为每个进程清空。而且，虚拟处理器可以经常避免从磁盘读取数据，因为数据已经作为较早读取操作的结果存在于共享内存中了。减少磁盘 I/O 将减少执行时间。

共享内存提供最快的进程间通信方法，因为它以内存传送的速度处理读写消息。

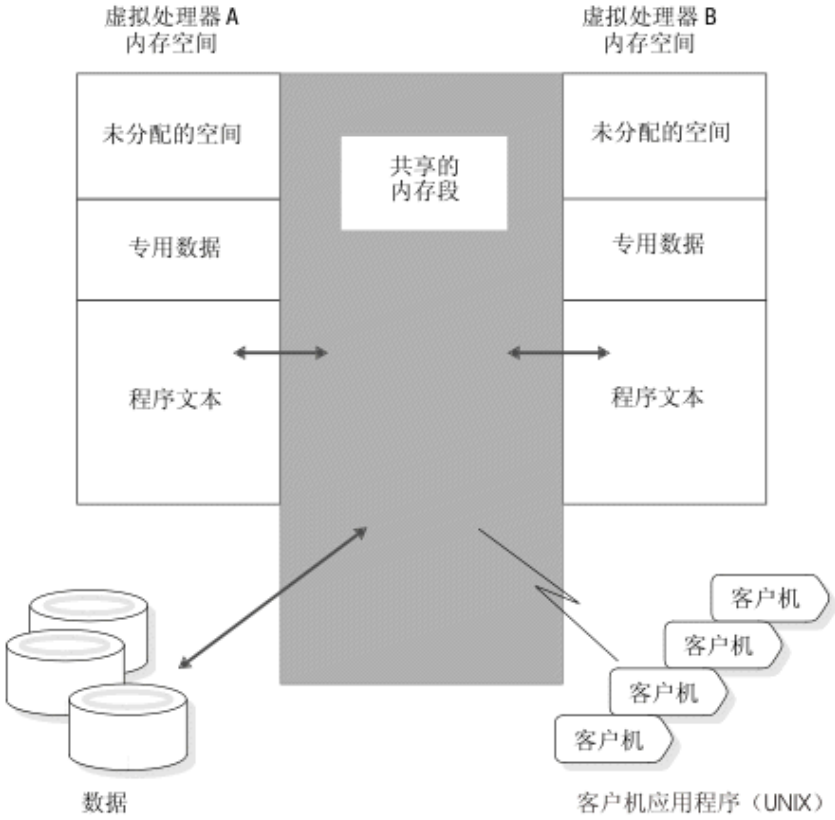
3.3.2 共享内存使用

数据库服务器将共享内存用于以下用途：

- 使虚拟处理器和实用工具能够共享数据
- 为使用 IPC 通信的本地客户机应用程序提供快速通信通道

下图说明了共享内存方案。

图: 数据库服务器使用共享内存的方式



共享内存分配

数据库服务器将创建共享内存的以下部分：

- *常驻*部分
- *虚拟*部分
- IPC *通信*或消息部分

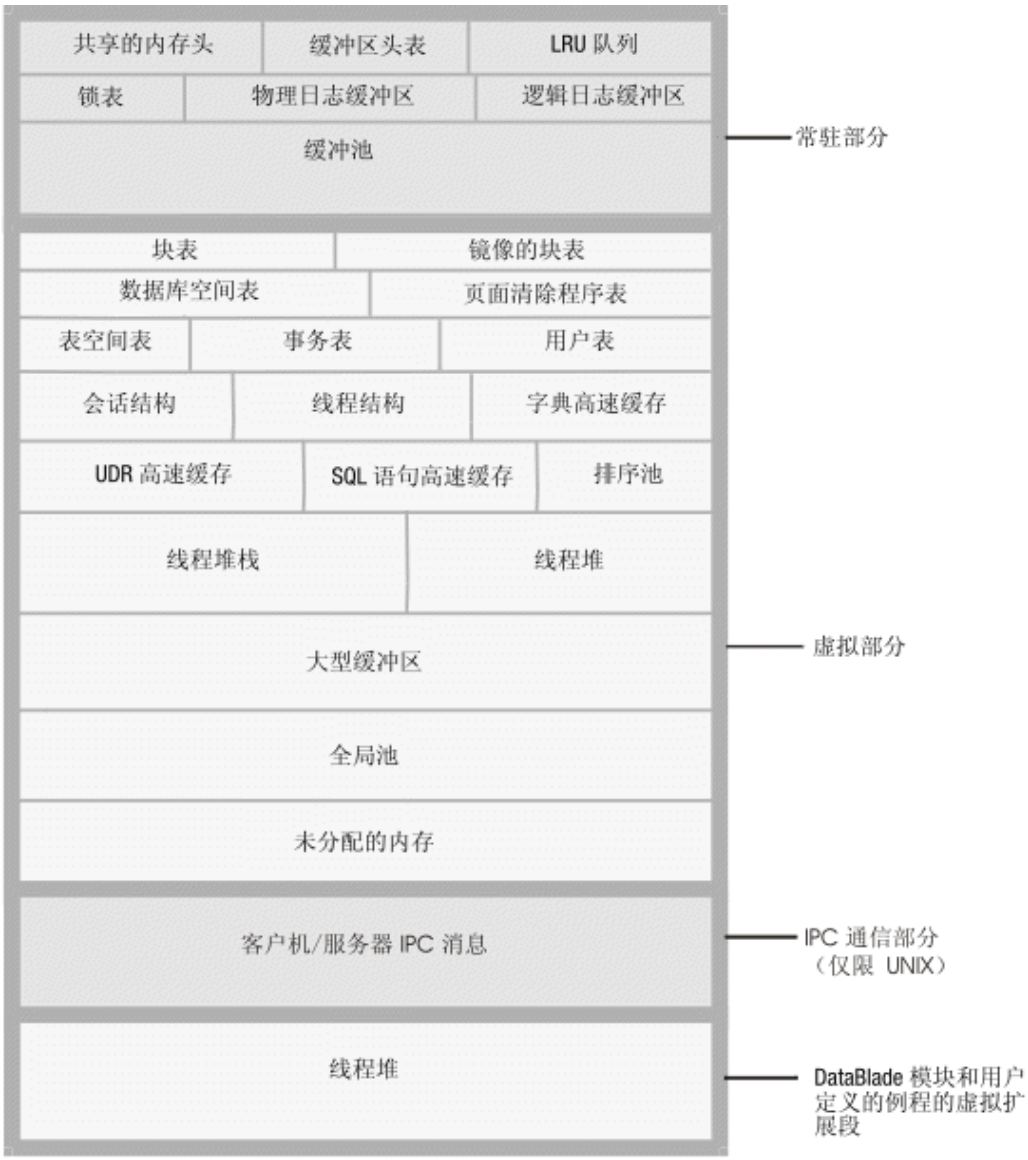
  如果 sqlhosts 文件指定了共享内存通信，那么数据库服务器将为通信部分分配内存。
- *虚拟扩展*部分

数据库服务器根据需要将操作系统段添加到共享内存的虚拟部分和虚拟扩展部分。

有关平台的共享内存设置的更多信息，请参阅机器说明。下图显示了共享内存各部分的内容。

所有的数据库服务器虚拟处理器都可以访问相同的共享内存段。每个虚拟处理器通过保留其自身对共享内存资源（如缓冲区、锁定和锁存器）的指针集来管理其工作。在将数据库服务器从脱机方式转换到静默、管理或联机方式时，虚拟处理器将连接共享内存。数据库服务器将使用锁定和锁存器来管理多个线程对共享内存资源的并发访问。

图：数据库服务器共享内存的内容



共享内存大小

数据库服务器共享内存的每个部分都由一个或多个操作系统内存段组成，每个内存段分成一系列大小为 4 KB 的块，并由位图进行管理。

gstat 实用程序输出的标题行包含数据库服务器共享内存的大小（以 KB 表示）。您还可以使用 gstat -g seg 监视数据库服务器为共享内存的每一部分分配多少内存。有关如何使用 gstat 的信息，请参阅《GBase 8s 管理员参考》。

可以在 onconfig 文件中设置 SHMTOTAL 参数来限制数据库服务器可以在计算机或节点上安排的内存开销量。SHMTOTAL 参数指定数据库服务器可用于所有内存分配的共享内存总量。 然而，如果数据库服务器需要的内存量大于 SHMTOTAL 中设置的量，那么某些操作可能会失败。如果发生这种情况，数据库服务器会在消息日志中显示以下消息：

```
size of resident + virtual segments x + y > z

total allowed by configuration parameter SHMTOTAL
```

此外，数据库服务器会将错误消息返回到已启动违规操作的应用程序中。例如，如果数据库服务器在尝试执行操作（例如，索引构建或散列连接）时需要的内存量大于在 SHMTOTAL 中指定的内存量，那么该服务器将向应用程序返回类似于以下某条消息的错误消息：

-567	Cannot write sorted rows.
-116	ISAM error: cannot allocate memory.

数据库服务器在发送了这些消息后会回滚任何由违规查询执行的部分结果。

内部操作（例如页清除程序或检查点活动）也会导致数据库服务器超过 SHMTOTAL 的上限。当发生这种情况时，数据库服务器会向消息日志发送一条消息： 例如，假设数据库服务器试图为页清除程序活动分配附加的内存但失败了。结果，数据库服务器会向消息日志发送类似于以下消息的信息：

17:19:13	Assert Failed: WARNING! No memory available for page cleaners
17:19:13	Who: Thread(11, flush_sub(0), 9a8444, 1)
17:19:13	Results: Database server may be unable to complete a checkpoint
17:19:13	Action: Make more virtual memory available to database server
17:19:13	See Also: /tmp/af.c4

在数据库服务器通知您分配附加的内存失败后，该服务器会回滚导致其超过 SHMTOTAL 限制的事务。一经回滚，操作就不会再因为内存不足而失败，数据库服务器将继续如平常一样处理事务。

超过 SHMTOTAL 时要执行的操作

当数据库服务器需要的内存量大于 SHMTOTAL 所允许的值时，将出现瞬时状态，这可能是因突发超过正常处理负载的活动所引起。只有导致数据库服务器暂时耗尽内存的操作才会暂时失败。其他操作继续以正常方式处理。

如果有消息定期指示数据库服务器需要的内存量大于 SHMTOTAL 所允许的值，那么表示您没有正确配置数据库服务器。减小 BUFFERPOOL 配置参数中的 DS_TOTAL_MEMORY 值或 **buffers** 值是一种可能的解决方案；增加 SHMTOTAL 的值是另一种解决方案。

3.3.3 连接到共享内存的进程

以下进程将连接到数据库服务器共享内存：

- 通过共享内存通信部分 (**ipcshm**) 与数据库服务器通信的客户机应用程序进程
- 数据库服务器虚拟处理器
- 数据库服务器实用程序

客户机如何连接到通信部分 (UNIX™)

通过共享内存 (nettype ipcshm) 与数据库服务器通信的客户机应用程序进程透明地连接到共享内存的通信部分。自动编译到应用程序的系统库函数使其能够连接到共享内存的通信部分。有关指定共享内存连接的信息，请参阅客户机/服务器通信和网络虚拟处理器。

如果没有设置 GBASEDBTSHMBASE 环境变量，那么客户机应用程序将连接到特定于平台的地址的通信部分。如果客户机应用程序连接到其他共享内存段（不是数据库服务器共享内存），那么用户可以将 GBASEDBTSHMBASE 环境变量设置为指定连接数据库服务器共享内存通信段所在的地址。当您指定处理共享内存通信段所在的地址时，可以防止数据库服务器与应用程序使用的其他共享内存段发生冲突。有关如何设置 GBASEDBTSHMBASE 环境变量的信息，请参阅《GBase 8s SQL 指南：参考》。

实用程序如何连接到共享内存

数据库服务器实用程序（如 gstat、gadmin 和 gtape）通过以下文件之一连接到共享内存。

操作系统	文件
UNIX™	$GBASEDBTDIR/etc/.infos.*servername*

变量 **servername** 是 onconfig 文件中 DBSERVERNAME 参数的值。实用程序从 **GBASEDBTSERVER** 环境变量中获得文件名的 servername 部分。

oninit 进程在启动数据库服务器时读取 onconfig 文件并创建文件 .infos.servername。该文件将在数据库服务器终止时除去。

虚拟处理器如何连接到共享内存

数据库服务器虚拟处理器在安装期间连接共享内存。在此过程中，数据库服务器必须满足以下两个要求：

- 确保所有虚拟处理器可以定位和访问相同的共享内存段
- 确保共享内存段位于同一台计算机上但与指定给其他数据库服务器实例（如果 有）的共享内存段不同的物理内存位置

数据库服务器使用两个配置参数（SERVERNUM 和 SHMBASE）以满足这些要求。

当虚拟处理器连接到共享内存时，它会执行以下主要步骤：

- 从 onconfig 文件访问 SERVERNUM 参数
- 使用 SERVERNUM 可计算共享内存键值
- 使用共享内存键值请求共享内存段

操作系统返回第一个共享内存段的共享内存标识。

- 引导操作系统将第一个共享内存段连接到其位于 SHMBASE 的进程空间
- 若需要，连接附加共享内存段以与第一个段邻接

获取共享内存段的键值

SERVERNUM 配置参数和 *shmkey* 的值（内部计算的数字）确定每个共享内存段的唯一键值。

要查看共享内存段的键值，请运行 gstat -g seg 命令。

当虚拟处理器请求操作系统连接第一个共享内存段时，它将提供唯一的键值来标识该段。作为回应，操作系统传回与键值相关联的共享内存段标识。使用此标识，虚拟处理器将请求操作系统将共享内存段连接到虚拟处理器地址空间。

指定连接第一个共享内存段的位置

onconfig 文件中的 SHMBASE 参数指定每个虚拟处理器连接第一个（或基本）共享内存段所在的虚拟地址。每个虚拟处理器在同一个虚拟地址上连接第一个共享内存段。此种情况使相同数据库服务器实例中的所有虚拟处理器都能够引用共享内存中的相同位置，而无需计算共享内存的地址。数据库服务器实例的所有共享内存地址都是 SHMBASE 的相对地址。

警告： 不更改 SHMBASE 的值。

SHMBASE 的值易受以下原因影响：

- SHMBASE 的特定值取决于平台以及处理器是 32 位还是 64 位的。SHMBASE 的值不是任意数，它将在虚拟处理器动态获取附加的内存空间时保证共享内存段的安全。
- 不同的操作系统在不同的虚拟地址上接纳附加的内存。有些体系结构扩展虚拟处理器数据段的最高虚拟地址以便接纳下一个段。在此情况下，数据段可能会发展成共享内存段。
- 某些版本的 UNIX™ 需要用户将虚拟地址的 SHMBASE 参数指定为零。零地址会通知 UNIX 内核其挑选连接共享内存段所在的最佳地址。然而，并不是所有 UNIX 体系结构都支持此选项。而且，在一些系统中，内核所做的选择可能不是最好的选择。

连接附加共享内存段

每个虚拟处理器必须连接到数据库服务器已经获取的总共享内存量。在虚拟处理器连接了每个共享内存段后，它将计算已经连接的共享内存数量以及剩余的共享内存数量。数据库服务器通过将共享内存头写入第一个共享内存段促进此进程。虚拟处理器可以深入到头的十六字节获取以下数据：

- 此数据库服务器的共享内存的总大小
- 每个共享内存段的大小

要连接附加共享内存段，虚拟处理器将从操作系统中请求这些段，很多地方就像它请求第一个段一样。然而，对于附加段，虚拟处理器对 *shmkey* 的先前值加 1。 虚拟处理器将引导操作系统连接位于从以下计算获得的地址的段：

```
SHMBASE +（seg_size x 已连接的段数）
```

虚拟处理器将重复此过程直到它获得总共享内存量。

假设初始的键值是 (SERVERNUM * 65536) + *shmkey*，那么数据库服务器在可以请求由同一台计算机上另一数据库服务器实例使用的共享内存键值前，最多请求 65,536 个共享内存段。

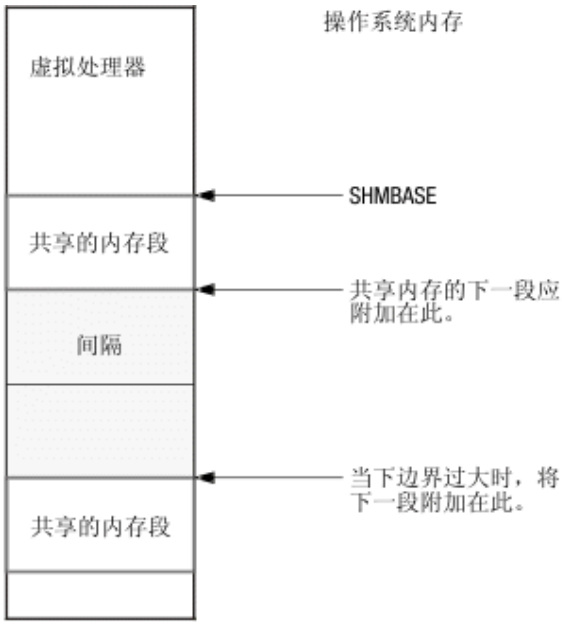
定义共享内存下边界地址

如果操作系统使用参数来定义共享内存的下边界地址，并且该参数未正确设置，那么会使共享内存段无法连续地连接。

下图说明了该问题。如果下边界地址小于上一个段的结束地址加上当前段的大小，那么操作系统将在上一个段的结束位置外的点上连接当前段。此操作将在两个段之间创建间隔。由于共享内存必须连接到虚拟处理器以使其看上去就像连续的内存，所以此间隔将产生问题。 数据库服务器将在这种情况发生时接收到错误。

要更正该问题，请检查指定下边界地址的操作系统内核参数，或者重新将内核配置成允许更大的共享内存段。

图: 共享内存下边界地址概述



3.3.4 常驻共享内存段

操作系统在运行在系统上的进程间切换时通常将各部分内存的内容与磁盘交换。然而，当一部分内存指定为*常驻*时，它不会与磁盘交换。使频繁访问的数据常驻在内存中可以提高性能，因为这样就可以减少访问该数据所需的磁盘 I/O 操作的数目。

数据库服务器请求操作系统在以下两个条件存在时将这些虚拟部分保持在物理内存中：

- 操作系统支持共享内存驻留。
- onconfig 文件中的 RESIDENT 参数已设置为 -1 或大于 0 的值。

警告： 当考虑是否将 RESIDENT 参数设置为 -1 时，应该考虑所有应用程序对共享内存的使用。为使用 GBase 8s 数据库服务器而锁定所有共享内存会对同一台计算机上的其他应用程序（如果有）的性能产生不利的影响。

有关 RESIDENT 配置参数的更多信息，请参阅《GBase 8s 管理员参考》。

3.3.5 共享内存的常驻部分

数据库服务器共享内存的常驻部分存储了下列数据结构，这些数据结构的大小在数据库服务器运行时不会更改：

- 共享内存头
- 缓冲池
- 逻辑日志缓冲区
- 物理日志缓冲区
- 锁表

共享内存头

共享内存头包含有关共享内存中所有其他结构的描述，包括内部表和缓冲池。

共享内存头还包含指向这些结构的位置的指针。当虚拟处理器首次连接到共享内存时，它会读取共享内存头中的地址信息以便引导到所有其他结构。

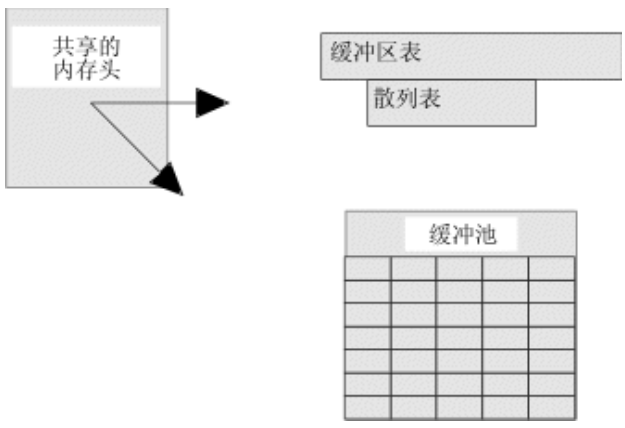
共享内存头的大小大约有 200 KB，但该大小会根据计算机平台而发生变化。您不能调整该头大小。

共享内存缓冲池

共享内存常驻部分中的缓冲池包含存储从磁盘读取的数据库空间页的缓冲区。缓冲池包含对共享内存的常驻部分的最大分配。

下图说明了共享内存头和缓冲池。

图: 共享内存缓冲池



使用 BUFFERPOOL 配置参数可指定有关缓冲池的信息，包括缓冲池中缓冲区的数量。要分配适当的缓冲区数，对于每个用户至少要启动 4 个缓冲区。对于 500 个以上的用户，最少要求 2000 个缓冲区。缓冲区数目过少会严重影响性能，因此必须监视数据库服务器并调节缓冲区数的值以确定可接受的值。如果非缺省页大小的缓冲池不存在，那么数据库服务器将自动创建大页缓冲区。

如果您正在创建非缺省页大小的数据库空间，那么该数据库空间必须具有对应的缓冲池。例如，如果创建页大小为 6 KB 的数据库空间，那么必须创建大小为 6 KB 的缓冲池。

自动 LRU（最近最少使用）调节可影响所有缓冲池并调整 BUFFERPOOL 配置参数中的 **lru_min_dirty** 和 **lru_max_dirty** 值。

有关设置 BUFFERPOOL 配置参数的更多信息，请参阅《GBase 8s 管理员参考》。

缓冲区状态通过缓冲区表跟踪。在共享内存中，缓冲区将组织到 FIFO/LRU 缓冲区队列中。缓冲区获取是通过使用锁存器（称为*互斥*）和锁访问信息来管理的。

缓冲区溢出至虚拟部分

由于 64 位寻址中的最大缓冲区数可以大到 231-1，所以共享内存的常驻部分可能不能在一个大的缓冲池中容纳所有这些缓冲区。在此情况下，数据库服务器共享内存的虚拟部分可能会容纳这些缓冲区中的一些。

缓冲区大小

每个缓冲区就是一个数据库服务器页的大小。

通常，数据库服务器以整页单位（缓冲区大小）来执行 I/O。从大缓冲区、BLOB 空间缓冲区或轻量级 I/O 缓冲区执行的 I/O 是例外。（请参阅大缓冲区和创建 BLOB 页缓冲区）

gstat -b 命令将显示有关缓冲区的信息。有关 gstat 的信息，请参阅《GBase 8s 管理员参考》。

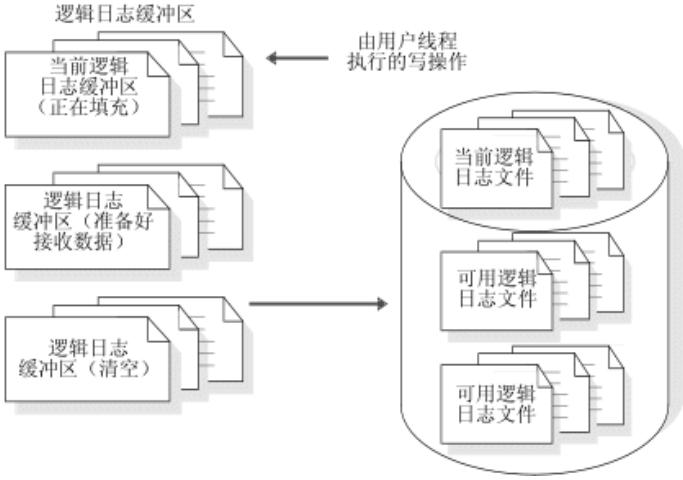
逻辑日志缓冲区

数据库服务器将使用逻辑日志来存储自上次数据库空间备份以来对数据库服务器所作更改的记录。 逻辑日志存储代表数据库服务器工作的逻辑单元的记录。逻辑日志包含以下五种类型日志记录（此外还有很多其他类型）：

- 用于所有数据库的 SQL 数据定义语句
- 用于通过日志记录创建的数据库的 SQL 数据操作语句
- 对数据库的日志记录状态所作更改的记录
- 检查点记录
- 对配置所作更改的记录

数据库服务器一次只使用逻辑日志缓冲区中的一个。此缓冲区是当前的逻辑日志缓冲区。在数据库服务器将当前的逻辑日志缓冲区清空到磁盘之前，它将使第二个逻辑日志缓冲区成为当前的缓冲区以便其可以在第一个缓冲区清空时继续写入。如果第二个逻辑日志缓冲区在第一个缓冲区完成清空前充满，那么第三个逻辑日志缓冲区将成为当前的缓冲区。下图说明了此过程。

图: 逻辑日志缓冲区及其与磁盘上的逻辑日志文件的关系



有关数据库服务器如何清空逻辑日志缓冲区的描述，请参阅清空逻辑日志缓冲区。

LOGBUFF 配置参数指定逻辑日志缓冲区的大小。如果存储的记录大于缓冲区的大小（例如，数据库空间中的 TEXT 或 BYTE 数据），那么小缓冲区可能会产生问题。逻辑日志缓冲区大小的推荐值是 64 KB。无论何时该设置小于推荐值，数据库服务器都将在服务器启动时建议另一个值。有关可指定给此配置参数的可能的值，请参阅《GBase 8s 管理员参考》。

有关 TEXT 和 BYTE 数据对共享内存缓冲区的影响的信息，请参阅缓冲区大对象数据。

物理日志缓冲区

数据库服务器使用物理日志缓冲区来容纳一些经过修改的数据库空间页的前映像。物理日志和逻辑日志记录中的前映像使数据库服务器能够在系统故障后将一致性复原到其数据库。

物理日志缓冲区实际上是两个缓冲区。双缓冲允许数据库服务器进程在其他缓冲区正清空到磁盘上的物理日志时写入活动物理日志缓冲区。有关数据库服务器如何清空物理日志缓冲区的描述，请参阅清空物理日志缓冲区。有关监视物理日志文件的信息，请参阅监视物理和逻辑日志记录活动。

onconfig 文件中的 PHYSBUFF 参数指定物理日志缓冲区的大小。一次写入物理日志缓冲区正好写满一页。如果物理日志缓冲区的指定大小不能按页大小均匀分隔，数据库服务器会将该大小向下四舍五入到可按页大小平均分隔的最近的值。虽然有些操作要求缓冲区即刻清空，但通常数据库服务器将在缓冲区充满时才将缓冲区清空到磁盘上的物理日志文件。因此，缓冲区的大小将确定数据库服务器必须每隔多少时间将缓冲区清空到磁盘一次。

物理日志缓冲区大小的缺省值为 512 KB。如果您决定使用更小的值，数据库服务器将显示一条消息，指示这样可能会无法达到最佳性能。使用小于 512 KB 的物理日志缓冲区只会影响性能，而不会影响事务完整性。

有关此配置参数的更多信息，请参阅《GBase 8s 管理员参考》。

高可用性数据复制缓冲区

数据复制要求两个数据库服务器实例（一个主实例和一个辅助实例）分别运行在两个计算机上。如果为数据库服务器实现数据复制，那么主数据库服务器首先将逻辑日志记录保存在数据复制缓冲区中，然后再将这些记录发送到辅助数据库服务器。数据复制缓冲区的大小总是和逻辑日志缓冲区的大小相同。有关逻辑日志缓冲区的大小的信息，请参阅上一主题逻辑日志缓冲区。有关如何使用数据复制缓冲区的更多信息，请参阅数据复制的工作原理。

锁表

锁将在用户线程在锁表中写入条目时创建。锁表是可用锁的池。单个事务可以拥有多个锁。有关锁定和与锁定相关联的 SQL 语句的说明，请参阅《GBase 8s SQL 指南：教程》。

存储在锁表中的以下信息描述锁：

- 拥有定的事务的地址
- 锁定的类型（互斥、更新、共享、字节或意向）
- 已锁定的页或行标识
- 放置锁定的表空间
- 有关已锁定字节（智能大对象的字节范围）的信息：
  - 智能大对象标识
  - 智能大对象中已锁定字节开始处的偏移量
  - 已锁定的字节数（从偏移处开始）

要指定锁定表的初始大小，请设置 LOCKS 配置参数。

如果会话分配的锁定数超过 LOCKS 配置参数中指定的值，那么数据库服务器会将锁定表的大小加倍，最多可达 15 倍。数据库服务器通过尝试在每次增加时使锁定表翻倍来增加锁定表的大小。但是，每次增加期间增加的量不得超过最大值。对于 32 位平台，每次最大可增加 100,000 个锁定。因此，32 位平台允许的最大锁定总数为 8,000,000（启动锁定的最大数目）+ 99（动态锁定表扩展的最大数目）x 100,000（每个锁定表扩展添加的最大锁定数目）。对于 64 位平台，每次最大可增加 1,000,000 个锁定。因此，允许的最大锁定总数为 500,000,000（启动锁定的最大数目）+ 99（动态锁定表扩展的最大数目）x 1,000,000（每个锁定表扩展添加的最大锁定数目）。

可使用 DEF_TABLE_LOCKMODE 配置参数为新表的页或行设置锁定方式。

这些锁定能够使会话在已落实或回滚了并发事务之后再读取数据。对于使用事务日志记录创建的数据库，可使用 onconfig 文件中的 USELASTCOMMITTED 配置参数来指定数据库服务器是否使用上次落实的数据版本。上次落实的数据版本是任何更新发生之前存在的数据版本。使用 USELASTCOMMITTED 配置参数设置的值会覆盖在 SQL 语句 SET ISOLATION TO COMMITTED READ 中指定的隔离级别。有关使用 USELASTCOMMITTED 配置参数的更多信息，请参阅《GBase 8s 管理员参考》中有关配置参数的主题。

3.3.6 共享内存的虚拟部分

共享内存的虚拟部分可以由数据库服务器扩展并且可以由操作系统分页到磁盘。当数据库服务器执行时，它可以自动按需将附加的操作系统段连接到虚拟部分。

管理共享内存的虚拟部分

数据库服务器使用内存*池*跟踪类型和大小类似的内存分配。在池中保留相关的内存分配将减少内存分段存储。它还使数据库服务器能够一次释放大块的内存，与释放组成池的每个块相对。

所有的会话都有一个或多个内存池。当数据库服务器需要内存时，它会先查看指定的池。如果池中可用的内存不足以满足请求，那么数据库服务器将从系统池添加内存。如果数据库服务器不能在系统池中找到足够的内存，那么它会动态地给虚拟部分分配更多的段。

数据库服务器通过已链接的列表跟踪可用空间的池，为自己的每个子系统（会话池、堆栈、堆、控制块、系统目录、SPL 例程高速缓存、SQL 语句高速缓存、排序池和消息缓冲区）分配虚拟共享内存。当数据库服务器分配内存的部分时，它将首先在池的可用列表中搜索大小足够的*段*。如果服务器未找到段，那么会将新的块从虚拟部分带入池中。如果释放了内存，那么被释放内存将作为可用段返回到池中并留在其中，直到池被删除。例如，当数据库服务器启动了客户机应用程序会话时，它将为会话池分配内存。当会话终止时，数据库服务器会将已分配的内存作为可用段返回。

共享内存虚拟部分的大小

使用配置参数可指定共享内存虚拟部分的初始大小、以后要添加的段的大小以及可用于 PDQ 查询的内存量。

要指定虚拟共享内存部分的初始大小，请设置 SHMVIRTSIZE 配置参数。要指定以后添加到虚拟共享内存的段的大小，请设置 SHMADD 和 EXTSHMADD 配置参数。

要指定 PDQ 查询可用的内存量，请设置 DS_TOTAL_MEMORY 参数。

如果要增加可用于非 PDQ 查询的内存量，并且 PDQ 优先级设置为 0（零），那么可以通过以下任一方式更改该内存量：

- 设置 DS_NONPDQ_QUERY_MEM 配置参数
- 运行 gadmin -wm 或 gadmin -wf 命令

例如，如果使用 gadmin 实用程序，请按以下示例所示指定值：

```
gadmin -wf DS_NONPDQ_QUERY_MEM=500
```

DS_NONPDQ_QUERY_MEM 的最小值为 128 KB。支持的最大值为 DS_TOTAL_MEMORY 值的25%。

共享内存虚拟部分的组件

共享内存的虚拟部分存储以下数据：

- 内部表
- 大缓冲区

- 会话数据
- 线程数据（堆栈和堆）
- 数据分发高速缓存
- 字典高速缓存
- SPL 例程高速缓存
- SQL 语句高速缓存
- 排序池
- 全局池

共享内存内部表

数据库服务器共享内存包含跟踪共享内存资源的七个内部表。共享内存的内部表如下所示：

- 缓冲区表
- 块表
- 数据库空间表
- 页清除程序表
- 表空间表
- 事务表
- 用户表

缓冲区表

缓冲区表将跟踪共享内存池中单个缓冲区的地址和状态。当使用缓冲区时，该缓冲区将包含来自磁盘的数据或索引页的映像。有关磁盘页的用途和内容的更多信息，请参阅页。

缓冲区表中的每个缓冲区都包含以下缓冲区管理所需的控制信息：

缓冲区状态
: 缓冲区状态可以描述为空的、未修改的或已修改的。 未修改的缓冲区包含数据，但可以覆盖这些数据。 已修改的（脏）缓冲区包含必须在可以覆盖前写入磁盘的数据。

当前®锁定访问级别
: 缓冲区将根据用户线程正在执行的操作类型接收锁定访问级别。数据库服务器支持两种缓冲区锁定访问级别：共享和互斥。

等待缓冲区的线程
: 每个缓冲区头都保留一系列正在等待缓冲区的线程以及每个处于等待状态的线程所要求的锁访问级别。

每个数据库服务器缓冲区在缓冲区表中都有一个条目。

有关数据库服务器缓冲区的信息，请参阅共享内存的常驻部分。有关如何监视缓冲区的信息，请参阅监视缓冲区。

数据库服务器根据已分配的缓冲区数确定缓冲区表散列表中的条目数。散列值的最大数是小于 **buffers** 值的 2 的最大幂数，它是在某个 BUFFERPOOL 配置参数字段中指定的。

块表

块表将跟踪数据库服务器中的所有块。如果已启用了镜像，那么还会在启动共享内存时创建相应的镜像块表。镜像块表跟踪所有镜像块。

共享内存中的块表包含了使数据库服务器能够在磁盘上找到块的信息。这些信息包括数据库空间中初始块的编号以及下一个块的编号。标志也描述了块的状态：是镜像块还是主块；是脱机、联机还是恢复方式；以及此块是否是 BLOB 空间的一部分。

块表中的最大条目数可能受操作系统所允许的每个进程的最大文件描述符数的限制。通常可以使用操作系统内核配置参数指定每个进程的文件描述符数。

数据库空间表

数据库空间表跟踪数据库服务器中的存储器空间。数据库空间表信息包含有关每个数据库空间的以下信息：

- 数据库空间编号
- 数据库空间名称和所有者
- 数据库空间的镜像状态（已镜像或未镜像）
- 创建数据库空间的日期和时间

如果存储空间是 BLOB 空间，那么标志将指示该 BLOB 空间所在的介质：是磁性介质还是可移动介质。如果存储空间是个智能大对象空间，那么它将包含跟踪智能大对象的元数据以及包含用户数据的大连续块的页的内部表。

如果存储空间是个 BLOB 空间，那么标志将指示 BLOB 空间所在的介质：是磁性的、可移动的还是光学的介质。如果存储空间是个智能大对象空间，那么它将包含跟踪智能大对象的元数据以及包含用户数据的大连续块的页的内部表。

页清除程序表

页清除程序表将跟踪每个页清除程序线程的状态和位置。页清除程序线程数由 onconfig 文件中的 CLEANERS 配置参数指定。要获得有关应指定多少页清除程序线程的建议，请参阅《GBase 8s 管理员参考》中有关配置参数的章节。

不管 onconfig 文件中 CLEANERS 参数指定的页清除程序线程数是多少，页清除程序表始终包含 128 个条目。

有关监视页清除程序线程活动的信息，请参阅《GBase 8s 管理员参考》中有关 gstat -F 选项的信息。

表空间表

表空间表跟踪数据库服务器实例中的所有活动的表空间。活动的表空间是当前由数据库会话使用的表空间。每个活动的表空间占表空间表中的一个条目。活动的表空间包括数据库表、临时表以及内部控制表（如系统目录表）。每个表空间表条目都包含有关表空间的头信息，表空间名以及指向磁盘上数据库空间中的表空间 **tblspace** 的指针。（共享内存的活动表空间表与表空间 **tblspace** 不同。）有关监视表空间的信息，请参阅监视表空间和扩展数据块。

数据库服务器将为每个数据库空间管理一个表空间。

事务表

事务表将跟踪数据库服务器中的所有事务。

从事务表派生的跟踪信息在 gstat -x 显示中显示。

数据库服务器将根据当前的事务数自动增加事务表中的条目数（最多 32,767 个）。

有关事务和用于事务的 SQL 语句的更多信息，请参阅《GBase 8s SQL 指南：教程》、《GBase 8s SQL 指南：参考》和《GBase 8s SQL 指南：语法》。

仅限 UNIX： 事务表同样明确支持 X/Open 环境。支持 X/Open 环境需要 TP/XA。

用户表

用户表跟踪所有用户线程和系统线程。每个客户机会话根据指定的并行性级别有一个主线程以及零到许多个辅助线程。系统线程包含一个监视和控制检查点的线程、一个处理 gadmin 命令的线程、B 型树扫描程序线程以及页清除程序线程。

数据库服务器会按需增加用户表中的条目数。可以用 gstat -u 命令监视用户线程。

大缓冲区

大缓冲区是由多个页组成的单个缓冲区。实际页数将取决于平台。数据库服务器将分配大缓冲区以提高大量读写时的性能。

无论何时数据库服务器向磁盘写入多个物理上相邻的页，它都使用大缓冲区。例如，数据库服务器尝试使用大缓冲区执行一系列顺序读（轻度扫描）或将存储在数据库空间的简单大对象读入共享内存。

用户对大缓冲区没有控制权。如果数据库服务器使用轻度扫描，那么它将从共享内存分配大缓冲区。

会话数据

当客户机应用程序请求连接到数据库服务器时，数据库服务器开始与客户机的*会话*并在称为*会话控制块*的共享内存中为该会话创建数据结构。会话控制块存储会话标识、用户标识、客户机进程标识、主机名称以及各种状态标志。

数据库服务器会按需为会话数据分配内存。

线程数据

当客户机连接到数据库服务器时，除了启动会话以外，数据库服务器还会启动主会话线程并在共享内存中为其创建*启动控制块*。

数据库服务器还代表其自身启动内部线程并为这些线程创建线程控制块。当数据库服务器从运行一个线程切换为运行另一个线程（上下文切换）时，它在线程控制块中保存有关线程的信息（如寄存器内容、程序计数器（下一个指令的地址）以及全局指针）。有关线程控制块以及其使用方法的更多信息，请参阅上下文切换。

数据库服务器会按需为线程控制块分配内存。

堆栈

数据库服务器中的每个线程在共享内存的虚拟部分中都有其自己的堆栈区域。用户线程的堆栈空间大小由 onconfig 文件中的 STACKSIZE 参数指定。堆栈的缺省大小为 32 KB。如有必要，可通过更改 STACKSIZE 的值来更改所有用户线程的堆栈大小。有关设置堆栈大小的信息和警告，请参阅《GBase 8s 管理员参考》中有关配置参数的主题中的 STACKSIZE。

要更改特定会话主线程的堆栈大小，请设置 GBASEDBTSTACKSIZE 环境变量。GBASEDBTSTACKSIZE 的值将覆盖特定用户的 STACKSIZE 的值。有关如何覆盖特定用户的堆栈大小的信息，请参阅《GBase 8s SQL 指南：参考》中有关 GBASEDBTSTACKSIZE 环境变量的描述。

要更加安全地更改堆栈空间的大小，请使用 GBASEDBTSTACKSIZE 环境变量而不要更改配置参数 STACKSIZE。GBASEDBTSTACKSIZE 环境变量只会影响一个用户的堆栈空间，并且该变量不太可能影响最初未测量的新的客户机应用程序。

堆

每个线程有一个堆来容纳其在运行时创建的数据结构。堆将在创建线程时动态分配。线程堆的大小是不可配置的。

数据分发高速缓存

数据库服务器使用在 MEDIUM 或 HIGH 方式下由 UPDATE STATISTICS 语句生成的分布统计信息来确定成本最低的查询计划。当数据库服务器第一次访问有关特定列的分布统计信息时，它将从磁盘上的 **sysdistrib** 系统目录表中读取分布统计信息并且将这些统计信息存储在数据分发高速缓存中。然后可以读取这些统计信息以便优化访问列的后续查询。

如果这些统计信息有效地从数据分发高速缓存中存储和访问，性能将会提高。可使用 DS_HASHSIZE 和 DS_POOLSIZE 配置参数来配置数据分发高速缓存的大小。

字典高速缓存

当会话执行需要访问系统目录表的 SQL 语句时，数据库服务器将从系统目录表中读取数据。数据库服务器将每个查询的表的目录数据存储在各个结构中，这样数据库服务器在对该表进行子查询期间就能更有效的访问这些结构。这些结构将创建在共享内存的虚拟部分中以供所有会话使用。这些结构构成了字典高速缓存。

可使用 DD_HASHSIZE 和 DD_HASHMAX 配置参数来配置字典高速缓存的大小。

SQL 语句高速缓存

SQL 语句高速缓存将减少查询的内存使用量和准备时间。数据库服务器使用 SQL 语句高速缓存来存储用户执行的已解析和优化的 SQL 语句。当用户执行 SQL 语句高速缓存中存储的语句时，数据库服务器不再对该语句进行解析和优化，从而提高了性能。

排序内存

以下数据库操作可以使用大量的共享内存虚拟部分来将数据排序。

- 涉及连接、分组、聚集和排序操作的决策支持查询
- 索引构建
- SQL 中的 UPDATE STATISTICS 语句

数据库服务器为排序所分配的虚拟共享内存量将取决于要排序的行数、行的大小以及其他因素。

SPL 例程和 UDR 高速缓存

数据库服务器将 SPL 例程转换为可执行格式并将该例程存储在 UDR 高速缓存中，任何会话都可以在其中访问该例程。

当第一次需要会话访问 SPL 例程或其他用户定义的例程时，数据库服务器将从系统目录表中读取定义，并将定义存储到 UDR 高速缓存中。

可使用 PC_HASHSIZE 和 PC_POOLSIZE 配置参数来配置 UDR 高速缓存的大小。

全局池

全局池存储对于数据库服务器是全局的结构。例如，全局池包含消息队列，网络通信的轮询线程在该处存储来自客户机的消息。**sqlexec** 线程将从全局池中挑选这些消息并处理它们。

3.3.7 共享内存的通信部分 (UNIX™)

如果至少将一个连接配置为 IPC 共享内存连接，那么数据库服务器将为共享内存的 IPC 通信部分分配内存。数据库服务器将在您设置共享内存时执行此分配。通信部分包含本地客户机应用程序的消息缓冲区，这些应用程序使用共享内存与数据库服务器进行通信。

共享内存通信部分的大小大约等于 12 KB 乘以共享内存通信所需的期望连接数 (**nettype** ipcshm)。如果 **nettype** ipcshm 不存在，那么期望的连接数将缺省为 50。有关客户机如何连接到共享内存的通信部分的信息，请参阅客户机如何连接到通信部分 (UNIX)。

3.3.8 共享内存的虚拟扩展部分

共享内存的虚拟扩展部分包含附加的虚拟段和虚拟扩展段。

虚拟扩展段包含在用户定义的虚拟处理器中运行的用户定义的例程。

EXTSHMADD 配置参数可设置虚拟扩展段的大小。SHMADD 和 SHMTOTAL 配置参数适用于共享内存的虚拟扩展部分，正如它们也适用于共享内存的其他部分一样。

3.3.9 并行控制

运行在同一个虚拟处理器上和运行在个别虚拟处理器上的数据库服务器线程将共享对共享内存中资源的访问权。当线程写入共享内存时，它将使用称为*互斥*和*锁定*的机制来防止其

他线程同时写入相同的区域。互斥将给予线程访问共享内存资源的权限。防止其他线程写入缓冲区直到放置锁的线程完成对缓冲区的操作并且释放锁定为止的锁定。

共享内存互斥

数据库服务器在尝试修改共享内存中的数据时使用*互斥*来协调线程。每个可修改的共享内存资源都与互斥相关联。在线程可以修改共享内存资源之前，它必须先获取与该资源关联的互斥。在线程获取该互斥后，它便可以修改该资源。当修改完成时，线程将释放互斥。

如果线程试图获取互斥并发现该互斥正由另一个线程所持有，那么进入的线程必须等待互斥被释放。

例如，两个线程可以尝试访问块表中的同一个插槽，但只有一个线程可以获取与块表关联的互斥。只有持互斥的线程才可以将其条目写入块表。第二个线程必须等待该互斥被释放，然后才能获取该互斥。

有关监视互斥（也称为锁存器）的信息，请参阅监视共享内存概要文件和锁存器。

共享内存缓冲区锁定

共享内存的主要好处在于数据库服务器线程能够共享对存储在共享内存缓冲池中的磁盘页的访问权。数据库服务器将在通过有关锁定数据缓冲区的策略获得此增长的并行性时保持线程隔离。

缓冲区锁定的类型

数据库服务器使用两种类型的锁定来管理到共享内存缓冲区的访问权：

- 共享锁定
- 互斥锁定

这些锁定类型中的每种类型都将在执行期间强制实施所需级别的线程隔离。

共享锁定

如果多个线程可以访问缓冲区以读取数据但没有一个想要修改数据，那么缓冲区为共享方式或有一个共享锁定。

互斥锁定

如果线程要求对缓冲区进行互斥访问，那么缓冲区为互斥方式或具有互斥锁定。所有访问缓冲区的其他线程请求都放置在等待队列中。当正在执行的线程准备好释放互斥锁定，它将唤醒等待队列中的下一个线程。

3.3.10 数据库服务器线程对共享缓冲区的访问

数据库服务器线程通过一系列队列访问共享缓冲区，并且使用互斥和锁定来同步访问以及保护数据。

FIFO/LRU 队列

缓冲区为高速缓存而容纳数据。数据库服务器使用最近最少使用 (LRU) 队列来替换高速缓存的数据。GBase 8s 还有一个先进先出 (FIFO) 队列。设置 LRU 队列数时，您实际上正在设置 FIFO/LRU 队列数。

使用 BUFFERPOOL 配置参数可指定有关缓冲池的信息，包括要在数据库服务器共享内存已设置时要创建的 LRU 队列数目的信息，还包括控制将共享内存缓冲区清空到磁盘的频率的 **lru_min_dirty** 和 **lru_max_dirty** 的值信息。

要提高事务吞吐量，请增加 **lru_min_dirty** 和 **lru_max_dirty** 值。然而，请勿更改 **lru_min_dirty** 和 **lru_max_dirty** 值之间的差额。

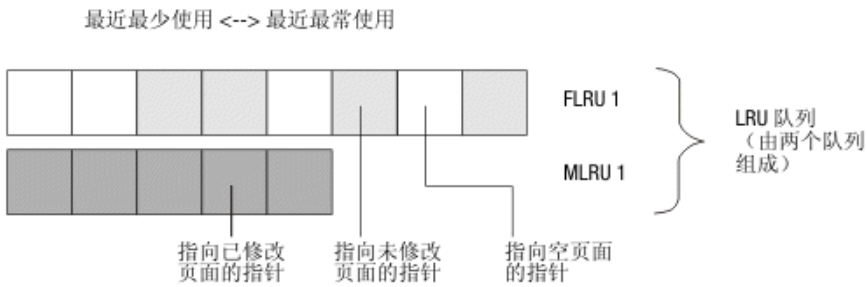
LRU 队列的组成部分

每个 LRU 队列由一对已链接的列表组成，如下所示：

- FLRU（可用的最近最少使用的）列表，它跟踪队列中可用的或未修改的页
- MLRU（已修改的最近最少使用的）列表，它跟踪队列中已修改的页

可用或未修改的页列表称为队列对的 FLRU 队列，而已修改的页列表称为 MLRU 队列。这两个不同的列表使您无需在队列中搜索可用或未修改的页。下图说明了 LRU 队列的结构。

图: LRU 队列



按最近最少使用的顺序排序的页

当数据库服务器处理对从磁盘读取页的请求时，它必须决定在内存中替换哪个页。 数据库服务器不是随机选择一个页，而是假定最近引用的页比一些时间没有引用的页更有可能在今后得到引用。因此，数据库服务器不会替换最近访问的页，而是替换最近最少访问的页。通过维护按最近最少使用到最近最多使用顺序排序的页，数据库服务器可以方便地在内存中定位最近最少使用的页。

LRU 队列和缓冲池管理

在处理开始之前，所有页缓冲区都是空的，并且每个缓冲区都由某个 FLRU 队列的条目代表。这些缓冲区在 FLRU 队列中是平均分布的。要计算每个队列中的缓冲区的数目，请将缓冲区总数除以 LRU 队列数。缓冲区和 LRU 队列的数目是在 BUFFERPOOL 配置参数中指定的。

当需要用户线程来获取缓冲区时，数据库服务器将随机选择一个 FLRU 队列，并使用列表中最旧的或最近最少使用的条目。如果可以锁存最近最少使用的页，该页将从队列中除去。

如果 FLRU 队列已锁定并且不能锁存结束页，那么数据库服务器将随机选择另一个 FLRU 队列。

如果用户线程正在共享内存中搜索特定的页，那么将从存储在缓冲区表中的控制信息中获取该页的 LRU 队列的位置。

正在执行的线程在完成其工作后将释放缓冲区。如果该页已修改，那么缓冲区将放置在 MLRU 队列的最近最多使用的一端。如果已读取该页但未修改，那么缓冲区将返回到 FLRU 队列的最近最多使用的一端。有关如何监视 LRU 队列的信息，请参阅监视缓冲池活动。

要配置的 LRU 队列数

配置多个 LRU 队列有两个用途：

- 它们将减少用户线程对队列的争用。
- 它们允许多个清除程序清空来自 LRU 队列的页并使脏页的百分比维持在可接受的水平。

根据计算机上提供的 CPU 数目推荐 LRUS 的初始值。如果您的计算机是单处理器的，那么一开始请将 BUFFERPOOL 配置参数中的 **lrus** 值设置为 4。如果您的计算机是多处理器的，请使用以下公式：

```
LRUS = max(4, (number_CPU_VPs))
```

在为 BUFFERPOOL 配置参数中的 **lrus** 提供了初始值后，请使用 gstat -R 监视 LRU 队列。如果发现脏 LRU 队列的百分比一直超过 **lru_max_dirty** 的指定值，请增加为 **lrus** 指定的值以添加更多的 LRU 队列。

例如，假设将 **lru_max_dirty** 设置为 70，并且一直发现有 75% 的 LRU 队列是脏的。请考虑增加 lrus 的值。如果增加 LRU 队列的数目，就会缩短各个队列的长度，从而减少页清除程序的工作。然而，您必须使用 CLEANERS 配置参数分配足够数量的页清除程序（如下一节中所说明）。保留 **lru_max_dirty** 和 **lru_min_dirty** 之间相同的差额。

要分配的清除程序数

通常必须为应用程序经常更新的每个磁盘都配置一个清除程序。但是，除此之外还必须考虑 LRU 队列的长度以及检查点的频率，如以下各段中所说明。

除了 LRU 队列数不够之外，另外一个影响页清除程序数是否跟得上清除的页数的因素是您是否分配了足够的页清除程序线程数。在一些队列 中，脏页的百分比可能超过指定给 **lru_max_dirty** 的BUFFERPOOL 值，因为没有清除程序可用于清除这些队列。一段时间过后，页清除程序可能实在赶不上了，这时缓冲池将会比您在 **lru_max_dirty** 中指定的百分比还要脏。

例如，假设 CLEANERS 参数已设置为 8，以及将 LRU 队列数从 8 增加到 12。您不能指望性能会有多大的提升，因为 8 个清除程序现在必须共享清除额外的 4 个队列的工作。

如果将 CLEANERS 的数值增加到 12，那么单个清除程序就可以更加有效地清除现在已缩短的队列中的每个队列。

将 CLEANERS 设置得太小会导致出现检查点时性能变差，因为页清除程序必须在检查点期间将所有已修改的页清空到磁盘。如果不配置足够数量的页清除程序，那么检查点会花费更长的时间，导致整体的性能变差。

有关更多信息，请参阅清空缓冲池缓冲区。

已添加到 MLRU 队列的页数

页清除程序线程会定期将 MLRU 队列中已修改的缓冲区清空到磁盘。要指定清除开始的点，请使用 BUFFERPOOL 配置参数指定 **lru_max_dirty** 的值。

通过指定页清除程序何时开始，**lru_max_dirty** 值会限制可附加到 MLRU 队列的页缓冲区数。**lru_max_dirty** 的初始设置是 60.00，因此页清除会在队列所管理的缓冲区的百分之六十被修改时开始。

实际上，页清除程序可以在若干条件下开始，但只有其中一个条件是 MLRU 队列达到 **lru_max_dirty** 值的情况。有关数据库服务器如何执行缓冲池清空的更多信息，请参阅将数据清空到磁盘。

以下示例显示 **lru_max_dirty** 的值如何应用于 LRU 队列以指定页清除程序何时开始，从而限制 MLRU 队列中的缓冲区数。

```
Buffers specified as 8000
lrus specified as 8
lru_max_dirty specified as 60 percent

Page cleaning begins when the number of buffers in the MLRU
    queue is equal to lru_max_dirty.

Buffers per lru queue = (8000/8) = 1000

Max buffers in MLRU queue and point at which page cleaning
    begins: 1000 x 0.60 = 600
```

MLRU 清除结束

还可以指定 MLRU 清除可以结束的点。BUFFERPOOL 配置参数中的 **lru_min_dirty** 值指定 MLRU 队列中缓冲区的可接受百分比。例如，如果将 **lru_min_dirty** 设置为 50.00，那么当修改了 LRU 队列中 50% 的缓冲区时就不需要清除页了。 实际上，页清除可以根据页清除程序线程的指令继续超过此点。

以下示例显示 **lru_min_dirty** 的值如何应用于 LRU 队列，以指定 MLRU 队列中缓冲区的可接受百分比以及页清除结束的点。

```
Buffers specified as 8000
lrus specified as 8
lru_min_dirty specified as 50 percent

The acceptable number of buffers in the MLRU queue and
    the point at which page cleaning can end is equal
    to lru_min_dirty.

Buffers per LRU queue = (8000/8) = 1000

Acceptable number of buffers in MLRU queue and the point
    at which page cleaning can end: 1000 x .50 = 500
```

可以对 **lru_max_dirty** 和 **lru_min_dirty** 值使用十进制。例如，如果把 **lru_max_dirty** 设置为 1.0333 并且把 **lru_min_dirty** 设置为 1.0，那么将触发 LRU 在 3,100 脏缓冲区时开始写并在 3,000 脏缓冲区时停止。

有关数据库服务器如何清空缓冲池的更多信息，请参阅将数据清空到磁盘。

预读操作

对于顺序表或索引扫描，可以将数据库服务器配置为在处理当前页的过程中预读多页。数据库服务器将自动预读正在为查询处理的当前页的后面几页，除非禁用自动预读操作。预读使应用程序能够更快地运行，因为它们将花费更少的时间等待磁盘 I/O。

自动*预读取*在连续扫描数据记录期间请求将页面放入缓冲池高速缓存，这可在服务器检测到查询（包括 OLTP 查询和索引扫描）遇到 I/O 时提高查询的性能。

缺省情况下，数据库服务器将根据查询遇到来自磁盘的 I/O 的时间，自动确定何时发出预读请求以及何时停止。

- 如果查询遇到 I/O，服务器将发出预读请求，以提高查询性能。因为预读请求通过对相对于 CPU 处理速度较慢的 I/O 处理进行补偿，可以极大地提高数据库处理的速度，所以使性能得到提升。
- 如果查询大部分已进行高速缓存，那么服务器将检测到没有执行任何 I/O，因此不会预读。

使用 AUTO_READAHEAD 配置参数可更改查询的自动预读方式或禁用查询的自动预读。您可以：

- 通过运行 gadmin -wm 或 gadmin -wf 命令，动态更改 AUTO_READAHEAD 配置参数的值。
- 运行 SET ENVIRONMENT AUTO_READAHEAD 语句以更改方式，或启用或禁用会话的自动预读。

顺序数据或索引读取期间，只要数据库服务器检测到有必要执行预读，就会执行预读。

onconfig 文件中的 RA_PAGES 参数指定数据库服务器执行预读时，要从磁盘或索引读取的页数。

RA_THRESHOLD 参数指定内存中导致数据库服务器执行另一个预读的未处理页的数量。例如，如果 RA_PAGES 设置为 10，且 RA_THRESHOLD 为 4，那么数据库服务器将在缓冲区中留有 4 个要处理的页时预读 10 个页。

可以使用 gstat -p 命令来查看数据库服务器读取和写入，并监视需要某个线程等待共享内存锁存器的次数。RA-pgsused 输出字段显示数据库服务器预读使用的页数，并监视数据库服务器对预读的使用。

使用 gstat -g rah 命令可显示有关预读请求的统计信息。

数据库服务器线程对缓存页的访问

数据库服务器使用共享锁定缓存以使多个数据库服务器线程能够在共享内存中同时访问相同的缓冲区。

数据库服务器使用两种类型的缓冲区锁来提供此并发性，而不会在线程隔离时有所损失。锁定访问的两个类型是共享和互斥。（有关更多信息，请参阅缓冲区锁定的类型。）

3.3.11 将数据清空到磁盘

将缓冲区写入磁盘称为*缓冲区清空*。当用户线程修改缓冲区中的数据时，它会将缓冲区标为*脏*。当数据库服务器将缓冲区清空到磁盘时，它随后会将缓冲区标为*不脏*并允许覆盖缓冲区中的数据。

数据库服务器将清空以下缓冲区：

- 缓冲池（包含在本节中）
- 物理日志缓冲区
  请参阅清空物理日志缓冲区。
- 逻辑日志缓冲区
  请参阅清空逻辑日志缓冲区。

页清除程序线程缓冲区清空。数据库服务器总是运行至少一个页清除程序线程。如果数据库服务器配置了多个页清除程序线程，那么 LRU 队列将分配到这些页清除程序中以便更有效的清空。有关指定数据库服务器运行多少页清除程序线程的信息，请参阅《GBase 8s 管理员参考》中的 CLEANERS 配置参数。

清空物理日志缓冲区、已修改的共享内存页缓冲区和逻辑日志缓冲区必须依照设计用以保持数据一致性的特定规则与页清除程序活动同步。

清空缓冲池缓冲区

缓冲区的清空由以下任何一种条件启动：

- MLRU 队列中缓冲区数达到 BUFFERPOOL 配置参数中 **lru_max_dirty** 值指定的数目。
- 页清除程序线程无法跟上。换句话说，用户线程必定需要获取缓冲区，但是没有未修改的缓冲区可用。
- 数据库服务器必须执行检查点。（请参阅检查点。）

自动 LRU 调节可影响所有缓冲池并调整 BUFFERPOOL 配置参数中的 **lru_min_dirty** 和 **lru_max_dirty** 值。

首先清空前映像

已修改页的前映像在已修改页本身之前清空到磁盘。

实际上，首先清空物理日志缓冲区然后再清空包含已修改的页的缓冲区。因此，即便由于用户线程正在试图获取缓冲区但没有缓冲区可用（前台写入），而必须对共享内存缓冲区页清空时，也只有在页的前映像已写入磁盘后才能将缓冲区页清空。

清空物理日志缓冲区

数据库服务器临时存储物理日志缓冲区中存储一些已修改的磁盘页的前映像。如果前映像已写入物理日志缓冲区但未写入磁盘上的物理日志，那么服务器会在将已修改的页清空到磁盘之前先将物理日志缓冲区清空到磁盘。

数据库服务器总是在将任何数据缓冲区清空到磁盘之前首先将物理日志缓冲区的内容清空到磁盘。

以下事件将导致活动的物理日志缓冲区清空：

- 活动物理日志缓冲区变满。
- 共享内存中的已修改页必须清空，但前映像仍然在活动物理日志缓冲区中。
- 出现检查点。

数据库服务器一次只使用两个物理日志缓冲区中的一个。此缓冲区是活动（或当前）物理日志缓冲区。在数据库服务器将当前物理日志缓冲区清空到磁盘之前，它使其他缓冲区称为当前物理日志缓冲区，以便该服务器可以在清空第一个缓冲区时继续写入。

物理日志缓冲区和物理日志都有助于保持数据在物理上和逻辑上的一致性。有关物理记录、检查点和快速恢复的信息，请参阅物理日志记录、检查点和快速恢复。

同步缓冲区清空

当首先启动共享内存时，所有的缓冲区都是空的。当处理发生时，数据页将从磁盘读入缓冲区，而用户线程则开始修改这些页。

描述清空活动

为了向您提供有关提示缓冲区清空活动的特定条件的信息，数据库服务器定义了三种类型的写入并计算发生每次写入的频率：

- 前台写入
- LRU 写入
- 块写入

要显示数据库服务器保留的写入计数，请如《GBase 8s 管理员参考》中所述使用 gstat -F。

如果实现数据库服务器的镜像，那么数据总是先写入主块。然后写入会在镜像块上重复。对镜像块的写入次数包含在计数中。有关监视数据库服务器所执行的写入类型的更多信息，请参阅监视缓冲池活动。

前台写入

只要 **sqlexec** 线程将缓冲区写入磁盘，都称为*前台写入*。当 **sqlexec** 线程代表用户在 LRU 队列中搜索但无法找到空的或未修改的缓冲区时将发生前台写入。为了腾出空间，**sqlexec** 线程将一次清空一个页以容纳要从磁盘读取的数据。（有关更多信息，请参阅 FIFO/LRU 队列。）

如果 **sqlexec** 线程必须执行缓冲区清空以获取共享内存缓冲区，那么性能会变差。必须避免前台写入。要显示对前台写入次数的计数，请运行 gstat -F。如果发现前台写入定期发生，请调整页清除参数的值。 要么增加页清除程序的数目，要么减少 BUFFERPOOL **lru_max_dirty** 值。

LRU 写入

与前台写入不同，LRU 写入由页清除程序执行而不是由 **sqlexec** 线程执行。数据库服务器将 LRU 写入作为后台写入执行，后者通常在脏缓冲区的百分比超过在 BUFFERPOOL 配置参数中为 **lru_max_dirty** 指定的百分比时发生。

此外，前台写入可触发 LRU 写入。当发生前台写入时，已执行写入的 **sqlexec** 线程将提醒页清除程序唤醒并清除该线程已为之执行前台写入的 LRU。

在已正确调整的系统中，页清除程序将确保有足够的未修改的缓冲区页可用于存储从磁盘读取的页。因此，执行查询的 **sqlexec** 线程无需在读入查询所需的磁盘页之前，先将页清空到磁盘。对于不利用前台写入的查询而言，这种情况可以使性能有显著地提高。

由于页清除程序线程执行缓冲区写入比 **sqlexec** 线程更有效率，因此 LRU 写入比前台写入更受欢迎。要监视写入的两种类型，请使用 gstat -F。

块写入

*块写入*统称由页清除程序线程在检查点期间执行，或在可能的情况下在共享内存缓冲池中的每个页已修改时执行。块写入（作为已排序的写入执行）是数据库服务器可用的最有效的写入。

在块写入期间，每个页清除程序线程都指定到一个或多个块。每个页清除程序线程在缓冲区头中读取并将一组指针创建到与其特定的块关联的页。（页清除程序对此信息有访问权，因为块编号包含在物理页编号地址中，它是页头的一部分。）此排序将最小化磁盘上的磁头移动（磁盘搜索时间）并在可能的情况下使页清除程序线程能够在写入时使用大缓冲区。

此外，由于用户线程必须等待检查点完成，因此页清除器线程不会与大量线程争用 CPU 时间。结果，页清除程序线程可以通过更少的上下文切换来完成它们的工作。

清空逻辑日志缓冲区

数据库服务器使用共享内存逻辑日志缓冲区作为临时存储器用于存储向数据库服务器页描述修改的记录。这些更改记录将从逻辑日志缓冲区写入磁盘上当前的逻辑日志文件，并最终写入逻辑日志备份介质。有关逻辑日志记录的描述，请参阅逻辑日志。

五个事件导致当前逻辑日志缓冲区清空：

- 当前的逻辑日志缓冲区变满。
- 事务已在带有未缓冲日志记录的数据库中准备或落实。
- 非日志记录数据库会话终止。
- 出现检查点。
- 已修改不需要物理日志中的前映像的页。

以下主题将详细讨论其中每个事件。

事务已在带有未缓冲日志记录的数据库中准备或终止之后

以下日志记录将导致逻辑日志缓冲区在带有未缓冲日志记录的数据库中清空：

- COMMIT
- PREPARE
- XPREPARE
- ENDTRANS

有关已缓冲日志记录和未缓冲日志记录的比较，请参阅 《GBase 8s SQL 指南：语法》 中的 SET LOG 语句。

使用非日志记录数据库或未缓冲日志记录的会话终止时

甚至对于非日志记录数据库，数据库服务器会记录某些改变数据库方式的活动，如创建表或扩展数据块。当数据库服务器终止那些使用未缓冲日志记录或非日志记录数据库的会话时，会清空逻辑日志缓冲区以确保已记录任何日志记录活动。

当出现检查点时

有关在检查点期间所发生事件的详细描述，请参阅检查点。

当已修改不需要物理日志文件中的前映像的页时

当修改了不需要物理日志中的前映像的页时，必须在页清空到磁盘前清空逻辑日志缓冲区。

3.3.12 缓冲区大对象数据

简单大对象（TEXT 或 BYTE 数据）可以存储在数据库空间中也可以存储在 BLOB 空间中。智能大对象（CLOB 或 BLOB 数据）仅存储在智能大对象空间中。数据库服务器使用不同方法来访问每种类型的存储空间。以下主题将描述用于每种类型的缓冲方法。

写入简单大对象

数据库服务器以写入任何其他数据类型时使用的相同方法将简单大对象写入数据库空间中的磁盘页。有关更多信息，请参阅将数据清空到磁盘。

您也可以将简单大对象指定给 BLOB 空间。数据库服务器将简单大对象写入 BLOB 空间（方法与将其他数据写入共享内存缓冲区时的不同），然后将其清空到磁盘。有关 BLOB 空间的描述，请参阅《GBase 8s 管理员参考》 中有关磁盘结构和存储器的章节。

BLOB 页和共享内存

BLOB 空间 BLOB 页存储大量数据。因此，数据库服务器不会经由共享内存缓冲池创建或访问 BLOB 页，并且它不会将 BLOB 空间的 BLOB 页写入逻辑或物理日志。

如果 BLOB 空间数据已通过共享内存池，它可能会通过去除索引页和数据页来降低池的有效性。反之，BLOB 页将在创建时直接写入磁盘。

要减少逻辑日志和物理日志的流量，数据库服务器会使用与写入数据库空间页时不同的方法将 BLOB 页从磁介质写入数据库空间备份磁带以及逻辑日志备份磁带。

由于光学介质的高可靠性，存储在光学介质上的 BLOB 页将不会写入到数据库空间和逻辑日志备份磁带中。

创建简单大对象

当将简单大对象数据写入磁盘时，它所属的行可能尚不存在。 例如，在插入期间，简单大对象会在其他行数据传送之前传送。存储简单大对象后，数据行将与指向其位置的 56 位描述符一同创建。有关如何以物理方式存储简单大对象的描述，请参阅《GBase 8s 管理员参考 》的磁盘存储器和结构章节中有关数据库空间 BLOB 页结构的部分。

创建 BLOB 页缓冲区

要从应用程序进程接收简单大对象数据，数据库服务器会创建一对 BLOB 空间缓冲区（一个用于读一个用于写），大小都是一个 BLOB 空间的 BLOB 页。 确保用户只有一组 BLOB 空间缓冲区，并因此一次只能访问一个简单大对象。

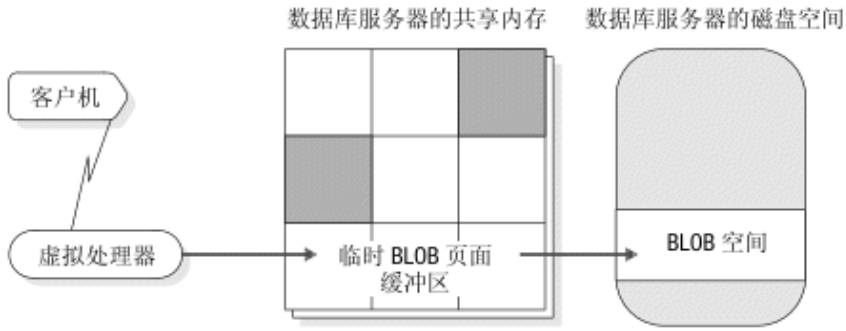
简单大对象数据将以 1 KB 段从客户机应用程序进程传输到数据库服务器。然后数据库服务器开始用 1 KB 块填充 BLOB 空间缓冲区，并尝试一次缓冲两个 BLOB 页。数据库服务器将缓冲两个 BLOB 页以便可以确定何时将转发指针从一个页添加到下一个。当它填充了第一个缓冲区并发现有更多的数据要传送时，它会在向磁盘写入页之前将转发指针添加到下一个页。当不再有数据要传送时，数据库服务器会不带转发指针将最后一页写入到磁盘。

当线程开始将第一个 BLOB 空间缓冲区写入到磁盘时，它尝试根据用户定义的 BLOB 页大小执行 I/O。例如，如果 BLOB 页的大小是 32 KB，那么数据库服务器将尝试以 32,768

字节的增量来读取或写入数据。如果底层硬件（如磁盘控制器）无法在单个操作中传送这些数据量，那么操作系统内核将在内部（以内核方式）循环直到传送完成为止。

BLOB 空间缓冲区会保留到创建它们的缓冲区完成为止。当将简单大对象写入到磁盘时，数据库服务器将取消分配 BLOB 空间缓冲区对。下图说明了将简单大对象写入 BLOB 空间的过程。

图: 将简单大对象写入 BLOB 空间



用自由图页分配和跟踪 BLOB 空间的 BLOB 页。按需创建连接 BLOB 页的链接和指向下一个 BLOB 页段的指针。

操作记录（插入、更新或删除）将写入到逻辑日志缓冲区。

访问智能大对象

数据库服务器通过共享内存缓冲区访问智能大对象，其方法与访问存储在数据库空间中的数据相同。然而，智能大对象的用户数据部分将在比正常缓冲区页更低的优先级上缓冲以防止将更高值的数据清空出缓冲池。缓冲允许对经常访问的智能大对象进行更快的访问。

智能大对象存储在智能大对象空间中。您不能将简单大对象存储在智能大对象空间中，也不能将智能大对象存储在 BLOB 空间中。智能大对象空间由用户数据区域和元数据区域组成。用户数据区域包含智能大对象数据。元数据区域包含有关智能大对象空间内容的信息。

因为智能大对象已通过共享内存缓冲池并且可以记录，所以您必须在分配缓冲区时考虑这些智能大对象。使用 BUFFERPOOL 配置参数分配共享内存缓冲区。通常，尝试分配足够的缓冲区，以便为每个同时打开的智能大对象包含两个智能大对象页。（附加页可用于预读的用途。）

使用 LOGBUFF 配置参数可指定逻辑日志缓冲区的大小。有关设置以下每个配置参数的信息，请参阅《GBase 8s 管理员参考》：

- BUFFERPOOL
- LOGBUFF

已记录的智能大对象的用户数据区域不会通过物理日志，因此无需为智能大对象更改 PHYSBUFF 参数。

有关智能大对象空间结构的更多信息，请参阅《GBase 8s 管理员参考》的磁盘结构和存储器章节中有关智能大对象空间结构的部分。有关创建智能大对象空间的信息，请参阅《GBase 8s 管理员参考》中有关 gspaces 实用程序的信息。

3.3.13 64 位平台上的内存使用

通过 64 位寻址，您可以有更大的缓冲池用于减少从磁盘获取数据的 I/O 操作量。因为 64 位平台允许更大的内存地址空间，所以以下与内存相关的配置参数的最大值在 64 位平台上更大：

- BUFFERPOOL
- CLEANERS
- DS_MAX_QUERIES
- DS_TOTAL_MEMORY
- LOCKS
- LRUS
- SHMADD
- SHMVIRTSIZE

每个 64 位平台的机器说明列出了以上配置参数和特定于平台的参数（例如 SHMMAX）的最大值。

3.4 管理共享内存

这些主题说明如何执行涉及管理共享内存的以下任务：

- 设置共享内存配置参数
- 设置共享内存
- 为数据库服务器共享内存的常驻部分打开或关闭驻留
- 将段添加到共享内存的虚拟部分
- 为关键活动预留内存
- 在具有内存限制的应用程序中保留目标内存量
- 监视共享内存

这些主题未涵盖 DS_TOTAL_MEMORY 配置参数。此参数将给决策支持查询的内存分配设置上限。

3.4.1 设置操作系统共享内存配置参数

一些操作系统配置参数可以影响数据库服务器使用共享内存。由于参数名会随平台的改变而改变，而且并不是所有的参数都存在于所有平台上，因此不提供参数名。以下列表按功能描述了这些参数：

- 最大操作系统共享内存段大小（用字节或 KB 表示）
- 最小共享内存段大小（用字节表示）
- 最大共享内存标识数

- 共享内存的下边界地址
- 每个进程的最大连接共享内存段数
- 整个系统中最大共享内存量

仅限 UNIX：

- 最大信号标识数
- 最大信号数
- 每个标识的最大信号数

在 UNIX™ 上，机器说明文件包含您可用于配置操作系统资源的推荐值。可以在配置操作系统时使用这些推荐值。有关如何设置这些操作系统参数的信息，请查询操作系统手册。

最大共享内存段大小

当数据库服务器创建所需的共享内存段时，它将尝试获取尽可能大的操作系统段。数据库服务器尝试获取的第一个段大小等于其正在分配的部分（常驻、虚拟或通信）的大小，并四舍五入到 8 KB 的最近倍数。

如果请求的段大小超过允许的最大大小，那么数据库服务器会从操作系统接收到错误。 如果数据库服务器接收到错误，那么它会将请求的大小除以二，然后再试一次。继续尝试获取，直到可以创建 8 KB 倍数的最大段大小为止。然后，数据库服务器将创建与它所需要的一样多的附加段。

最大共享内存标识数 (UNIX™)

共享内存标识会在虚拟处理器尝试连接共享内存时影响数据库服务器的运行。操作系统使用共享内存标识来标识共享内存段。对于大多数操作系统来说，虚拟处理器根据先来先服务的原则接收标识，直到基本上达到为操作系统所定义的限制。有关共享内存标识的更多信息，请参阅虚拟处理器如何连接到共享内存。

您或许能够通过将共享内存标识数乘以最大共享内存段大小来计算操作系统可以分配的最大共享内存量。

信号 (UNIX)

数据库服务器需要一个 UNIX™ 信号用于每个虚拟处理器，一个用于每个通过共享内存（**ipcshm** 协议）连接到数据库服务器的用户，六个用于数据库服务器实用程序，以及十六个用于其他用途。

3.4.2 设置数据库服务器共享内存配置参数

可以修改影响共享内存的常驻或虚拟部分的配置参数。

可以使用文本编辑器来修改共享内存配置参数。有关这些配置参数的列表，请参阅《GBase 8s 管理员参考》中 onconfig 门户网站：按功能类别排列的配置参数的内容。

在 UNIX™ 上，您必须是 **root** 或 **Gbasedbt** 用户才能使用其中任何一种。

设置常驻共享内存的参数

以下列表中包含 onconfig 文件中的参数，这些参数指定了缓冲池和内部表在共享内存的常驻部分中的配置。在对配置参数所做的任何更改生效前，您必须关闭并重新启动数据库服务器。有关配置参数的描述，请参阅《GBase 8s 管理员参考》。

BUFFERPOOL
指定缓冲池的信息，该缓冲池必须给定义给数据库空间使用的每个不同页大小。
LOCKS
指定数据库对象（例如，行、键值、页和表）的初始锁数。
LOGBUFF
指定逻辑日志缓冲区的大小。
PHYSBUFF
指定物理日志缓冲区的大小。
RESIDENT
指定数据库服务器共享内存的常驻部分的驻留。
SERVERNUM
指定本地主机上数据库服务器的唯一标识号。
SHMTOTAL
指定将由数据库服务器使用的总内存量。

设置虚拟共享内存的参数

有多个配置参数影响共享内存的虚拟部分。

以下列表包含用于配置共享内存的虚拟部分的配置参数：

DS_HASHSIZE
数据分发高速缓存中列表的散列存储区数。
DS_POOLSIZE
数据分发高速缓存中的最大条目数。
PC_HASHSIZE
为 UDR 高速缓存和数据库服务器使用的其他高速缓存指定散列存储区数。
PC_POOLSIZE
指定可以存储在 UDR 高速缓存中的 UDR 的数目（SPL 例程和外部例程）。此外，此参数指定了其他数据库服务器高速缓存的大小，如类型名高速缓存和 opclass 高速缓存。
SHMADD
指定动态添加的共享内存段的大小。
SHMNOACCESS
指定不用于连接共享内存的虚拟内存地址范围的列表。使用该参数可避免与其他进程发生冲突。
EXTSHMADD
指定用户定义的例程在用户定义的虚拟处理器中运行时，添加的虚拟扩展段的大小。
SHMTOTAL
指定将由数据库服务器使用的总内存量。
SHMVIRTSIZE
指定共享内存的虚拟部分的初始大小。
STACKSIZE
指定数据库服务器用户线程的堆栈大小。

设置共享内存性能的参数

可以修改用于指定共享内存信息的配置参数。

以下配置参数影响共享内存性能。

AUTO_READAHEAD
为查询指定自动预读方式或禁用自动预读操作。自动预读操作通过在数据库服务器检测到查询遇到 I/O 时发出异步页请求，以帮助提高查询性能。 异步页请求通过将查询处理与从磁盘检索数据并将数据放入缓冲池所需的处理相叠加，从而可以提高查询性能。

CKPTINTVL
如果需要检查点并且 RTO_SERVER_RESTART 配置参数没有设置为打开自动检查点调整，请指定在数据库服务器检查该检查点之前可以耗用的最大秒数。

CLEANERS
指定数据库服务器要运行的页清除程序线程数。

RA_PAGES
指定数据库服务器在执行数据或索引记录的顺序扫描时尝试预读的磁盘页数。
指定数据库服务器在顺序扫描数据或索引记录期间尝试预读的磁盘页数。如果启用了 AUTO_READAHEAD 配置参数，服务器将忽略 RA_PAGES 配置参数中指定的信息。

RA_THRESHOLD
指定未处理的内存页素，这些内存页在读取后导致数据库服务器在磁盘上预读。

使用文本编辑器设置共享内存参数

可以使用文本编辑器设置有关常驻和虚拟共享内存以及共享内存性能的配置参数。 在 onconfig 文件中找到该参数，输入一个或多个新的值，然后重新将文件写入磁盘。更改生效之前，您必须关闭并重新启动数据库服务器。

3.4.3 设置 SQL 语句高速缓存参数

下表显示了可以配置 SQL 语句高速缓存的不同方法。

表 1. 配置 SQL 语句高速缓存

配置参数	用途	gadmin 命令
STMT_CACHE	打开、启用或禁用内存中的 SQL 语句高速缓存。如果已打开，请指定 SQL 语句高速缓存能否保存已解析和优化的 SQL 语句。	gadmin -e ***mode***
STMT_CACHE_HITS	指定将语句完全插入到 SQL 语句高速缓存之前，命中（引用）该语句的次数。	gadmin -W STMT_CACHE_HITS
STMT_CACHE_NOLIMIT	控制是否在 SQL 语句高速缓存的大小大于 STMT_CACHE_SIZE 值之后将语句插入到该高速缓存中。	gadmin -W STMT_CACHE_NOLIMIT
STMT_CACHE_NUMPOOL	定义 SQL 语句高速缓存的内存池数。	无

配置参数	用途	gadmin 命令
STMT_CACHE_SIZE	指定 SQL 语句高速缓存的大小。	无

使用以下 gstat 选项可监视 SQL 语句高速缓存：

- gstat -g ssc
- gstat -g ssc all
- gstat -g ssc pool

有关这些配置参数、gstat -g 选项以及 gadmin 命令的更多信息，请参阅《GBase 8s 管理员参考》。

有关限定和恒等语句的详细信息，请参阅《GBase 8s SQL 指南：语法》。

3.4.4 设置共享内存

要设置共享内存，使数据库服务器脱机然后联机。有关如何使数据库服务器从联机方式转到脱机方式的信息，请参阅从任何方式立即更改到脱机方式。

3.4.5 打开或关闭常驻共享内存的驻留

可以使用以下两种方法中的任何一个为共享内存的常驻部分打开或关闭驻留

- 使用 gadmin 实用程序可在数据库服务器处于联机方式时立刻逆转共享内存的状态。
- 更改 onconfig 文件中的 RESIDENT 参数,可在下次设置数据库服务器共享内存时打开或关闭共享内存驻留。

有关共享内存常驻部分的描述，请参阅共享内存的常驻部分。

在联机方式下打开或关闭驻留

要在数据库服务器处于联机方式时打开或关闭驻留，请使用 gadmin 实用程序。

要立刻为共享内存的常驻部分打开驻留，请运行以下命令：% gadmin -r

要立刻为共享内存的常驻部分关闭驻留，请运行以下命令：% gadmin -n

这些命令不会更改 onconfig 文件中 RESIDENT 参数的值。也就是说，这种更改不是永久的，驻留会在您下一次设置共享内存时回复到 RESIDENT 参数所指定的状态。在 UNIX™ 上，您必须是 **root** 或 **gbasedbt**用户才能打开或关闭驻留。

重新启动数据库服务器时打开或关闭驻留

可以使用文本编辑器打开或关闭驻留。 要更改驻留的当前状态，请使用文本编辑器定位 RESIDENT 参数。将 RESIDENT 设置为 1 以打开驻留或设置为 0 以关闭常驻，然后重新将文件写入磁盘。更改生效之前，您必须关闭并重新启动数据库服务器。

3.4.6 将段添加到共享内存的虚拟部分

可以使用 gadmin 实用程序的 **-a** 选项将指定大小的段添加到虚拟共享内存。

正常情况下，无需将段添加到虚拟共享内存，因为数据库服务器会按需自动添加段。

如果限制了操作系统段的数目并且初始段大小与所需的数量相比太小，以至于快要超过操作系统对共享内存段的限制，这时使用 gadmin 实用程序添加段这一选项将是有用的。

3.4.7 为关键活动保留内存

保留特定量的内存，以供在需要执行关键活动（例如，回滚活动）且数据库服务器可用内存有限时使用。这样可以在执行关键活动期间服务器耗尽可用内存时，防止数据库服务器崩溃。

如果通过将 LOW_MEMORY_RESERVE 配置参数设置为指定值（以千字节为单位）来启用新的 LOW_MEMORY_RESERVE 配置参数，即使用户收到内存不足的错误，也可完成回滚之类的关键活动。如果 LOW_MEMORY_RESERVE 的值为 0，将关闭低内存保留功能。

例如，512 千字节即为合理的保留内存量。 要保留 512 千字节，请指定：

```
LOW_MEMORY_RESERVE 512
```

也可使用 gadmin -wm 或 gadmin -wf 命令来动态调整 LOW_MEMORY_RESERVE 配置参数的值。

使用 gstat -g seg 命令可监视 LOW_MEMORY_RESERVE 的值。查找输出的最后两行，其中包含短语“low memory reserve”。 这两个输出行中的第一行显示保留的内存大小（以字节为单位）。而第二行显示数据库服务器已使用此内存的次数和所需最大内存量。 重新启动服务器时，这两个值都将重置。

3.4.8 配置内存严重过低时的服务器响应

可配置内存严重过低时服务器为继续处理而采取的操作，而不是返回内存不足的错误。根据空闲时间、内存使用量和其他因素指定终止会话的条件，这样目标应用程序可继续处理。配置低内存响应对于存在内存限制的嵌入式应用程序很有用。

要设置自动低内存管理，请执行以下操作：

- 将 LOW_MEMORY_MGR 配置参数设置为 1，从而在数据库服务器启动时启用低内存管理。
- 通过使用带 scheduler lmm enable 自变量的 SQL 管理 API 命令来为要维护的内存量设置阈值参数。

要禁用自动低内存管理，请运行带 scheduler lmm disable 自变量的 SQL 管理 API 命令。

目标内存量的维护方案

本主题中的方案显示在具有内存限制的应用程序中可如何维护目标内存量。

假设您希望指定数据库服务器在可用内存小于或等于 10 MB 时开始运行低内存管理进程，此类进程可停止应用程序并释放内存。假设您还希望指定服务器在可用内存大于或等于 20 MB 时停止运行低内存管理进程：

1. 将 LOW_MEMORY_MGR 配置参数设置为 1 并重新启动服务器，或者运行 gadmin -wf 命令以更改 LOW_MEMORY_MGR 配置参数的值。
2. 运行带 scheduler lmm enable 自变量和低内存参数的 SQL 管理 API 命令，如下所示：

```
EXECUTE FUNCTION task("scheduler lmm enable",
        "LMM START THRESHOLD", "10MB",
        "LMM STOP THRESHOLD", "20MB",
        "LMM IDLE TIME", "300");
```

3. 运行 gstat -g lmm 命令以显示有关自动低内存管理设置的信息，包括服务器尝试维护的内存量、服务器当前使用的内存量、低内存启动和停止阈值，以及与内存有关的其他统计信息。

   也可在 online.log 文件中查看低内存管理信息。

3.4.9 监视共享内存

以下主题描述如何监视共享内存段、共享内存概要文件，以及如何使用特定的共享内存资源（缓冲区、锁存器和锁定）。

可以使用 gstat -o 实用程序捕获数据库服务器共享内存的静态快照用于以后分析和比较。

监视共享内存段

监视共享内存段可确定数据库服务器创建的段的数目和大小。数据库服务器将自动分配共享内存段，因此这些数字会更改。如果数据库服务器正在分配的共享内存段过多，那么您可以增加 SHMVIRTSIZE 配置参数。有关更多信息，请参阅 *GBase 8s 管理员参考* 中有关配置参数的主题。

gstat -g seg 命令列出了每个共享内存段的信息，包括段的地址和大小，以及可用或正在使用的内存量。有关 gstat -g seg 输出的示例，请参阅 *GBase 8s 管理员参考* 中有关 gstat 实用程序的信息。

监视共享内存概要文件和锁存器

监视数据库服务器概要文件以分析性能和共享内存资源的用法。

可以获取有关锁存器使用的统计信息以及有关特定锁存器的信息。这些统计信息可用作衡量系统活动的标准。

ON-Monitor 概要文件屏幕保留了有关共享内存使用情况的累积统计信息。

要将这些统计信息复位到零，请使用 gstat -z 选项。有关 gstat 显示的所有字段的描述，请参阅《GBase 8s 管理员参考》中有关 gstat 实用程序的信息。

用于监视共享内存和锁存器的命令行实用程序

您可使用以下命令行实用程序来监视共享内存和锁存器。

gstat -s
: 使用 gstat -s 命令可获取锁存器信息。

gstat -p
: 运行 gstat -p 以显示有关数据库服务器活动以及等待锁存器的统计信息（在 **lchwaits** 字段中）。有关 gstat -p 输出的示例，请参阅《GBase 8s 管理员参考》 中有关 gstat 实用程序的信息。

使用 ON-Monitor 监视共享内存概要文件和锁存器 (UNIX™)

选择状态 > 概要文件。该屏幕显示了共享内存统计信息以及当前的运行方式、引导时间、当前时间以及锁存器。

SMI 表

查询 **sysprofile** 表以获取共享内存统计信息。此表包含 gstat -p 输出中可用的所有统计信息，除 **ovbuff**、**usercpu** 和 **syscpu** 统计信息以外。

监视缓冲区

可以同时获取有关缓冲区使用的统计信息以及有关特定缓冲区的信息。统计信息包括已高速缓存到缓冲区的数据写入数的百分比，以及线程需要等待获取缓冲区的次数。已高速缓存的写入数百分比是重要的性能指标。

缓冲区等待数提供了系统并行性的测量方法。

有关特定缓冲区的信息包括线程所包含的共享内存中所有缓冲区的列表。可以使用这些信息来跟踪特定缓冲区的状态。例如，您可以确定另一个线程是否正在等待缓冲区。

用于监视缓冲区的命令行实用程序

可以使用以下命令行实用程序监视缓冲区：

- gstat -p 实用程序
- gstat -B 实用程序
- gstat -b 实用程序
- gstat -X 实用程序
- gstat -R 实用程序

gstat -p 实用程序

运行 gstat -p 可获取有关高速缓存读取数和写入数的统计信息。以下高速缓存统计信息在输出显示的项行中的四个字段内显示：

- 从共享内存缓冲区读取的次数 (**bufreads**)
- 已高速缓存的读取数百分比 (**%cached**)
- 写入共享内存的次数 (**bufwrits**)
- 已高速缓存的写入数百分比 (**%cached**)

- 有关通用页的信息（缓冲池中的非标准页）

在输出中，如果发生的读取数或写入数超过 232（取决于平台），那么读取数或写入数可能为负数。

gstat -p 选项还显示了指示会话需要等待缓冲区的次数的统计信息 (bufwaits)。

有关 gstat -p 输出的示例，请参阅《GBase 8s 管理员参考 》中有关 gstat 实用程序的信息。

gstat -B 实用程序

运行 gstat -B 可获取有关不在空闲列表中的所有缓冲区的信息，其中包括：

- 缓冲区的共享内存地址
- 当前持有缓冲区的线程的地址
- 正在等待每个缓冲区的第一个线程的地址
- 有关缓冲池的信息

有关 gstat -B 输出的示例，请参阅《GBase 8s 管理员参考》中有关 gstat 实用程序的信息。

gstat -b 实用程序

运行 gstat -b 可获取以下有关每个缓冲区的信息：

- 当前线程所持有的每个缓冲区的地址
- 缓冲区中所容纳的页的页号
- 缓冲区中所容纳的页类型（例如，数据页、表空间页等等）
- 放置在缓冲区上的锁定的类型（互斥或共享）
- 当前正持有缓冲区的线程的地址
- 正在等待每个缓冲区的第一个线程的地址
- 有关缓冲池的信息

您可以将用户线程的地址与 gstat -u 显示中显示的地址相比较以获取会话标识号。

有关 gstat 显示的字段的更多信息，请参阅《*GBase 8s 管理员参考*》中的 gstat 实用程序的信息。

gstat -X 实用程序

运行 gstat -X 获取的信息与 gstat -b 同样，此外还可以获取有关所有正在等待缓冲区的线程的完整列表，而不单单是第一个等待的线程。

gstat -R 实用程序

使用 gstat -R 显示有关缓冲池的信息，其中包括有关缓冲区的信息。

使用 ON-Monitor 监视缓冲区 (UNIX™)

可以使用 ON-Monitor 来获取有关高速缓存读操作数和写操作数的统计信息。

要访问 gstat -p（**bufreads**、**%cached** 和 **bufwrits %cached**）的 gstat -p 实用程序主题中提到的字段，请选择状态 > 概要文件选项。

以下是 ON-Monitor 状态菜单的概要文件选项中已高速缓存的读取数和写入数的统计信息的示例：

Disk Reads	Buff. Reads	%Cached	Disk Writes	Buff. Writes	%Cached
177	330	46.36	4	0	0.00

...

SMI 表

查询 **sysprofile** 表可获取有关已高速缓存的读取数和写入数以及总缓冲区等待数的统计信息。以下是相关行。

dskreads
从磁盘读取的次数

bufreads
从缓冲区读取的次数

dskwrites
写入磁盘的次数

bufwrites
写入缓冲区的次数

buffwts
任何线程必须等待缓冲区的次数

监视缓冲池活动

可以获得与缓冲区可用性相关的统计信息以及有关每个 LRU 队列中缓冲区的信息。

统计信息包括数据库服务器尝试超过最大缓冲区数和磁盘写入数的次数（按导致缓冲区清空的事件分类）。这些统计信息可帮助您确定缓冲区数是否适当。每个 LRU 队列中缓冲区的信息包括队列的长度和该队列中已修改的缓冲区的百分比。

用于获取有关缓冲池活动的信息的命令行实用程序

可以使用 gstat 实用程序获取有关缓冲池活动的信息。还可以运行 Server Administrator 中的 gstat 选项。

有关 gstat 选项的更多信息，请参阅《GBase 8s 管理员参考》 中有关 gstat 实用程序的信息。

gstat -p 实用程序

gstat -p 输出包含指出数据库服务器尝试超过由 BUFFERPOOL 配置参数中的 buffers 值指定的最大共享缓冲区数的统计信息 (**ovbuff**)。

gstat -F 实用程序

运行 gstat-F 可获取有关已执行写入的计数（按写入类型）。（有关不同写入类型的说明，请参阅描述清空活动。）

gstat-F 命令显示了以下所有写入类型：

- 前台写入
- LRU 写入
- 块写入

gstat-F 命令还列出了以下有关页清除程序的信息：

- 页清除程序编号
- 页清除程序的共享内存地址
- 页清除程序的当前®状态
- 指定了页清除程序的 LRU 队列

有关 gstat -F 输出的示例，请参阅《GBase 8s 管理员参考》中有关 gstat 实用程序的信息。

gstat -R 实用程序

运行 gstat -R 可获得有关每个 LRU 队列中的缓冲区数以及已修改或可用的缓冲区数以及百分比的信息。

有关 gstat -R 输出的示例，请参阅《GBase 8s 管理员参考》中有关 gstat实用程序的信息。

SMI 表

查询 **sysprofile** 表可获得有关下列行中所容纳的写入类型的统计信息。

fgwrites
前台写入数

lruwrites
LRU 写入数

chunkwrites
块写入数

3.4.10 服务器故障后删除共享内存段

数据库服务器发生故障之后，必须关闭共享内存段。

重要： 此过程必须由具有 GBase 8s 使用经验的 DBA 来执行。请咨询技术支持以获取协助。此过程仅适用于 UNIX™ 系统。

如果 GBase 8s 数据库服务器实例发生故障，请遵循以下过程来删除共享内存段：

1. 以用户 **gbasedbt** 的身份登录。
2. 使用 gadmin -k 命令使数据库服务器进入脱机方式，然后除去共享内存。
3. 如果 gadmin -k 命令失败并且服务器未脱机，请运行 onclean -k 命令，或执行以下步骤：
   a. 使用 gstat -g glo 命令显示多线程运行信息。
   b. 在上面命令的输出中，找到与 class 列中 **cpu** 的第一个实例关联的进程标识 (pid)。

   例如，在 gstat -g glo 命令的以下输出中的 class 列内，**cpu** 出现了四次，其 pid 分别为 2599、2603、2604 和 2605：

```
MT global info:
sessions  threads  vps  lngspins
  0         49       14       1
          sched calls     thread switches   yield 0     yield n     yield forever
```

```
total:      900100          898846          1238          27763       423778
per sec:    327             325             2             12          151
Virtual processor summary:
 class      vps         usercpu     syscpu     total
 cpu         4           0.92        0.10       1.02
 aio         4           0.02        0.02       0.04
 lio         1           0.00        0.00       0.00
 pio         1           0.00        0.00       0.00
 adm         1           0.00        0.01       0.01
 msc         1           0.00        0.00       0.00
 fifo        2           0.00        0.00       0.00
 total          14           0.94        0.13       1.07
Individual virtual processors:
 vp      pid         class       usercpu     syscpu     total
 1       2599        cpu         0.25        0.06       0.31
 2       2602        adm         0.00        0.01       0.01
 3       2603        cpu         0.23        0.00       0.23
 4       2604        cpu         0.21        0.03       0.24
 5       2605        cpu         0.23        0.01       0.24
 6       2606        lio         0.00        0.00       0.00
 7       2607        pio         0.00        0.00       0.00
 8       2608        aio         0.02        0.02       0.04
 9       2609        msc         0.00        0.00       0.00
 10      2610        fifo        0.00        0.00       0.00
 11      2611        fifo        0.00        0.00       0.00
 12      2612        aio         0.00        0.00       0.00
 13      2613        aio         0.00        0.00       0.00
 14      2614        aio         0.00        0.00       0.00
                     tot         0.94        0.13       1.07
```

c. 使用 kill 命令（按顺序）终止进程标识 2599、2603、2604 和 2605。

4. 如果尚未除去共享段，请遵循以下步骤：

a. 确定服务器编号。

可以通过检查 GBase 8s 实例的 onconfig 文件来找到服务器编号。

b. 将服务器编号与 21078 相加。

例如，如果服务器编号为 1，请将 1 与 21078 相加，从而得出 21079。

c. 将上一步中的和转换为十六进制。

在上一个示例中，21079 的十六进制数为 5257。

d. 在上一步中生成的十六进制值末尾添上 48。

例如，525748。

e. 以 root 用户身份运行 ipcs 实用程序，以显示服务器保持打开的共享内存段（如果有）。搜索 key 列以查找 4.d 中得出的数字。

f. 除去与 4.d 中得出的数字关联的每个共享内存标识。

有关 onclean 实用程序的更多信息，请参阅《GBase 8s 管理员参考》。

有关您系统的正确 **ipcm** 语法，请查阅操作系统文档。

3.5 数据存储

这些主题定义了术语并且对执行管理磁盘空间中描述的任务所必须理解的概念进行了说明。这些主题涵盖以下领域：

- 有关数据库服务器用来存储磁盘上数据的物理和逻辑单元的定义
- 有关如何计算存储数据所需的磁盘空间量的指示信息
- 有关如何安排磁盘空间以及哪里放置数据库和表的准则
- 有关使用外部表的指示信息

请参阅最新的 GBase 8s 发行说明以获取有关与这些主题中说明的与存储单元相关的最大值的补充信息。

3.5.1 物理存储单元和逻辑存储单元

数据库服务器使用物理存储单元分配磁盘空间。与大小会产生变动的逻辑存储单元不同，每个物理单元的大小都是固定或指定的，它们的大小由磁盘体系结构所确定。数据库服务器使用以下物理单元管理磁盘空间：

- 块
- 页面
- 扩展数据块
- BLOB 页
- 智能大对象页

数据库服务器在以下逻辑单元中存储数据：

- 数据库空间
- 临时数据库空间
- BLOB 空间
- 智能大对象空间
- 临时智能大对象空间
- 外部空间
- 数据库
- 表

- 表空间
- 分区

数据库服务器保留了下列存储结构以确保数据的物理和逻辑一致性：

- 逻辑日志
- 物理日志
- 保留页

以下主题描述了数据库服务器所支持的各种数据存储单元以及这些单元间的关系。有关保留页的信息，请参阅《GBase 8s 管理员参考》中有关磁盘结构和存储的主题。

3.5.2 块

*块*是专用于数据库服务器数据存储的物理磁盘最大单元。

块可以为管理员提供用于分配磁盘空间的特别大的单元。单个块的最大大小是 4 TB。允许的块数为 32,766。

以下存储空间由块组成：

- 数据库空间
- BLOB 空间
- 智能大对象空间
- 临时数据库空间
- 临时智能大对象空间

创建块时，请指定其路径、大小和关联的存储空间名称。

数据库服务器还将块用于镜像。对块制作镜像时，数据库服务器将在块上维护该数据的两个副本。每次执行写入主块的操作后，都会自动向镜像块执行相同的写操作。读操作在两个块之间平均分布。如果主块或镜像块出现故障，发生故障的块会标记为关闭，另一个块会执行所有操作，而不会中断用户对数据的访问。

创建表、索引和其他数据库对象时，会将块空间分配或指定给这些对象。分配的空间并不一定会使用。例如，创建表时，为其分配了空间，但只有在向该表添加数据时才会使用该空间。当数据库空间中的所有块都报告可用页数为 0 时，无法在该数据库空间中创建新的数据库对象。但是，只要现有数据库对象具有未使用的空间，您就可以继续向这些数据库对象添加数据。可以通过使用 gstat -d 命令或 OpenAdmin Tool (OAT) 来监视块。

块的磁盘分配

数据库服务器可以使用常规操作系统文件或*原始磁盘设备*存储数据。在 UNIX™ 上，只要性能是重要因素时，就必须使用原始磁盘设备存储数据。

GBase 8s 存储空间可位于使用常规操作系统文件的 NFS 安装的文件系统上。

UNIX 上的未缓冲或已缓冲磁盘的访问

可以使用两种方法分配磁盘空间。既可以使用通过操作系统缓冲的文件，也可以使用未缓冲的磁盘存取。

通过操作系统缓冲的文件通常称为*熟*文件。

未缓冲的磁盘存取也称为*原始*磁盘空间。

当数据库空间位于*原始磁盘设备*（也称为*字符专用设备*）上时，数据库服务器使用未缓冲的磁盘存取。

要创建原始设备，用原始界面配置*块设备*（硬盘）。该设备提供的存储空间称为*原始磁盘空间*。原始磁盘空间的块在物理上是连续的。

块的名称是 /dev 目录中字符专用文件的名称。在许多操作系统中，您可以通过文件名的第一个字母（通常是 r）将字符专用文件与块专用文件区别开来。例如，/dev/rsd0f 是与 /dev/sd0f 块专用设备相对应的字符专用设备。

有关更多信息，请参阅在 UNIX 上分配原始磁盘空间。

*熟文件*是操作系统管理的常规文件。熟文件块和原始磁盘块是同等可靠的。与原始磁盘空间不同，熟文件的逻辑连续块可能在物理上是不连续的。

您可以比原始磁盘空间更方便地分配熟文件。要分配熟文件，必须在任何现有的分区上创建该文件。块的名称是该文件的完整路径名。这些步骤在在 UNIX 上分配熟文件空间中描述。

在对性能要求不是很高的学习环境中，或对于静态数据而言，熟文件会很方便。如果您必须使用 UNIX™ 熟文件，请在那些文件中存储最不频繁存取的数据。将文件存储在最少活动的文件系统中。

对于熟文件块，操作系统将处理来自其自身的缓冲池的所有块 *I/O*，并确保所有对块的写入都在物理上写入磁盘。

重要： 虽然通常必须在 UNIX 上使用原始磁盘设备来获得更好的性能，但如果启用了 DIRECT_IO 配置参数，那么熟文件的性能可接近用于数据库空间块的原始设备的性能。此情况的发生是因为直接 I/O 会绕过文件系统缓冲区的使用。如果有 AIX® 操作系统，那么还可以为 GBase 8s 启用并发 I/O，以便在对使用熟文件的块执行读写时使用直接 IO。

要针对性能确定最佳设备，请对具有用于数据库空间和表布局的两种设备类型的系统执行基准测试。

使用原始磁盘时，无需采取任何特殊的操作来创建大于 2 GB 的块和文件。如果要在熟文件中创建大块，或者如果要将各种数据库导出和导入实用程序用于大文件，那么必须确保正确配置将保存大文件的文件系统。

可扩展块

*可扩展块*是 GBase 8s 可自动扩展的块，或当应用程序需要额外存储空间时可手动扩展的块。如果具有可扩展块，那么无需添加新块或耗费时间尝试确定哪种存储空间将耗尽空间，以及何时将耗尽空间。

通过配置 GBase 8s 自动添加更多存储空间，可以防止以下情况下可能发生的错误：分区需要更多存储空间，但是在分区所在空间内的任何一个块中都找不到该空间。

可扩展块必须位于非镜像数据库空间或临时数据库空间中。

可使用带 modify space sp_sizes 自变量的 SQL 管理 API 命令来修改可扩展块所在空间的扩展大小和创建大小。

偏移量

系统管理员可以将物理磁盘分成多个分区，这些分区是一个磁盘的不同部分，并且具有不同的路径名。虽然在原始磁盘设备上分配块时必须使用整个磁盘分区，但可以使用偏移量将分区或熟文件进一步分成更小的块。有关更多信息，请参阅磁盘布局准则。

提示： 有了对块大小的 4 TB 的限制，您可以通过为每个磁盘驱动器指定单个块来避免对磁盘进行分区。

可以使用偏移量来指示磁盘分区、文件或设备上给定块的位置。例如，假设创建了一个 1000 KB 的块，您希望将其分成两个块，每个块 500 KB。可以用偏移量 0 KB 来标记第一个块的开始，然后用偏移量 500 KB 来标记第二个块的开始。

您可以在创建、添加或从数据库空间、BLOB 空间或智能大对象空间删除块时指定偏移量。

还可能需要指定偏移量来阻止数据库服务器覆盖分区信息。在 UNIX 上分配原始磁盘空间说明了应在何时以及如何指定偏移量。

3.5.3 页

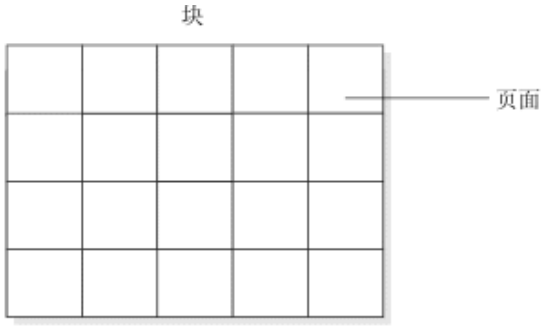
*页*是数据库服务器用于在 GBase 8s 数据库中读取和写入的物理磁盘存储单元。下图说明了页的概念，在磁盘片中用加深部分表示。

图: 磁盘上的页

在多数 UNIX™ 平台上，页大小是 2 KB。因为硬件会确定页的大小，所以您不能更改此值。

块包含一定数量的页，如下图所示。页总是完全包含在块中；也就是说，页不能穿过块边界。

图: 块，在逻辑上分成一系列页



有关数据库服务器如何在页中构造数据的信息，请参阅《GBase 8s 管理员参考》中有关磁盘结构和存储的章节

3.5.4 BLOB 页

*BLOB 页*是数据库服务器用于在 BLOB 空间中存储简单大对象（TEXT 或 BYTE 数据）的磁盘空间分配单元。有关 BLOB 页的描述，请参阅 BLOB 空间。

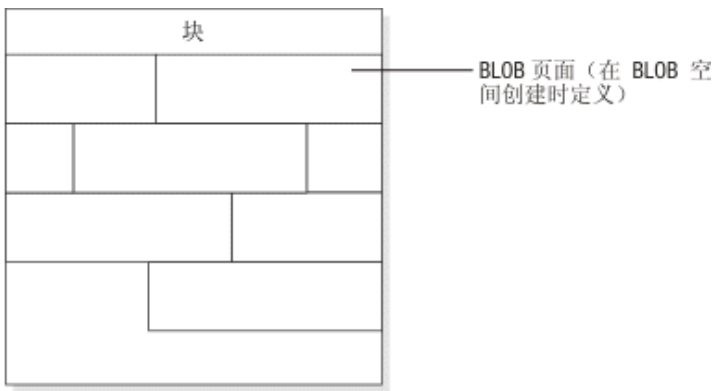
可将 BLOB 页的大小指定为数据库服务器页大小的倍数。 由于数据库服务器将 BLOB 页分配为连续的空间，所以在与数据大小尽可能接近的 BLOB 页中存储简单大对象将更加有效。下图说明了 BLOB 页的概念，以数据页的倍数（三倍）表示。

图: 磁盘上的 BLOB 页

有关 GBase 8s 如何构造存储在 BLOB 页中的数据的信息，请参阅《GBase 8s 管理员参考》的磁盘结构和存储主题中有关 BLOB 空间 BLOB 页的结构的部分。

就像块中的页一样，一定数量的 BLOB 页可在 BLOB 空间中组成块，如下图所示。BLOB 页总是完全包含在一个块中而不能穿过块的边界。

图: BLOB 空间中的块，在逻辑上分为一系列 BLOB 页



您可以选择在数据库空间中存储简单大对象数据，而不要将其存储在 BLOB 空间中。然而，对于大于两个页的简单大对象，如果在 BLOB 页中存储简单大对象，那么性能将提高。存

储在数据库空间中的简单大对象可以共享一个页，但是存储在 BLOB 空间中的简单大对象不可以共享页。

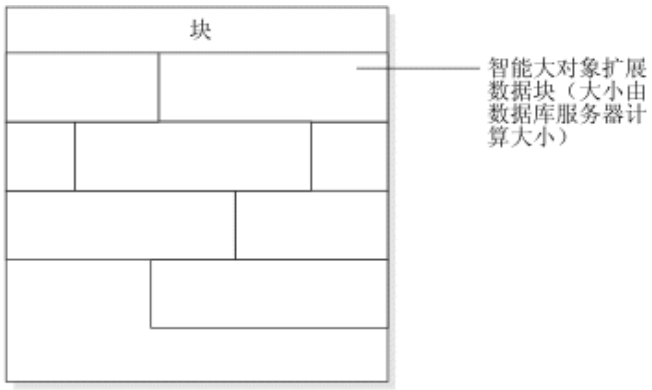
有关如何确定 BLOB 页大小的信息，请参阅确定 BLOB 页大小。

3.5.5 智能大对象页

*智能大对象页*是数据库服务器用于在智能大对象空间内存储智能大对象的页类型。有关智能大对象空间的描述，请参阅智能大对象空间。与 BLOB 页不同，智能大对象页是不可配置的。智能大对象页的大小与数据库服务器页的相同，在 UNIX™ 上通常是 2 KB。

智能大对象空间中的分配单元是扩展数据块，而 BLOB 空间中的分配单元是 BLOB 页。就像块中的页一样，一定数量的智能大对象扩展数据块可在智能大对象空间中组成块，如下图所示。扩展数据块总是完全包含在一个块中而不能穿过块的边界。

图: 智能大对象空间中的块，在逻辑上分成一系列扩展数据块



智能大对象无法存储在数据库空间或 BLOB 空间。有关更多信息，请参阅智能大对象空间以及《GBase 8s 管理员参考》的磁盘结构和存储章节中有关智能大对象空间结构的部分。

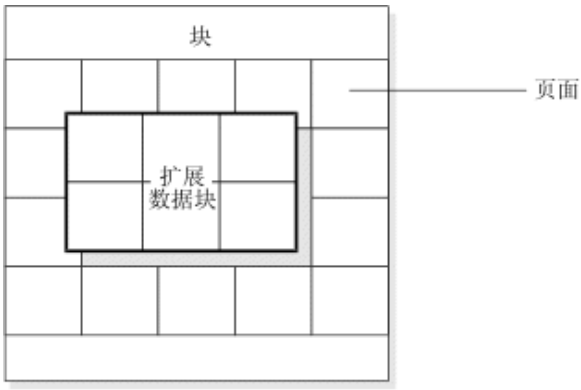
数据库服务器根据一组启发式搜索（如写操作中的字节数）计算智能大对象的数据块大小。

3.5.6 扩展数据块数

当您创建表时，数据库服务器会分配固定数量的空间以包含要存储在该表中的数据。当此空间填满时，数据库服务器必须分配额外的存储空间。数据库服务器用来同时分配初始和后续存储空间的物理存储单元称为*扩展数据块*。

下图说明了扩展数据块的概念。

图: 由原始磁盘设备上 6 个连续页构成的扩展数据块



扩展数据块包含了为指定的表存储数据的邻接页的集合。（请参阅表。）每个永久数据库表都有两个与其关联的扩展数据块大小。*初始扩展数据块*大小是在表第一次创建时分配给该表的 KB 数。*下一个扩展数据块*大小是在初始扩展数据块（以及任何后续的扩展数据块）变满时分配给该表的 KB 数。对于永久表以及用户定义的临时表，下一个扩展数据块大小会在每个扩展数据块之后开始加倍。对于系统创建的临时表，下一个扩展数据块大小会在已添加了 4 个扩展数据块之后开始加倍。

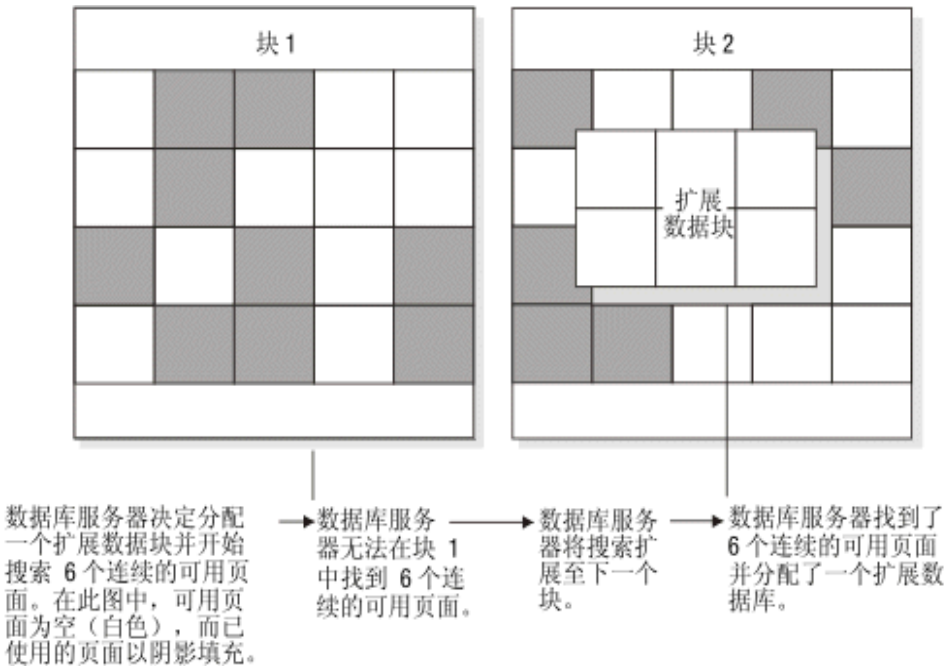
在创建表时，您可以指定初始扩展数据块的大小，以及表增长时要添加的扩展数据块的大小。 还可以修改数据库空间的表中扩展数据块的大小，以及修改新的后续扩展数据块的大小。要指定初始扩展数据块大小和下一个扩展数据块大小，请使用 CREATE TABLE 和 ALTER TABLE 语句。有关更多信息，请参阅《GBase 8s SQL 指南：语法》和 《GBase 8s 管理员参考》中有关磁盘结构的部分。

当创建带有 CLOB 或 BLOB 数据类型列的表时，还应为智能大对象空间定义扩展数据块。有关更多信息，请参阅智能大对象空间的存储特征。

下图显示数据库服务器如何为扩展数据块分配 6 个页：

- 扩展数据块总是完全包含在一个块中；扩展数据块不能穿过块的边界。
- 如果数据库服务器找不到指定给下一个扩展数据块大小的连续磁盘空间，那么它将在数据库空间的下一个块中搜索连续的空间。

图: 扩展数据块的分配过程



3.5.7 数据库空间

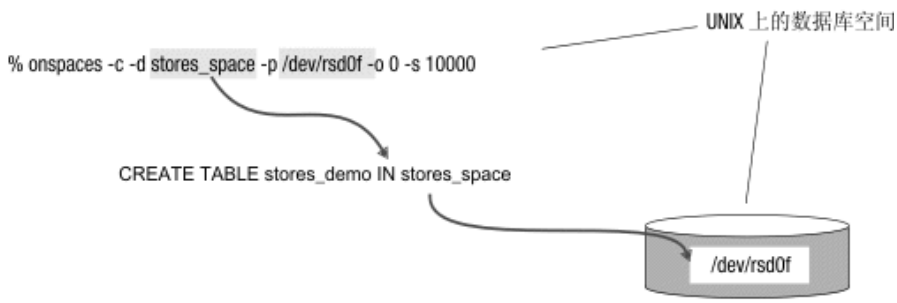
*数据库空间*是一种可包含 1 到 32766 个块的逻辑单元。放置数据库、表、逻辑日志文件以及数据库空间中的物理日志。

控制简单大对象数据的存储位置

数据库服务器管理员的关键职责是控制数据库服务器存储数据的位置。通过在最快的磁盘驱动器上存储访问频率很高的表或关键数据库空间（根数据库空间、物理日志和逻辑日志），可以提高性能。通过将关键数据存储在单独的物理设备上，您可以确保当包含非关键数据的磁盘中有一个发生故障时，该故障只会影响该磁盘上的数据可用性。

如下图所示，要控制数据库或表的放置，可使用 CREAT DATABASE 或 CREAT TABLE 语句的 IN dbspace 选项。

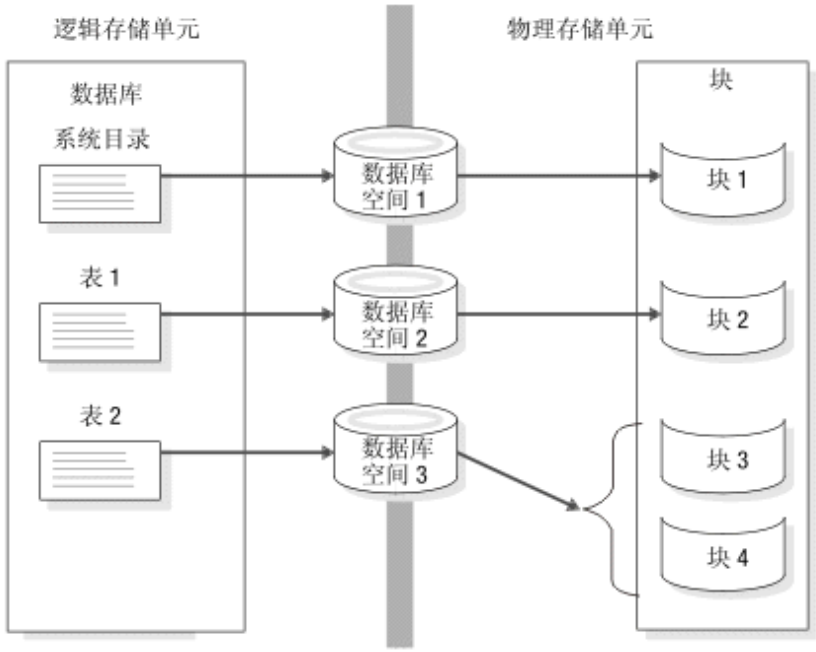
图：使用 CREATE TABLE... IN 语句控制表的放置



在数据库空间中创建数据库或表之前，您必须先创建数据库空间。

数据库空间可以包含一个或多个块，如下图所示。您可以在任何时候添加更多的块。监视数据库空间块的填充度以及预期是否有必要向数据库空间分配更多的块，是高优先级的数据库服务器管理员任务。当数据库空间包含多个块时，不能指定数据所在的块。

图: 链接逻辑和物理存储单元的数据库空间



数据库服务器使用数据库空间来存储数据库和表。

创建标准或临时数据库空间时，可指定该数据库空间的页大小。您不能指定BLOB 空间、智能大对象空间或外部空间的页大小。如果不指定页大小，那么根数据库空间的大小将为缺省页大小。有关更多信息，请参阅创建具有非缺省页大小的数据库空间。

创建标准数据库空间时，可在数据库空间中指定表空间 **tblspace** 的第一个和下一个扩展数据块大小。如果要在必须将表空间 **tblspace** 扩展数据块放入非主块中时减少表空间 **tblspace** 扩展数据块的数量并减少这些情况发生的频率，请执行此操作。 有关更多信息，请参阅为表空间 tblspace 指定第一个和下一个扩展数据块大小。

您可以为镜像数据库空间中的每个块建立镜像。一旦数据库服务器分配了镜像块，它会立刻将该镜像块中的所有空间都标记为已满。请参阅监视磁盘使用量。

有关使用 Server Administrator 或 gspaces 执行以下任务的信息，请参阅管理磁盘空间。

- 创建数据库空间
- 向数据库空间添加块
- 重命名数据库空间
- 删除块
- 删除数据库空间、BLOB 空间或智能大对象空间

根数据库空间

*根数据库空间*是数据库服务器创建的初始数据库空间。根数据库空间是特殊的，因为它包含了保留页和内部表，它们将描述和跟踪所有物理和逻辑存储单元。（有关这些主题的更

多信息，请参阅表以及《GBase 8s 管理员参考》中有关磁盘结构和存储的章节。）根数据库空间的初始块及其镜像是唯一在磁盘空间初始化期间创建的块。可在磁盘空间设置后将其他块添加到根数据库空间。

onconfig 配置文件中的以下磁盘配置参数引用根数据库空间的第一个（初始）块：

- ROOTPATH
- ROOTOFFSET
- ROOTNAME
- MIRRORPATH
- MIRROROFFSET
- TBLTBLFIRST
- TBLTBLNEXT

根数据库空间也是由 CREAT DATABASE 语句创建的任何数据库的缺省数据库空间位置。

根数据库空间是由数据库服务器为执行所请求的数据管理而创建的所有临时表的缺省位置。

请参阅根数据库空间的大小以获取有关应为根数据库空间分配多少空间的信息。还可以在设置数据库服务器磁盘空间后将额外的块添加到根数据库空间。

临时数据库空间

*临时数据库空间*是专门为临时表的存储而保留的数据库空间。您不能为临时数据库空间建立镜像。

数据库服务器不会删除临时数据库空间，除非明确指示应这么做。临时数据库空间是临时的指的只是数据库服务器在其不正常关闭时不保留任何数据库空间内容。

无论何时设置数据库服务器，所有临时数据库空间都将设置。数据库服务器会清除可能从数据库服务器上次关闭后留下的任何表。

数据库服务器不会为临时数据库空间执行逻辑或物理日志记录。由于没有用物理方式记录临时数据库空间，因此只有较少的检查点和 I/O 操作出现，以此提高了性能。

对于标准数据库空间中的临时表，数据库服务器会记录表的创建、扩展数据块的分配以及表的删除。相反，数据库服务器不会记录存储在临时数据库空间中的表。临时数据库空间中禁止使用逻辑日志可减少在逻辑恢复期间要前滚的日志记录数，从而提高关键停机时间内的性能。

使用临时数据库空间存储临时表还会减少存储空间备份的大小，因为数据库服务器不会备份临时数据库空间。

数据库服务器使用临时磁盘空间来存储备份期间被覆盖以及内存中发生查询处理而溢出的之前数据映像。请确保正确设置 DBSPACETEMP 环境变量或参数，以便指定的数据库空间具有足够空间，能满足您的需求。如果指定的数据库空间中空间不足，备份将失败，并且将使用根数据库空间，或者在填满根数据库空间之后，备份将失败。

如果您有多个临时数据库并且在临时表中执行了 SELECT 语句，那么查询结果将以循环顺序插入。

有关如何创建临时数据库空间的详细指示信息，请参阅创建临时数据库空间。

重要： 当数据库服务器作为辅助数据库服务器运行时，它需要临时数据库空间存储由只读查询生成的任何内部临时表。

3.5.8 BLOB 空间

*BLOB 空间*是由一个或多个只存储 TEXT 和 BYTE 数据的块组成的逻辑存储单元。BLOB 空间会以可能的最有效的方法存储 TEXT 和 BYTE 数据。可将与不同的表（请参阅表）关联的 TEXT 和 BYTE 列存储在相同的 BLOB 空间中。

数据库服务器将存储在 BLOB 空间中的数据直接写入磁盘。这些数据不会穿过常驻共享内存。如果穿过常驻共享内存，那么数据卷可能会占用大量缓冲池页，以至于使其他数据和索引页强制退出。由于同样的原因，数据库服务器不会将指定给 BLOB 空间的 TEXT 或 BYTE 对象写入逻辑日志或物理日志。当您备份逻辑日志时，数据库服务器会通过将 BLOB 空间对象从磁盘直接写入到逻辑日志备份带来记录这些 BLOB 空间对象。BLOB 空间对象不会穿过逻辑日志文件。

当您创建 BLOB 空间时，您可将其指定给一个或多个块。 您可以在任何时候添加更多的块。数据库服务器管理员的任务之一是监视块的填充度以及预期是否有必要向 BLOB 空间分配更多的块。有关如何监视块的填充度的指示信息，请参阅监视 BLOB 空间中的简单大对象。有关如何创建 BLOB 空间、向 BLOB 空间添加块或从 BLOB 空间删除块的指示信息，请参阅管理磁盘空间。

有关 BLOB 空间结构的信息，请参阅《GBase 8s 管理员参考》中有关磁盘结构和存储的主题。

3.5.9 智能大对象空间

智能大对象空间是由存储智能大对象的一个或多个块组成的逻辑存储单元。智能大对象由 CLOB（字符大对象）和 BLOB（二进制大对象）数据类型组成。用户定义的数据类型也可以使用智能大对象空间。有关数据类型的更多信息，请参阅《GBase 8s SQL 指南：参考》。

使用智能大对象空间的优势

与 BLOB 空间相比，智能大对象空间有下列优势：

- 类似于标准的 UNIX™ 文件，它们有读取、写入和搜索的属性。

  程序员可以使用类似于 UNIX 函数的函数来读取、写入和查找智能大对象。GBase 8s 在 GBase 8s ESQL/C 编程界面中提供该智能大对象接口。

- 它们是可以恢复的。

  您可以记录对存储在智能大对象空间中的数据进行的所有写入操作。 如果事务期间发生故障，可以落实或回滚更改。

- 它们服从事务隔离方式。

  可以锁定粒度级别不同的智能大对象，而锁定持续时间将遵循事务隔离级别的规则。

- 表行中的智能大对象无需在一个语句中检索。

  应用程序可以使用 GBase 8s ESQL/C 编程接口存储或检索以块计的智能大对象。

智能大对象空间和 Enterprise Replication

当您为 Enterprise Replication 定义复制服务器时，您必须创建智能大对象空间。Enterprise Replication 会将已复制的数据假脱机到智能大对象。在 CDR_QDATA_SBSPACE 配置参数中指定智能大对象空间名称。Enterprise Replication 使用缺省的日志方式，通过这种方式可创建智能大对象空间以用于假脱机行数据。CDR_QDATA_SBSPACE 配置参数接受多个智能大对象空间，最多可达 32 个智能大对象空间。Enterprise Replication 可以支持日志记录和非日志记录智能大对象空间的组合以便存储假脱机行数据。

在定义复制服务器时，可以使 Enterprise Replication 自动从存储池中配置磁盘空间，并设置相应的配置参数。如果 CDR_QDATA_SBSPACE 或 CDR_DBSPACE 配置参数未设置或设置为空，那么 cdr define server 命令将自动创建必需的磁盘空间，并将这些配置参数设置为相应值。

元数据、用户数据和保留区域

与 BLOB 空间和数据库空间一样，当您创建智能大对象空间时，可以为其指定一个或多个块。然而，智能大对象空间的第一个块始终有三个区域：

元数据区域
: 元数据标识了智能大对象空间的关键方面以及存储在智能大对象空间中的每个智能大对象，并且使数据库服务器能够操纵和恢复其中的智能大对象。

用户数据区域
: 用户数据是由用户应用程序存储在智能大对象空间中的智能大对象数据。块最多可有两个用户数据区域。

保留区域
: 当需要更多的空间时，数据库服务器会将保留区域中的空间分配给元数据或用户数据区域。 块最多可有两个保留区域。

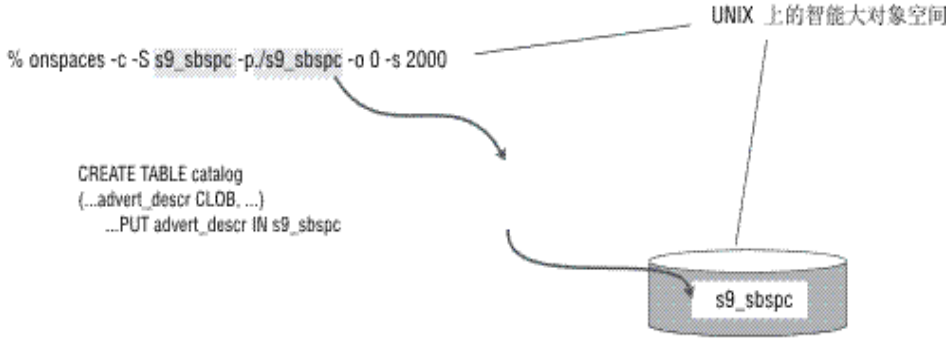
当您将块添加到智能大对象空间时，您可以指定块是否包含元数据区域和用户数据区域或者是否要专门为用户数据保留块。您可以在任何时候添加更多的块。如果您正在更新智能大对象，那么原始磁盘上对用户数据的 I/O 将比熟的块文件上的 I/O 快很多。有关如何创建智能大对象空间、向智能大对象空间添加块或从智能大对象空间删除块的指示信息，请参阅管理磁盘空间。

重要： 不管数据库的日志记录设置如何，总是记录智能大对象空间元数据。

控制智能大对象数据的存储位置

您可在创建表时指定列的数据类型。对于智能大对象，可以指定 CLOB、BLOB 或用户定义的数据类型。如下图所示，要控制智能大对象的放置，可以使用 CREAT TABLE 语句 PUT 子句中的 IN *sbspace* 选项。

图: 控制智能大对象的放置



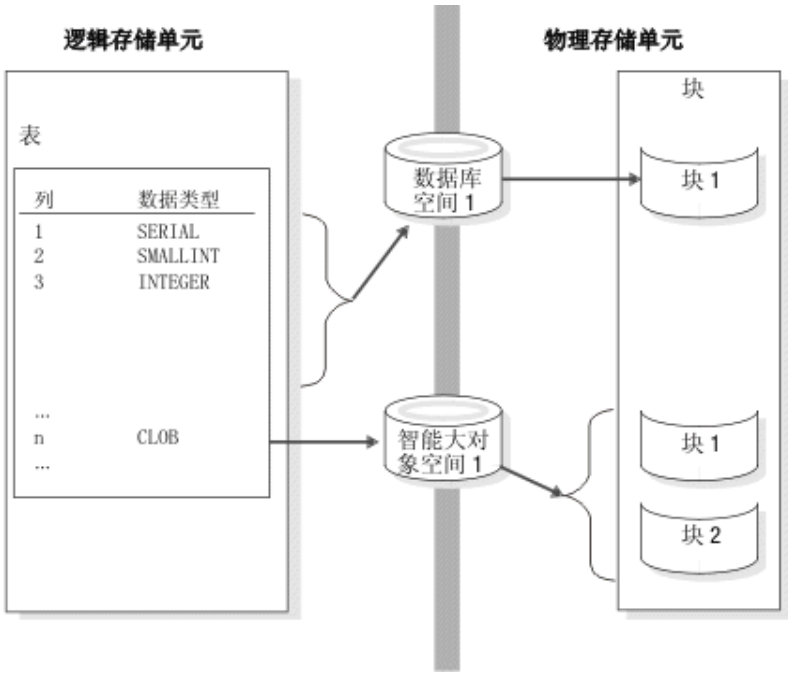
在 PUT 子句中指定智能大对象空间之前，必须首先创建智能大对象空间。有关如何使用 gspaces -c -S 命令创建智能大对象空间的更多信息，请参阅向数据库空间或 BLOB 空间添加块。有关如何在 PUT 子句中指定智能大对象特征的更多信息，请参阅《GBase 8s SQL 指南：语法》 中的 CREAT TABLE 语句。

如果未指定 PUT 子句，那么数据库服务器将在 SBSPACENAME 配置参数中指定的缺省智能大对象空间中存储智能大对象。有关 SBSPACENAME 的更多信息，请参阅《GBase 8s 管理员参考》中的配置参数主题。

智能大对象空间可以包含一个或多个块，如下图所示。当智能大对象空间包含多个块时，不能指定数据所在的块。

您可以在任何时候添加更多的块。监视智能大对象空间块的填充度以及预期是否有必要向智能大对象空间分配更多的块，是高优先级数据库服务器管理员任务。

图: 链接逻辑和物理存储单元的智能大对象空间



数据库服务器可使用智能大对象空间存储包含智能大对象的表列。数据库服务器使用数据库空间可存储其余的表列。

您可以为智能大对象空间建立镜像以便在介质故障的情况下加速恢复。有关更多信息，请参阅镜像。

有关使用 gspaces 执行以下任务的信息，请参阅管理磁盘空间。

- 创建智能大对象空间
- 向智能大对象空间添加块
- 更改智能大对象的存储特征
- 创建临时智能大对象空间
- 删除智能大对象空间

智能大对象空间的存储特征

作为数据库服务器管理员，您可以使用这些存储特征的系统缺省值，或者可以在用 gspaces -c 创建智能大对象空间时在 **-Df** 标记中指定这些特征。稍后，您可以使用 gspaces -ch 选项更改这些智能大对象空间特征。该管理员或程序员可以覆盖存储特征的这些缺省值以及单个表的属性。

智能大对象空间的扩展数据块大小

与表中的扩展数据块类似，智能大对象空间中的扩展数据块包含了存储智能大对象数据的邻接页的集合。

智能大对象空间中的分配单元是扩展数据块。数据库服务器根据一组启发式搜索（如写操作中的字节数）计算智能大对象的数据块大小。例如，如果操作请求写入 30 KB，那么数据库服务器会尝试分配一个大小为 30 KB 的扩展数据块。

重要：对于大多数应用程序来说，必须使用数据库服务器为扩展数据块的大小计算的值。

如果您知道智能大对象的大小，您可以使用下列函数之一来设置扩展数据块大小。数据库服务器可将整个智能大对象分配为一个扩展数据块（如果块中存在该大小的扩展数据块）：

- GBase 8s ESQL/C ifx_lo_specset_estbytes 函数

  有关智能大对象的 GBase 8s ESQL/C 函数的更多信息，请参阅《GBase 8s ESQL/C 程序员手册》。

平均智能大对象大小

智能大对象的长度经常会变化。您可以提供智能大对象的平均大小以计算智能大对象空间的空间。可使用 gspaces -c -Df 选项的 AVG_LO_SIZE 标记来指定此平均大小。

要指定元数据区域的大小和位置，请在 **gspaces** 命令中指定 **-Ms** 和 -Mo 标志。如果未使用 **-Ms** 标志，数据库服务器将使用 AVG_LO_SIZE 的值来估计要为元数据区域分配的空间量。有关更多信息，请参阅计算智能大对象空间元数据的大小。

缓冲方式

当您创建智能大对象空间时，缺省的缓冲方式打开，这意味着要在共享内存的常驻部分使用缓冲池。

作为数据库管理员，您可以用 gspaces -c -Df 选项的 BUFFERING 标记来指定缓冲方式。缺省值是“buffering=ON”，这意味着要使用缓冲池。如果您关闭缓冲，数据库服务器将在共享内存的虚拟部分中使用专用缓冲区。

重要： 通常，如果对智能大对象的读取和写入操作小于 8 KB，那么创建智能大对象空间时不要指定缓冲方式。如果正读取或写入短的数据块（如 2 KB 或 4 KB），那么保留缺省值“buffering=ON”以便获取更好的性能。

上次访问时间

创建智能大对象空间时，可以指定数据库服务器是否必须保存上次用 gspaces -c -Df 选项的 ACCESSTIME 标记读取或更新的智能大对象的时间。 缺省值为“ACCESSTIME=OFF”。数据库服务器会将此最近访问时间保留在元数据区域中。

锁定方式

创建智能大对象空间时，可以指定数据库服务器是使用 gspaces -c -Df 选项的 LOCK_MODE 标记来锁定整个智能大对象还是智能大对象中一定范围的字节。缺省值为“LOCK_MODE=BLOB”，意味着锁定整个智能大对象。

日志记录

创建智能大对象空间时，可以指定是否为智能大对象启用日志记录。缺省值是不进行日志记录。有关更多信息，请参阅记录智能大对象空间和智能大对象。

重要： 当您使用日志记录数据库时，请为智能大对象空间启用日志记录。如果发生故障需要恢复日志，您可以使这些智能大对象与数据库中的其余智能大对象保持一致。

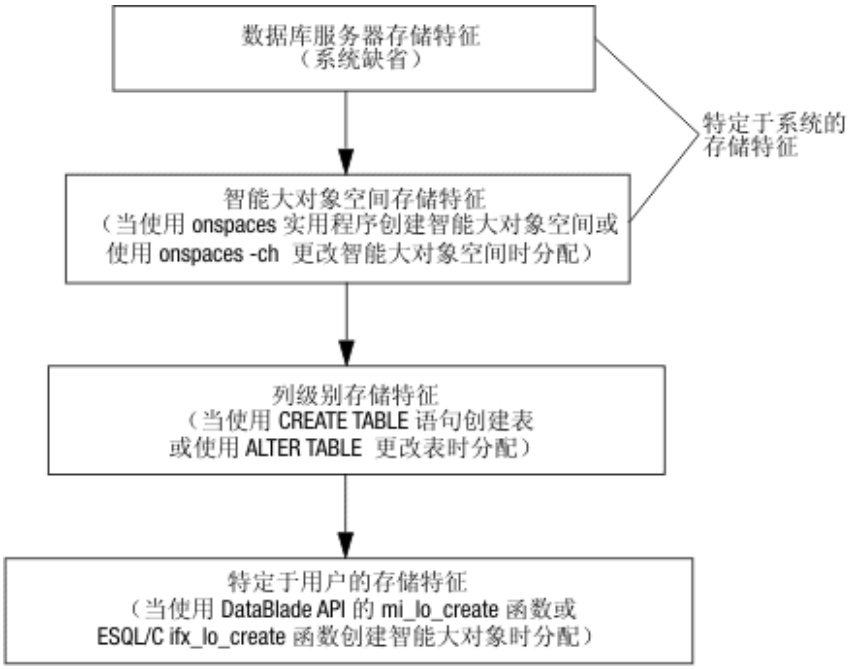
可使用 gspaces -c -Df 选项的 LOGGING 标记来指定日志记录状态。缺省值为“LOGGING=off”。可以用 gspaces -c -Df 选项更改日志记录状态。可以用 SQL 语句 CREATE TABLE 或 ALTER TABLE 中的 PUT 子句覆盖此日志记录状态。有关这些 SQL 语句的更多信息，请参阅《GBase 8s SQL 指南：语法》。

当您为智能大对象空间打开日志记录时，智能大对象将穿过共享内存的常驻部分。虽然应用程序可以检索多个智能大对象，但仍必须考虑可能穿过缓冲池和逻辑日志缓冲区的更大数据大小。有关更多信息，请参阅访问智能大对象。

智能大对象空间特征的继承级别

智能大对象空间特征的四种继承级别是系统、智能大对象空间、列和智能大对象。您可以使用智能大对象空间属性的系统缺省值或覆盖特定智能大对象空间、表中的列或智能大对象的系统缺省值。下图显示了智能大对象的存储特征层次结构。

图: 存储特征层次结构



此图显示您可以用以下方法覆盖系统缺省值：

- 使用 gspaces -c -S 命令的 **-Df** 标记可覆盖特定智能大对象空间的系统缺省值。可以稍后用 gspaces -ch 选项更改智能大对象空间的这些智能大对象空间属性。有关 -Df 标记的有效范围的更多信息，请参阅《GBase 8s 管理员参考》中的 gspaces 主题。
- 在 CREAT TABLE 或 ALTER TABLE 语句的 PUT 子句中指定这些属性时，可以覆盖特定列的系统缺省值。

关于智能大对象空间的更多信息

下表列出了有关与使用和管理智能大对象空间相关的各种任务的信息源。

表 1. 查找智能大对象空间任务的信息

任务	引用
为智能大对象设置内存配置参数	管理共享内存
了解智能大对象页	智能大对象页
为智能大对象空间指定 I/O 特征	智能大对象空间的存储特征中的 gspaces 选项
为智能大对象空间分配空间	创建智能大对象空间
向智能大对象空间添加块	向智能大对象空间添加块
定义或改变智能大对象的存储特征	更改智能大对象的存储特征 *《GBase 8s SQL 指南：语法》* 中的 CREATE TABLE 或 ALTER TABLE 语句的 PUT 子句
监视智能大对象空间	监视智能大对象空间
为智能大对象空间设置日志记录	记录智能大对象空间和智能大对象
备份智能大对象空间	备份智能大对象空间

任务	引用
检查智能大对象空间的一致性	验证元数据
了解智能大对象空间的结构	*《GBase 8s 管理员参考》*中有关磁盘结构的主题
使用智能大对象空间的 gspaces	*《GBase 8s 管理员参考》*中有关实用程序的主题
用 CLOB 或 BLOB 数据类型创建表	*《GBase 8s SQL 指南：语法》*
访问应用程序中的智能大对象	*《GBase 8s ESQL/C 程序员手册》*

3.5.10 临时智能大对象空间

使用*临时智能大对象空间*可在不用元数据日志记录和用户数据日志记录的情况下存储临时智能大对象。如果您将临时智能大对象存储在标准的智能大对象空间中时，将记录元数据。临时智能大对象空间类似于临时数据库空间。要创建临时智能大对象空间，使用带有 -t 选项的 **gspaces -c -S** 命令。

您可将临时大对象存储在标准的智能大对象空间或临时的智能大对象空间中。

- 如果在 SBSPACETEMP 参数中指定临时智能大对象空间，那么可在该临时智能大对象空间中存储临时智能大对象。
- 如果在 SBSPACENAME 参数中指定标准智能大对象空间，那么可在该标准智能大对象空间中存储临时和永久智能大对象。
- 如果在 CREATE TEMP TABLE 语句中指定临时智能大对象空间名称，那么可在该临时智能大对象空间中存储临时智能大对象。
- 如果在 CREATE TABLE 语句中指定永久智能大对象空间名称，那么可在该永久智能大对象空间中存储临时智能大对象。
- 如果省略 SBSPACETEMP 和 SBSPACENAME 参数并创建智能大对象，那么可能会显示错误消息 -12053。
- 如果在 SBSPACENAME 参数中指定临时智能大对象空间，那么就不能在该智能大对象空间中存储永久智能大对象。 您可将临时智能大对象存储在该智能大对象空间中。

比较临时和标准智能大对象空间

下表比较了标准和临时智能大对象空间。

表 1. 临时和标准智能大对象空间

特征	标准智能大对象空间	临时智能大对象空间
存储智能大对象	是	否
存储临时智能大对象	是	是
记录元数据	始终记录元数据	不记录元数据

特征	标准智能大对象空间	临时智能大对象空间
记录用户数据	如果 LOGGING=ON，对于临时智能大对象，不记录用户数据；对于永久智能大对象，会记录用户数据	不记录用户数据 创建和删除空间，且记录块的添加
快速恢复	是	否（智能大对象空间在数据库服务器重新启动时清空） 要设置共享内存而不清除临时智能大对象，请指定 oninit -p。如果保留临时大对象，那么它们的状态是不确定的。
备份与复原	是	否
添加和删除块	是	是
配置参数	SBSPACENAME	SBSPACETEMP

临时智能大对象

使用*临时智能大对象*可存储不需要从备份复原或在快速恢复中记录重放的文本或映像数据（CLOB 或 BLOB）。临时智能大对象将在该用户会话中持续，并且比智能大对象更快更新。

除非在 ifx_lo_specset_flags 或 mi_lo_specset_flags 函数中设置了 LO_CREATE_TEMP 标志，否则将以与创建永久智能大对象相同的方式创建临时智能大对象。使用 mi_lo_copy 或 ifx_lo_copy 从临时智能大对象创建永久智能大对象。

重要： 只可在临时表中存储指向临时大对象的指针。如果您将它们存储在标准表中并重新引导数据库服务器，那么结果会生成一条错误信息，称大对象不存在。

下表比较了标准和临时智能大对象。

表 1. 临时和标准智能大对象

特征	智能大对象	临时智能大对象
创建标志	LO_CREATE_LOG 或 LO_CREATE_NOLOG	LO_CREATE_TEMP
回滚	是	否
日志记录	是，如果打开	否
持续时间	永久（直到用户将其删除）	在用户会话或事务结束时删除
存储的表类型	永久或临时表	临时表

3.5.11 外部空间

*外部空间*是与表示外部数据位置的随机字符串关联的逻辑名。外部空间所引用的资源取决于访问其内容的用户定义的访问方法。

例如，数据库用户可能需要对以专有格式编码的二进制文件进行访问。首先，开发者会创建访问方法，它是存取数据的一组例程。这些例程将负责数据库服务器和外部文件之间的所有交互。DBA 然后将添加一个外部空间，该外部空间会将该文件作为其对数据库的目标。DBA 在外部空间中创建了表之后，用户可通过 SQL 语句访问这些专有文件中的数据。要找到这些文件，请使用外部空间信息。

外部空间不必是文件名。例如，它可以是网络位置。存取数据的例程可以用任何方式使用在与外部空间关联的字符串中找到的信息。

3.5.12 数据库

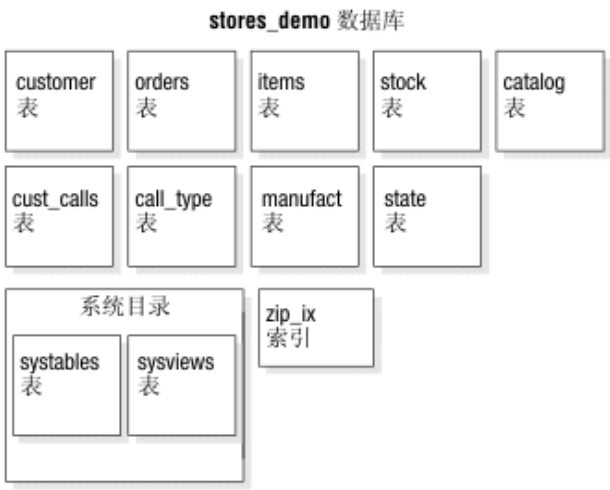
数据库是包含表和索引的逻辑存储单元。 （请参阅表。）每个数据库还包含跟踪有关数据库中许多元素（包括表、索引、SPL 例程和完整性约束）的信息的系统目录。

数据库位于由 CREATE DATABASE 语句指定的数据库空间中。当未在 CREATE DATABASE 语句中明确指定数据库空间时，数据库将位于根数据库空间中。当在 CREAT DATABASE 语句中明确指定了数据库空间时，此数据库空间的位置就是以下表的位置：

- 数据库系统目录表
- 属于该数据库的任何表

下图显示了 stores_demo 数据库中包含的表。

图: stores_demo 数据库



应用于数据库的大小限制与它们在数据库空间中的位置相关。要确保数据库中的所有表都创建在特定的物理设备上，只能向该设备指定一个块并创建仅包含该块的数据库空间。将数据库放置在该数据库空间中。当您在指定给特定的物理设备的块中放置数据库时，该数据库的大小不能超过该块的大小。

有关如何列出创建的数据库的指示信息，请参阅显示数据库。

3.5.13 表

在关系数据库系统中，*表*是一行列标题加上零行或多行数据值。列标题行标识了一个或多个列以及每一列的数据类型。

当创建表时，数据库服务器会为称为扩展数据块的页块中的表分配磁盘空间。（请参阅扩展数据块数。）您可以指定第一个以及任何后续扩展数据块的大小。

可以在创建表（通常用 CREAT TABLE 的 IN *dbspace* 选项）时，通过命名数据库空间将表放到特定的数据库空间中。不指定数据库空间时，数据库服务器会将该表放置在数据库所在数据库空间中。

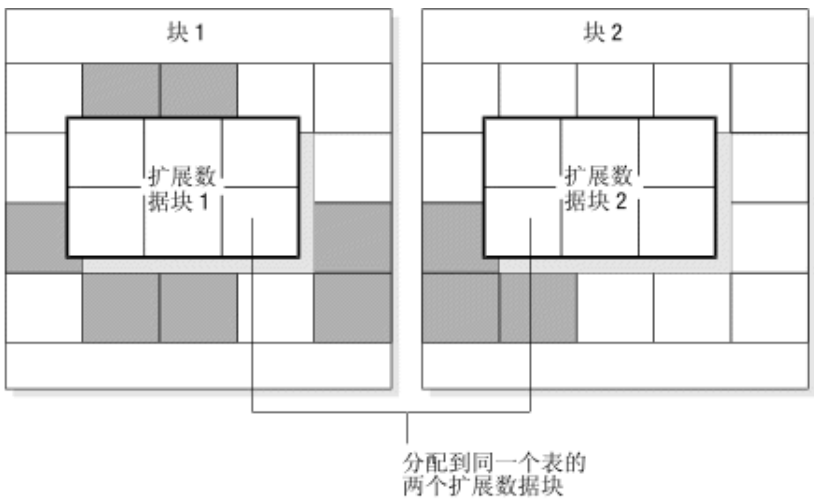
您还可以：

- 在多个数据库空间上将表分段。然而，您不能将这些分段放在不同页大小的数据库空间中。所有分段可能都需要有相同的页大小。
  您必须为指定哪些表行位于哪些数据库空间的表定义分布方案。
- 如果分段表使用基于表达式的或循环分发方案，请在单个数据库空间内创建分段表的多个分区。

表或表分段完全位于在其中创建它们的数据库空间中。数据库服务器管理员可以使用此事实来限制表的增长，方法是将表放置在数据库空间中然后当数据库空间变满时拒绝向其添加块。

由扩展数据块组成的表可以跨多个块，如下图所示。

图: 跨多个块的表



简单大对象位于 BLOB 页中，这些 BLOB 页可以位于带有表的数据页的数据库空间中或位于独立的 BLOB 空间中。

简单大对象位于 BLOB 页中，这些 BLOB 页可以位于带有表的数据页的数据库空间中或位于独立的 BLOB 空间中。使用 Optical Subsystem，您还可以将简单大对象存储在光存储子系统中。

已损坏的表

以下各项可能损坏表：

- 不正确的缓冲区清空

- 用户错误
- 在块顶部安装文件系统或另一个块
- 当更改的范围超出您要求的范围时进行删除或更新

已损坏的索引会导致表看起来已损坏，即使该表实际并未损坏也是如此。

gcheck 命令不能修复大多数已损坏的表。如果某个页已损坏，gcheck 可检测并尝试修理该页，但无法更正该页中的数据。

3.5.14 GBase 8s 的表类型

您可以在 GBase 8s 上的日志记录数据库中创建日志记录或非日志记录表。两种表类型为 STANDARD（日志记录表）和 RAW（非日志记录表）。缺省标准表如同较早版本中创建的没有指定特殊关键字的表一样。您既可以创建 STANDARD 也可以创建 RAW 表，并且可以将表从一个类型更改为另一个类型。

在非日志记录数据库中，STANDARD 和 RAW 表都是非日志记录的。在非日志记录数据库中，STANDARD 和 RAW 表之间的唯一区别在于 RAW 表不支持主键约束、唯一约束、引用约束或回滚。然而，可以为这些表建立索引并进行更新。

下表列出了 GBase 8s 中可用类型表的属性。标志值是 **systables** 的 **flags** 列中每种表类型的十六进制值。

表 1. GBase 8s 的表类型

特征	STANDARD	RAW	TEMP
永久	是	是	否
已记录	是	否	是
索引	是	是	是
约束	是	无引用约束或唯一约束 允许 NULL 和 NOT NULL 约束	是
回滚	是	否	是
可复原	是	是，如果未更新	否
可复原	是	是，如果未更新	否
可载入	是	是	是
Enterprise Replication 服务器	是	否	否
高可用性集群中的主服务器	是	是，不能更改日志记录方式	是
高可用性集群中的辅助服务器	是	是，但任何操作都不可访问	是

特征	STANDARD	RAW	TEMP
标志值	无	0x10	无

标准永久表

STANDARD 表与数据库服务器创建的已记录的数据库中的表相同。STANDARD 表不使用轻量级追加。所有操作都是一条条地记录的，因此 STANDARD 表可以恢复和回滚。可备份与复原 STANDARD 表。日志记录使您能够在执行热复原或时间点复原时应用自上次物理备份起的更新。Enterprise Replication 允许位于 STANDARD 表上。

STANDARD 表是日志记录和非日志记录这两种数据库上的缺省类型。STANDARD 表如果存储在日志记录数据库中，将被记录；如果存储在非日志记录数据库中，将不被记录。

RAW 表

RAW 表是非日志记录的永久表，类似于非日志记录数据库中的表。支持但不记录 RAW 表中行内的更新、插入和删除操作。可在 RAW 表中定义索引，但是不能在 RAW 表中定义唯一约束、主键约束或引用约束。不支持将轻量级追加用于加载 RAW 表，但在 High-Performance Loader (HPL) 操作中和指定 INTO TEMP ... WITH NO LOG 的查询中例外。

无论是存储在日志记录数据库还是非日志记录数据库中，RAW 表都具有相同属性。如果更新了 RAW 表，那么除非在更新之后执行 0 级备份，否则不能可靠复原数据。如果自备份后尚未更新过表，那么可以从上次物理备份复原 RAW 表，但是仅备份逻辑日志并不足以复原 RAW 表。快速恢复可回滚 STANDARD 表上未完成事务，但不会回滚 RAW 表上的未完成事务。

RAW 表适用于初始装入和数据的验证。要装入 RAW 表，可以使用任何装入实用程序，包括 dbexport、DB-Access 的 LOAD 语句或表达式方式下的 HPL。如果在装入 RAW 表时发生错误或故障，那么得到的数据是发生故障时磁盘上的任何数据。

限制： 不要在事务中使用 RAW 表。在已装入数据后，请首先使用 ALTER TABLE 语句将表更改为 STANDARD 类型并执行 0 级备份，然后再在事务中使用该表。

限制： 不要在 RAW 或 TEMP 表上使用 Enterprise Replication。

在高可用性集群环境中使用 RAW 表时，有一些限制。因为不记录对 RAW 表所作修改，并且辅助服务器（包括 HDR、RSS 和 SDS）使用日志记录与主服务器保持同步，所以将限制对 RAW 表执行某些操作：

- 在主服务器上，可以创建、删除和访问 RAW 表；但是，不允许将表方式从未记录更改为已记录，或从已记录更改为未记录。在高可用性集群环境中更改表的方式会生成错误 -19845。
- 在辅助服务器（HDR、SDS 或 RSS）上，任何操作都不能访问 RAW 表。尝试从 SQL 访问 RAW 表会生成错误 -19846。

Temp 表

Temp 表是临时已记录的表，它们将在用户会话关闭、数据库服务器关闭或故障后重新引导时删除。Temp 表支持索引、约束和回滚。您不能复原、备份或复原 Temp 表。Temp 表支持批量操作（如轻量级追加），快速将行添加到每个表分段的末尾。

表类型的属性

这些主题说明了装入表、快速恢复以及表类型的备份与复原。

将数据装入表

GBase 8s 创建在缺省情况下使用日志记录的 STANDARD 表。数据仓库应用程序可以有需要花费长时间装入的巨型表。装入非日志记录表比装入日志记录表要快。可使用 CREATE RAW TABLE 语句创建 RAW 表或使用 ALTER TABLE 语句在装入 STANDARD 表之前将该表更改为 RAW 表。在装入表后，请对该表运行 UPDATE STATISTICS。

有关使用 ALTER TABLE 将表从日志记录更改为非日志记录的更多信息，请参阅《GBase 8s SQL 指南：语法》。

表类型的快速恢复

下表显示了 GBase 8s 中可用的表类型的快速恢复方案。

表 1. 表类型的快速恢复

表类型	快速恢复行为
Standard	快速恢复成功。所有已落实的日志记录将前滚，而所有未完成的事务将回滚。
RAW	如果自上次修改原始表后已完成检查点，那么所有数据都可恢复。 上一个检查点后的插入、更新和删除将丢失。 RAW 表中未完成的事务将不回滚。

备份与复原 RAW 表

下表说明了 GBase 8s 中可用的表类型的备份方案。

表 1. 在 GBase 8s 上备份表

表类型	是否允许备份？
Standard	是。
Temp	否。
RAW	是。如果更新 RAW 表，必须对其进行备份，这样就可以复原其中的所有数据。仅备份逻辑日志是不够的。

重要： 在装入 RAW 表或将 RAW 表更改为 STANDARD 类型后，必须执行 0 级备份。

下表显示了这些表类型的复原方案。

表 2. 在 GBase 8s 上复原表

表类型	是否允许复原？

表类型	是否允许复原？
Standard	是。热复原、冷复原和时间点复原都可以起作用。
Temp	否。
RAW	复原 RAW 表时，它只包含上次备份时位于磁盘上的数据。由于未记录 RAW 表，因此任何上次备份以来发生的更改都不会复原。

备份与复原 RAW 表

下表说明了 GBase 8s 中可用的表类型的备份方案。

表 1. 在 GBase 8s 上备份表

表类型	是否允许备份？
Standard	是。
Temp	否。
RAW	是。如果更新 RAW 表，必须对其进行备份，这样就可以复原其中的所有数据。仅备份逻辑日志是不够的。

重要： 在装入 RAW 表或将 RAW 表更改为 STANDARD 类型后，必须执行 0 级备份。

下表显示了这些表类型的复原方案。

表 2. 在 GBase 8s 上复原表

表类型	是否允许复原？
Standard	是。热复原、冷复原和时间点复原都可以起作用。
Temp	否。
RAW	复原 RAW 表时，它只包含上次备份时位于磁盘上的数据。由于未记录 RAW 表，因此任何上次备份以来发生的更改都不会复原。

临时表

数据库服务器必须为以下两类临时表提供磁盘空间：

- 使用 SQL 语句（例如 CREATE TEMP TABLE. . . 或（使用 Extended Parallel Server）SELECT .... INTO SCRATCH）创建的临时表. .
- 数据库服务器在其处理查询时创建的临时表

请确保您的数据库服务器已为用户创建的和数据库服务器创建的这两种临时表配置了足够的临时空间。对数据库服务器的一些使用需要与永久存储空间一样多的临时存储空间，或更多。

在缺省情况下，数据库服务器会将临时表存储在根数据库空间中。如果您决定不在根数据库空间中存储临时表，那么可以使用 DBSPACETEMP 环境变量或 DBSPACETEMP 配置参数来指定临时表的数据库空间列表。

创建的临时表

可以使用以下任一 SQL 语句创建临时表：

- CREATE TABLE 语句的 TEMP TABLE 选项
- SELECT 语句的 INTO TEMP 子句，例如 `SELECT * FROM customer INTO TEMP` *cust_temp*

只有创建了临时表的会话才可以使用该表。 当会话退出时，该表将自动删除。

当您创建临时表时，数据库服务器将使用下列条件：

- 如果用于填充 TEMP 表的查询未产生任何行，那么数据库服务器将创建一个空的且未分段的表。
- 如果查询生成的行未超过 8 KB，那么临时表将只位于一个数据库空间中。
- 如果生成的行超过 8 KB，那么数据库服务器会创建多个分段并使用循环分段存储方案进行填充，除非您为该表指定了分段存储方法和位置。

如果使用 CREATE TEMP 和 SELECT...INTO TEMP SQL 语句并且 DBSPACETEMP 已被设置：

- 列表中的 LOGGING 数据库空间被用来创建指定或暗示 WITH LOG 子句的表。
- 列表中的 NON-LOGGING 临时数据库空间被用来创建指定 WITH NO LOG 子句的表。

当使用了 CREATE TEMP 和 SELECT...INTO TEMP SQL 语句，并且 DBSPACETEMP 还未设置或不包含正确类型的数据库空间时，GBase 8s 会使用数据库的数据库空间来存储临时表。请参阅《GBase 8s SQL 指南：语法》以获取更多信息。

存储用户创建的临时表的位置

如果应用程序允许您指定临时表的位置，那么您可以指定日志记录空间，也可以指定专门为临时表创建的非日志记录空间。

有关创建临时数据库空间的信息，请参阅《GBase 8s 管理员参考》中的 gspaces 主题。

如果未指定临时表的位置，那么数据库服务器会将临时表存储在指定为 DBSPACETEMP 配置参数或环境变量的自变量的某一个空间中。数据库服务器将记住上一个用于临时表的数据库空间的名称。当数据库服务器接收到临时存储空间的另一个请求时，它将使用下一个可用的数据库空间均匀地将 I/O 分布在临时存储空间中。

有关在您未列出作为 DBSPACETEMP 的自变量的任何列表时数据库存储临时表位置的信息，请参阅《GBase 8s 管理员参考》 中 DBSPACETEMP 一节。

当您使用应用程序创建临时表时，您可以使用该临时表直到该应用程序退出或执行下列操作之一为止：

- 关闭创建该表所在的数据库并打开其他数据库服务器中的数据库
- 关闭创建该表所在的数据库
- 显式地删除临时表

数据库服务器创建的临时表

数据库服务器有时会在对数据库运行查询或备份该数据库时创建临时表。

该数据库服务器可能在下列任何情况下创建临时表：

- 包含 GROUP BY 或 ORDER BY 子句的语句
- 使用带有 UNIQUE 或 DISTINCT 关键字的聚集函数的语句
- 使用自动索引或散列连接的 SELECT 语句
- 复杂 CREATE VIEW 语句
- 创建滚动游标的 DECLARE 语句
- 包含相关子查询的语句
- 包含在 IN 或 ANY 子句中发生的子查询的语句
- CREATE INDEX 语句

当启动表创建的进程完成时，数据库服务器会删除其创建的临时表。

如果数据库服务器在没有除去临时表的情况下关闭，那么数据库服务器在下次启动时将除去这些临时表。要启动数据库服务器而不除去临时表，请运行带 **-p** 选项的 oninit 命令。

应用程序和分析工具可以定义其中派生表包含与基本表连接的多个视图的查询，可能包括数百列。数据库服务器会尝试将视图或派生表折叠放入主查询中。无法折叠的任何此类视图或派生表都会具体化到临时表中。临时表会排除主查询中未引用的所有列。临时表仅使用在 Projection 子句以及父查询的其他子句（包括 WHERE、HAVING、GROUP BY 和 ON 子句）中引用的列进行创建。

通过从系统生成的临时表中排除主查询中未引用的任何列，这一简化模式可以节省存储资源，避免在未使用的列中进行不必要的数据 I/O，从而提高查询性能。

但是，在嵌套查询中，只会检查父查询中来自视图和派生表的投影列，而不会检查高于直接父查询的级别中的投影列。

重要： 除了临时表以外，数据库服务器还使用临时磁盘空间来存储发生备份时覆盖的数据记录的前映像，以及用于内存中发生的查询处理中的溢出。确保正确设置了 **DBSPACETEMP** 环境变量或 DBSPACETEMP 配置参数，以指定具有满足您需求的足够空间的数据库空间。如果指定的数据库空间中空间不足，备份将失败，并且将使用根数据库空间，或者在填满根数据库空间之后，备份将失败。

存储数据库服务器创建的临时表的位置

当数据库服务器创建临时表时，它会将临时表存储在您在 DBSPACETEMP 配置参数或环境变量中指定的某个数据库空间中。环境变量将取代配置参数。

当您未在 DBSPACETEMP 中指定任何临时数据库空间，或指定的临时数据库空间的空间不够时，数据库服务器将根据以下规则在标准数据库空间中创建该表：

- 如果已使用 CREATE TEMP TABLE 创建了临时表，那么数据库服务器会将该表存储在包含该表的数据库的数据库空间中。
- 如果用 SELECT 语句的 INTO TEMP 选项创建临时表，那么数据库服务器在根数据库空间中存储此表。

3.5.15 表空间

数据库服务器管理员有时必须跟踪特定表的磁盘使用。*表空间*包含分配到给定表或表分段（如果表已分段）的所有磁盘空间。独立表空间包含已分配给关联索引的磁盘空间。

例如，表空间与块的任意特定部分或者甚至是任意特定块都不一致。组成表空间的索引和数据可能散射在您所有块中。然而，表空间代表了专用于特定表的块中空间的方便记帐实体。

表中的最大表空间数

您可以在表中指定最多达 2**20（或 1,048,576）的表空间数。

表和索引表空间

表表空间包含下列类型的页：

- 分配给数据的页
- 分配给索引的页
- 用于在数据库空间中存储 TEXT 或 BYTE 数据的页（不是用于在 BLOB 空间中存储 TEXT 或 BYTE 数据的页）
- 跟踪页在表扩展数据块内使用的位图页

索引表空间包含下列类型的页：

- 分配给索引的页
- 跟踪页在索引扩展数据块内使用的位图页

下表说明了用于构成部分 **stores_demo** 数据库的三个表的表空间。每个表空间上只有一个表（或表分段）。索引位于独立于关联表的表空间中。BLOB 页代表存储在数据库空间中的 TEXT 或 BYTE 数据。

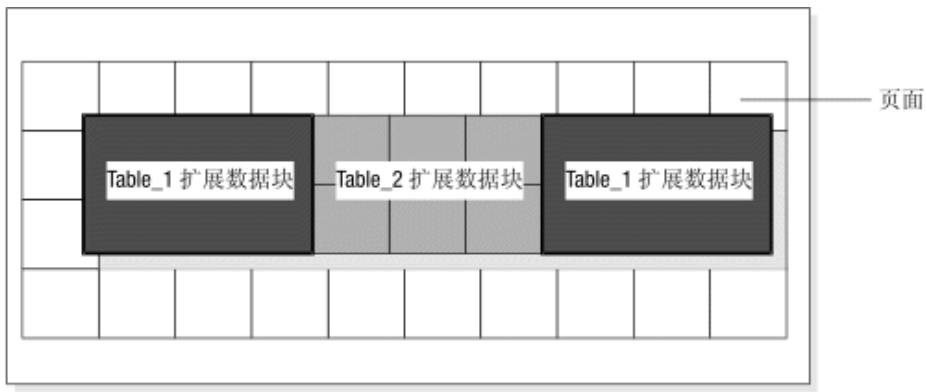
图: stores_demo 数据库中的样本表空间

扩展数据块交错

数据库服务器将属于表空间的页分配为扩展数据块。虽然扩展数据块中的页是连续的，但扩展数据块还是可能会分散在表所在整个数据库空间中（甚至在不同的块上）。

下图描绘的情境是：两个非连续扩展数据块属于 **table_1** 的表空间，而第三个扩展数据块属于 **table_2** 的表空间。**table_2** 扩展数据块位于第一个 table_1 扩展数据块和第二个 **table_1** 扩展数据块之间。当发生这种情况时，扩展数据块会交错。 因为在 **table_1** 中的顺序访问搜索需要磁盘头在 **table_2** 扩展数据块中进行搜寻，因此性能会比 **table_1** 扩展数据块连续时要差。

图: 属于一个数据库空间中两个不同表空间的三个扩展数据块



3.5.16 表分段存储和数据存储

分段存储功能将赋予您对数据库存储数据的位置的额外控制。您不但可以指定单个表和索引的位置，而且可以指定表和索引*分段*的位置（这些分段是位于不同存储空间的表或索引的不同部分）。您可以将下列存储空间分段：

- 数据库空间
- 智能大对象空间

通常您在最初创建表时将表分段。CREATE TABLE 语句将采用以下某种表格：

```
CREATE TABLE tablename ... FRAGMENT BY ROUND ROBIN IN dbspace1,
dbspace2, dbspace3;


CREATE TABLE tablename ...FRAGMENT BY EXPRESSION
    <Expression 1> in dbspace1,
    <Expression 2> in dbspace2,
    <Expression 3> in dbspace3;
```

FRAGMENT BY ROUND ROBIN 和 FRAGMENT BY EXPRESSION 关键字指的是两种不同的分布方案。两个语句都将分段与数据库空间关联。

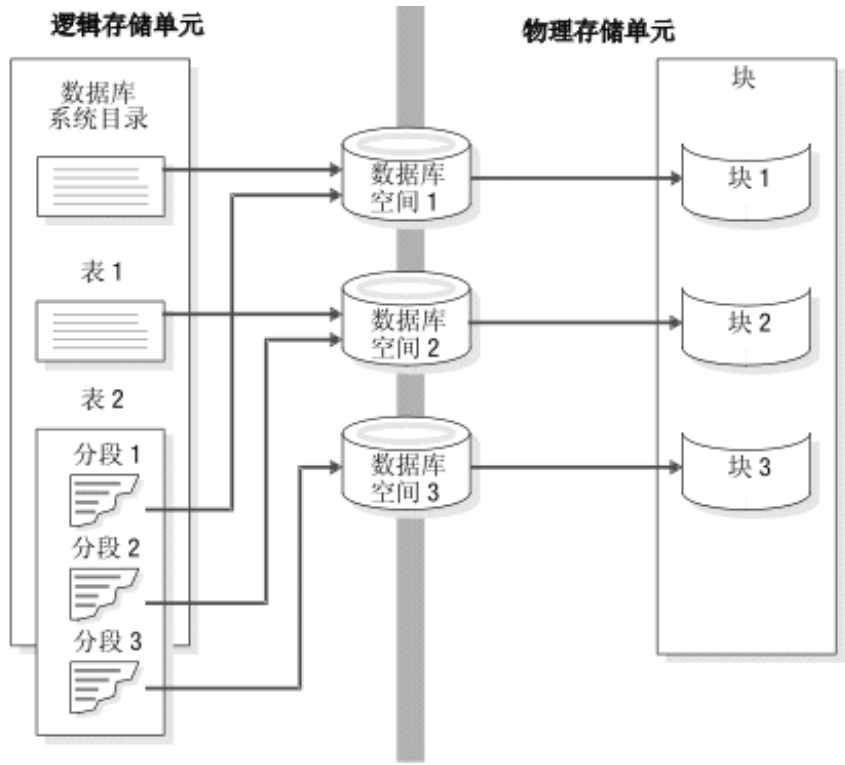
对表进行分段时，还可在相同的数据库空间内为表创建多个分区，如此示例中所示：

```
CREATE TABLE tb1(a int)
```

```
FRAGMENT BY EXPRESSION
        PARTITION part1 (a >=0 AND a < 5) in dbs1,
        PARTITION part2 (a >=5 AND a < 10) in dbs1
         ...
    ;
```

下图说明了分段在指定数据位置中的角色。

图: 链接逻辑单元（包括表分段）和物理存储单元的数据库空间



有关空间和分区的信息，请参阅管理磁盘空间。

3.5.17 存储数据所需的磁盘空间量

要确定需要多少磁盘空间，请遵循以下步骤：

1. 计算需要多少根数据库空间。
2. 估计要分配给所有数据库服务器数据库的总磁盘空间量，包括用于开销和增长的空间。

以下主题说明了这些步骤。

根数据库空间的大小

可以计算根数据库空间的大小，此空间存储了描述数据库服务器的信息。

要计算根数据库空间的大小，请考虑下列存储结构：

- 物理日志（最小 200 KB）
- 逻辑日志文件（最小 200 KB）

- 临时表
- 数据
- 系统数据库（**sysmaster**、**sysutils**、**syscdr**、**sysuuid**）以及系统目录（大小随版本的不同而变化）
- 保留页 (~24 KB)
- 表空间 tblspace（最小 100 到 200 KB）
- 额外空间

此估计值是初始化数据库服务器之前根数据库空间的大小。根数据库空间的大小取决于您是计划在根数据库空间还是在其他数据库空间中存储物理日志、逻辑日志和临时表。根数据库空间必须足够大以便在磁盘初始化期间进行最小大小配置。

建议： 使用小日志大小（例如，三个 1000 KB 的日志文件或总日志大小为 3000 KB）设置系统。在设置完成后，请创建新的数据库空间、移动逻辑日志文件并调整其大小、删除根数据库空间中的原始日志。然后将物理日志移至另一个数据库空间。此过程可最大限度降低日志在根数据库空间中的影响，原因是：

- 移动这些日志后，根数据库空间中不会有大量空间保留不用。
- 日志不会与根数据库空间在相同磁盘上争用空间和 I/O。

有关如何移动日志的详细信息，请参阅将逻辑日志文件移至另一个数据库空间和更改物理日志的位置和大小。

如果在服务器已初始化后，需要将根数据库空间设置得更大，可以向根数据库空间添加新块。此外，可以通过使用自动空间管理来扩展根数据库空间中的块。

物理日志和逻辑日志

存储在 ONCONFIG 参数 PHYSFILE 中的值定义了在最初创建数据库服务器时物理日志的大小。在使用 oninit -i 命令初始化磁盘空间并使数据库服务器进入联机方式后，请使用 glogadmin 实用程序更改物理日志的位置和大小。有关估算物理日志的建议包含在物理日志的大小和位置中。

要计算逻辑日志文件的大小，请将 ONCONFIG 参数 LOGSIZE 的值乘以逻辑日志文件数。有关估算逻辑日志的建议，请参阅估计日志文件的大小和数量。

临时表

分析最终用户应用程序以估计数据库服务器可能要求临时表具备的磁盘空间量。尝试估计这些语句中有多少将会并发运行。已返还的行和列所占据的空间将提供估计所需的空间量的良好基础。

数据库服务器在热复原期间创建的最大的临时表与逻辑日志的大小相等。您可以通过增加所有逻辑日志文件的大小来计算逻辑日志的大小。

您还必须分析最终用户应用程序以估计数据库服务器可能需要用于显式临时表的磁盘空间量。

关键数据

限制： 不要将数据库和表存储在根数据库空间中。为包含关键数据（如物理日志和逻辑日志）的根数据库空间和其他数据库空间建立镜像。估计需要为存储在根数据库空间中的表分配的磁盘空间量（如果有）。

额外空间

允许系统数据库的根数据库空间中的额外空间增长以便获得扩展保留页和充足的可用空间。扩展保留页数取决于数据库服务器中的主要块、镜像块、逻辑日志文件和存储空间的数目。

数据库所需空间量

数据库服务器数据存储需要的附加磁盘空间量取决于用户的需求，以及开销和增长。用户运行的每个应用程序都有不同的存储需要。 以下列表提出了一些您可以用来计算要分配的磁盘空间（根数据库空间除外）量的步骤：

- 决定必须存储多少数据库和表。计算每个数据库和表所需的空间量。
- 计算每个表的增长率并为每个表指定一些磁盘空间量以适应增长。
- 决定希望监视哪些数据库和表。

3.5.18 存储池

GBase 8s 的每个实例都具有存储池。 存储池中包含有关以下对象的信息：服务器可在必要时用于自动扩展现有数据库空间、临时数据库空间、智能大对象空间、临时智能大对象空间或 Blob 空间的目录、熟文件和原始设备。

如果存储空间低于 SP_THRESHOLD 配置参数中定义的阈值，GBase 8s 可自动运行空间扩充任务，方法是扩展空间中的现有块或添加新块。

使用 SQL 管理 API 可以执行以下操作：

- 添加、删除或修改用于描述存储池中的一个目录、熟文件或原始设备的条目。必要时，服务器可使用指定的目录、熟文件或原始设备来向现有存储空间自动添加空间。
- 通过以下方法控制存储池条目的使用方式：修改与扩充存储空间相关的两个不同的数据库空间大小，即扩展大小和创建大小。
- 将块标记为可扩展或不可扩展。
- 不希望 GBase 8s 自动扩充空间时，立即扩充空间大小。
- 立即按指定的最小量扩展块的大小。
- 通过存储池中的条目创建存储空间或块
- 将空的空间从已删除的存储空间或块返还给存储池

sysadmin 数据库中的 **storagepool** 表包含有关 GBase 8s 实例的存储池中所有条目的信息。

3.5.19 磁盘布局准则

有效磁盘布局具有以下典型目标：

- 限制磁盘头的移动
- 减少磁盘争用
- 平衡负载
- 最大化可用性

您必须在设计磁盘布局时对这些目标作一些折中。例如，将系统目录表、逻辑日志和物理日志分开可以帮助减少对这些资源的争用。然而，此操作也会增加必须执行系统复原的机率。

数据库空间和块准则

本主题列出了磁盘布局的一些常规策略，这些策略不需要有关特定数据库特征的任何信息：

- 将磁盘分区与块关联并为根数据库空间分配至少一个附加块。

  已经分区的磁盘可能需要使用偏移量。 有关详细信息，请参阅在 UNIX 上分配原始磁盘空间。

  提示： 有了块的 4 太字节的最大大小的限制，您可以通过为每个磁盘驱动器指定块来避免进行分区。

- 为关键数据库空间（根数据库空间、包含逻辑日志和逻辑日志文件的数据库空间）建立镜像。还可为使用率高的数据库和表建立镜像。

  可以在数据库空间级别上指定镜像。对于所有属于数据库空间的块来说，镜像可以打开也可以关闭。在不同的磁盘上定位主要和镜像数据库空间。理论上，不同的控制器将处理不同的磁盘。

- 在多个磁盘上分布临时表并将文件排序。

  要为临时表和排序文件定义数个数据库空间，可使用 gspaces -t。 当在不同的磁盘上放置这些数据库空间并将它们列在 DBSPACETEMP 配置参数中时，可以展开与临时表相关联的 I/O 并在多个磁盘上对文件进行排序。有关使用 DBSPACETEMP 配置参数或环境变量的信息，请参阅《GBase 8s 管理员参考》中有关配置参数的章节。

- 将物理日志保留在根数据库空间中，而将逻辑日志从根数据库空间移出。然而，如果您计划将系统目录存储在根数据库空间中，请将物理日志移至另一个数据库空间。

  有关日志存储的位置的建议，请参阅指定物理日志的位置和逻辑日志文件的位置。另请参阅将逻辑日志文件移至另一个数据库空间和更改物理日志的位置和大小。

- 要提高备份与复原的性能，请执行以下操作：
  - 将系统目录与它们跟踪的数据集群起来。
  - 如果使用 ON-Bar 对高速磁带机执行并行备份，请将这些数据库存储在多个小数据库空间中。

有关附加的性能建议，请参阅《GBase 8s 备份与复原指南》。

表位置准则

本主题将列出用于优化磁盘布局的一些策略，前提是数据库中存在有关表的某些特征。您可以使用表分段存储通过更高程度的控制来实现这些策略中的许多策略：

- 隔离独立磁盘上使用率高的表。

  要在其自身的磁盘设备上隔离使用率高的表，请将该设备指定给一个块，然后将同一个块指定给数据库空间。最后，将使用频繁的表放置在刚使用 CREAT TABLE 的 IN *dbspace* 选项创建的数据库空间中。

要显示对每个块的 I/O 操作级别，请运行 gstat -g iof 选项。

- 将使用率高的表在多个磁盘上分段。
- 将有关的表在数据库空间中分组。

  如果包含数据库空间的设备发生故障，该数据库空间中的所有表都是不可访问的。然而，其他数据库空间中的表将仍然可以访问。尽管在包含关键信息的数据库空间发生故障时必须执行冷复原，然而如果只是非关键的数据库空间发生故障，那么只须执行热复原即可。

- 将使用率高的表放置在磁盘的中间分区上。
- 优化表扩展数据块的大小。

有关 gstat 选项的信息，请参阅《GBase 8s 管理员参考》。

3.5.20 样本磁盘布局

当开始组织磁盘空间时，数据库服务器管理员通常会想到下列目标中的一个或多个目标：

- 高性能
- 高可用性
- 简单和频繁的备份与复原

满足这些目标中的任何一个都需要折中。例如，将系统配置成高性能通常会导致数据的可用性降低。下列各节提供了一个示例，该示例中数据库服务器管理员必须在磁盘资源有限的情况下作出磁盘布局的选择。下节描述两种不同的磁盘布局解决方案。第一个解决方案代表性能优化，而第二个解决方案代表可用性和复原优化。

样本磁盘布局的设置是使用 stores_demo 数据库的结构（但不是卷）的虚构的体育用品数据库。 在此示例中，数据库服务器被配置成处理大约 350 个用户和 3 千兆字节的数据。磁盘空间的资源显示在下表中。

磁盘驱动器	驱动器大小	高性能
磁盘 1	2.5 千兆字节	否
磁盘 2	3 千兆字节	是
磁盘 3	2 千兆字节	是
磁盘 4	1.5 千兆字节	否

数据库包含两个大表：**cust_calls** 和 **items**。假设这两个表都包含多于 1,000,000 个行。**cust_calls** 表提供了客户致电经销商的所有通话记录。items 表包含经销商已发货的每张订单的行式项目。

数据库包含两个使用率高的表：**items** 和 **orders**。这两个表都要经受来自整个国家或地区的用户的持续访问。

其余的表是低容量表，数据库服务器将使用这些表查询数据，如：邮政编码或制造商。

表名	最大大小	访问率
cust_calls	2.5 千兆字节	低

表名	最大大小	访问率
items	0.5 千兆字节	高
orders	50 兆字节	高
customers	50 兆字节	低
stock	50 兆字节	低
catalog	50 兆字节	低
manufact	50 兆字节	低
state	50 兆字节	低
call_type	50 兆字节	低

性能优先级最高时的样本布局

下图显示了一种针对性能优化的磁盘布局。此磁盘布局使用下列策略提高性能：

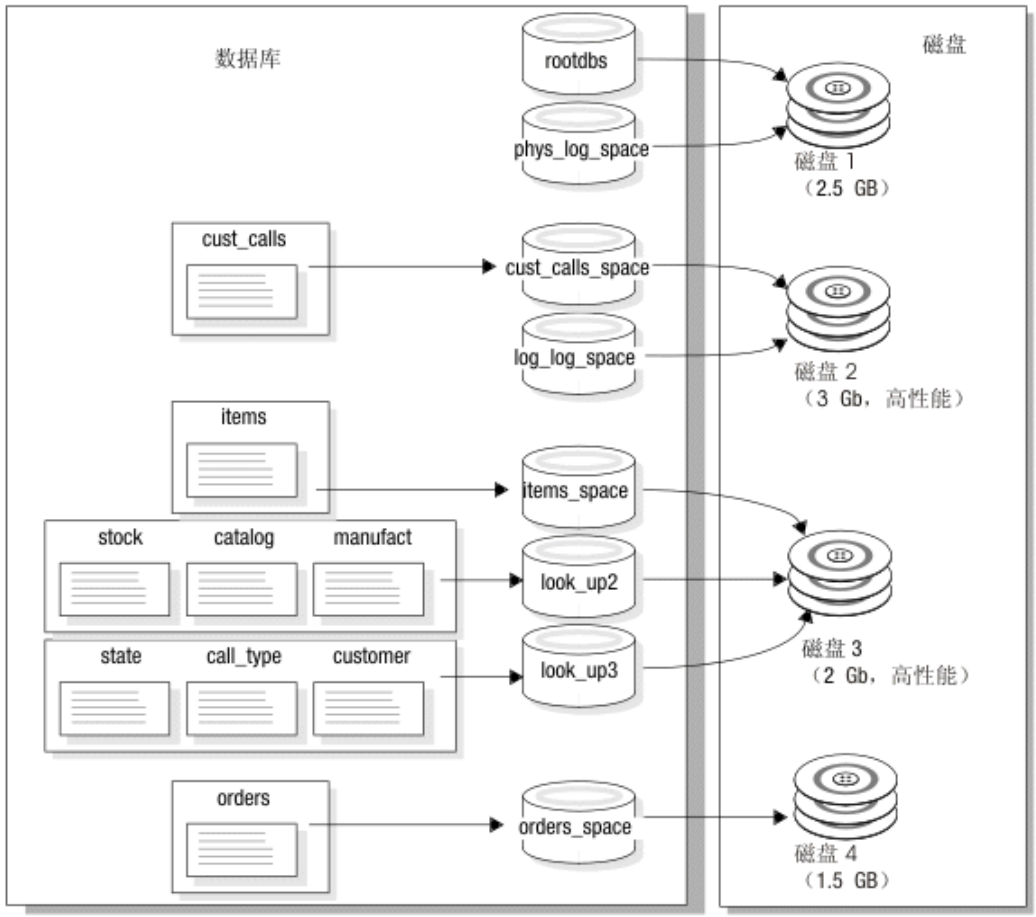
- 将逻辑日志从 rootdbs 数据库空间迁移到独立磁盘上的数据库空间

  此策略将逻辑日志与物理日志分开，从而减少了对根数据库空间的争用。

- 在独立磁盘上的数据库空间中经历最高使用率的两个表的位置

  这些磁盘既不存储逻辑日志也不存储物理日志。理论上，可以将 items 和 orders 表分别存储在不同的高性能磁盘上。然而，在当前的方案中，此策略无法使用，因为需要其中一个高性能磁盘来存储超大的 cust_calls 表（其他两个磁盘对于此任务来说过小）。

图: 针对性能优化的磁盘布局

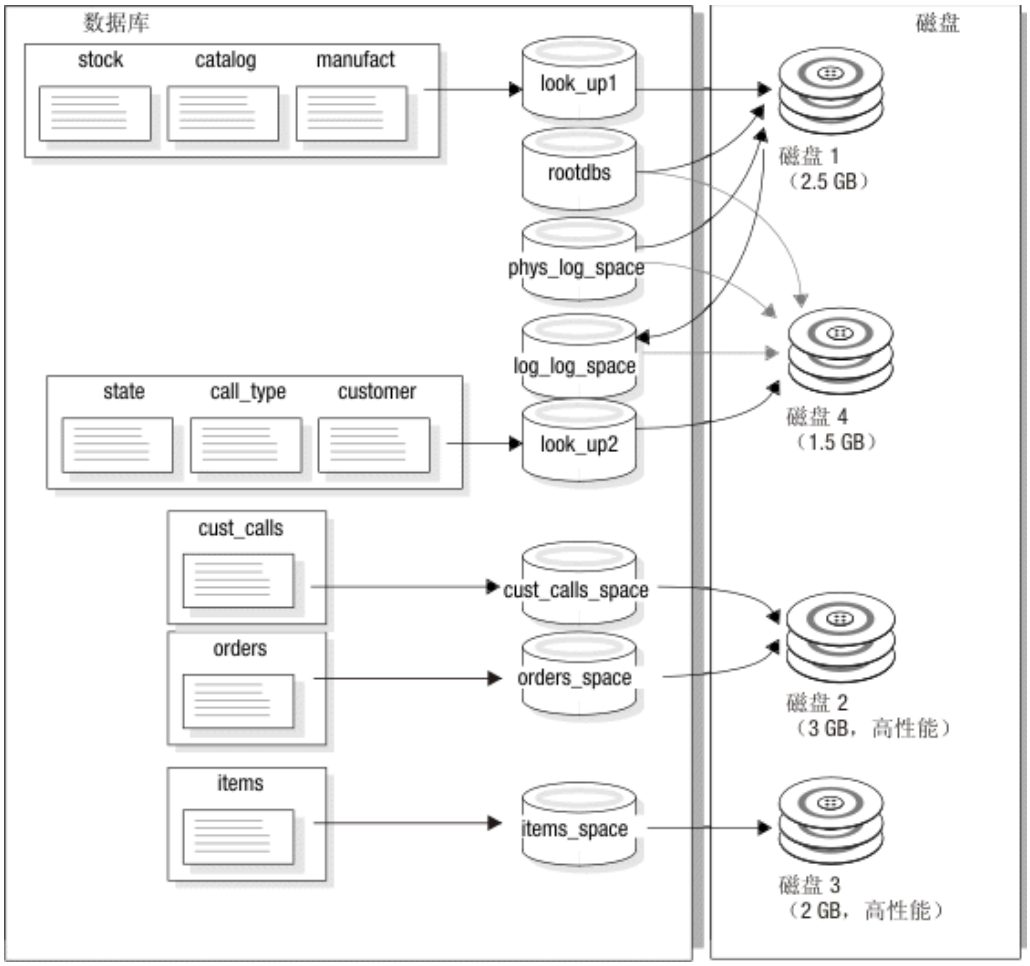


可用性优先级最高时的样本布局

上一种磁盘布局的缺点在于如果磁盘 1 或磁盘 2 发生故障，那么整个数据库服务器将停止运行直到您将这些磁盘上的数据库空间从备份中复原为止。换句话说，该磁盘布局对于可用性来说是不适合的。

下图中显示了一种可以针对可用性优化并涉及镜像的备用磁盘布局。此布局将所有关键数据空间（系统目录表、物理日志和逻辑日志）镜像到单独的磁盘。理论上，可以将逻辑日志和物理日志分开（与上一种布局一样），并且将每个磁盘镜像到其自身的镜像磁盘上。然而，在此方案中，所需的磁盘数不存在；因此逻辑日志和物理日志都位于根数据库空间中。

图：针对可用性优化的磁盘布局



3.5.21 逻辑卷管理器

可以使用逻辑卷管理器 (LVM) 实用程序通过用户定义的逻辑卷来管理磁盘空间。

许多计算机制造商提供带专有 LVM 的计算机。可使用数据库服务器在由大多数专有 LVM 管理的磁盘上存储和检索数据。本节的剩余部分将对逻辑卷管理器的优势和劣势进行说明。

大多数 LVM 可管理多个千兆字节的磁盘空间。 数据库服务器块限制到 4 TB 大小，且此大小只能在正被分配的块偏移量为零时获取。因此，必须限制所有要分配为块的卷的大小（每个块 4 TB）。

由于可使用 LVM 将磁盘驱动器分区成多个卷，因此可以控制将数据放置在给定磁盘的哪个位置。可以通过定义由磁盘驱动器的最中间的柱面组成的卷并在该卷种放置使用率高的表来提高性能。（从技术上讲，您不能将表直接放置在卷中。您必须先将块分配为卷，然后将块指定到数据库空间，最后将表放置在该数据库空间中。有关更多信息，请参阅控制简单大对象数据的存储位置。

提示： 如果选择使用大磁盘驱动器，那么可以将块指定给一个驱动器并且不必将该磁盘分区。

还可以通过使用逻辑卷管理器定义在多个磁盘上展开的卷并随后将表放置到该卷中来提高性能。

许多逻辑卷管理器还允许标准操作系统格式实用程序所不允许的灵活性程度。其中一个特征是能够在您定义了逻辑卷后重新将其定位。这样，与操作系统格式实用程序相比，初次获取磁盘空间布局就不那么重要了。

LVM 经常提供操作系统级别的镜像工具。有关更多信息，请参阅 镜像备用方法。

3.6 管理磁盘空间

可以使用多个实用程序和工具来管理数据库服务器控制的磁盘空间和数据。

这些主题假设您熟悉数据存储中包含的术语和概念。

您可使用以下实用程序来管理存储空间：

- gspaces 实用程序命令
- OAT
- Server Administrator

有关使用 SQL 管理 API 命令（而不是一些 gspaces 命令）的信息，请参阅使用 SQL 管理 API 执行远程管理和《GBase 8s 管理员参考》。

可以生成 SQL 管理 API 或 **gspaces** 命令，用于再现文件中存在的存储空间、块和日志。使用 **dbschema** 实用程序可执行此操作。

3.6.1 分配磁盘空间

本节说明如何分配数据库服务器的磁盘空间。请在分配磁盘空间之前阅读下列章节：

- UNIX 上的未缓冲或已缓冲磁盘的访问
- 存储数据所需的磁盘空间量
- 磁盘布局准则

在您可以创建存储空间或块、或对现有存储空间制作镜像之前，必须为块文件分配磁盘空间。您可以为数据库服务器磁盘空间分配空文件或部分原始磁盘。

仅限 UNIX：在 UNIX™ 上，如果分配原始磁盘空间，必须使用 UNIX ™ ln 命令来创建字符专用设备名称和另一个文件名之间的链接。有关此主题的更多信息，请参阅创建到原始设备的符号链接 (UNIX)。

将 UNIX 文件及其固有操作系统接口用于数据库服务器磁盘空间称为使用*熟空间*。

您可以在磁盘和控制器上平衡块。在单个磁盘上放置多个块可提高吞吐量。

指定偏移量

当您将磁盘空间块分配至数据库服务器时，为达到以下两个目的之一，请指定偏移量：

- 防止数据库服务器覆盖分区信息
- 在分区、磁盘设备或熟文件上定义多个块

偏移量的最大值为 4 太字节。

许多计算机系统和一些磁盘驱动器的制造商会将有关物理磁盘驱动器的信息保存在驱动器本身中。此信息有时称为*卷目录* (VTOC) 或磁盘标签。VTOC 通常存储在驱动器的第一个磁道上。备用扇区及坏扇区映射表（又称为*重指向表*）也可能存储在第一个磁道上。

如果计划从磁盘起始处分配分区，那么可能需要使用偏移量来防止数据库服务器覆盖操作系统所需的关键信息。有关所需的精确偏移量，请参阅磁盘驱动器手册。

重要： 如果您正运行数据库服务器的两个或两个以上实例，那么要非常小心，不要定义重叠的块。重叠的块会导致数据库服务器用不相关的重叠块中数据覆盖另一个块中的数据。此覆盖实际上会删除重叠数据。

为根数据库空间的初始块指定偏移量

对于根数据库空间的初始块及其镜像（如果镜像存在），分别用 ROOTOFFSET 和 MIRROROFFSET 参数指定偏移量。有关更多信息，请参阅《GBase 8s 管理员参考》中有关配置参数的主题。

为附加块指定偏移量

要为数据库服务器空间的附加块指定偏移量，必须在为数据库服务器分配空间时将偏移量作为参数而提供。有关更多信息，请参阅创建使用缺省页大小的数据库空间。

使用偏移量创建多个块

通过指定偏移量以及分配小于可用总空间的块，您可从磁盘分区、磁盘设备或文件中创建多个块。偏移量指定块的开始位置。数据库服务器通过将块大小增加至偏移量来确定块的最后字节的位置。

对于第一个块，分配任何初始偏移量（如有需要）并将大小指定为少于分配磁盘空间总大小的数量。对于每个附加块，请指定偏移量以包含所有先前分配的块并加上初始偏移量，然后分配少于或等于分配中剩余的空间量大小。

在 UNIX 上分配熟文件空间

以下过程显示了在 UNIX™ 上为名为 usr/data/my_chunk 的熟文件分配磁盘空间的示例。

要分配熟文件空间，请执行以下操作：

1. 以用户 **gbasedbt** 身份登录：su gbasedbt
2. 将目录切换至将熟空间将位于的目录：cd /usr/data
3. 通过将 null 连接至数据库服务器要用于磁盘空间的文件名来创建块：cat /dev/null > my_chunk
4. 将文件许可权设置为 660 (rw-rw----)：chmod 660 my_chunk
5. 必须将组和文件所有者均设置到 **gbasedbt**：

```
ls -l my_chunk   -rw-rw----
1   gbasedbt     gbasedbt
0   Oct 12 13:43 my_chunk
```

6. 使用 gspaces 创建存储空间或块。

有关如何使用已分配的文件来创建存储空间的信息，请参阅创建使用缺省页大小的数据库空间、创建 BLOB 空间和创建智能大对象空间。

在 UNIX 上分配原始磁盘空间

要分配原始空间，您必须具有专用于未加工空间的可用磁盘分区。要创建原始磁盘空间，您可以将磁盘重新分区或卸载现有文件系统。在卸载设备之前请备份所有文件。

分配原始磁盘空间

1. 创建并安装原始设备。

   有关如何在 UNIX™ 上分配原始磁盘空间的特定指示信息，请参阅您的操作系统文档和 UNIX 上的未缓冲或已缓冲磁盘的访问。

2. 将字符专用设备的所有权和许可权更改到 **gbasedbt**。

   字符专用设备的文件名通常以字母 r 开头。有关过程，请参阅在 UNIX 上分配熟文件空间中的步骤 4 和步骤 5。

3. 验证指定特征的设备的操作系统许可权是否为 `crw-rw----`。
4. 使用 UNIX 链接命令 ln -s 创建字符专用设备名称和另一个文件名之间的符号链接。

有关详细信息，请参阅创建到原始设备的符号链接 (UNIX)。

限制： 在您创建数据库服务器用于磁盘空间的原始设备之后，请不要在您为数据库服务器磁盘空间分配的同一原始设备上创建文件系统。并且，也不要将同一原始设备用作您为数据库服务器磁盘空间分配的交换空间。

创建到原始设备的符号链接 (UNIX)

使用符号链接来分配标准设备名称并指向设备。要在字符专用设备名和另一个文件名之间创建链接，请使用 UNIX™ 链接命令（通常为 ln）。要验证设备和链接是否都存在，请在设备目录下运行 UNIX 命令 ls -l（DBS 上的 ls -lg）。以下示例显示原始设备的链接。如果您的操作系统不支持符号链接，那么硬链接也可起作用。

```
ln -s /dev/rxy0h /dev/my_root # orig_device link to symbolic_name
ln -s /dev/rxy0a /dev/raw_dev2
ls -l
crw-rw---    /dev/rxy0h
crw-rw---    /dev/rxy0a
lrwxrwxrwx /dev/my_root@->/dev/rxy0h
lrwxrwxrwx /dev/raw_dev2@->/dev/rxy0a
```

为何使用符号链接？如果是在原始设备上创建块，但该设备发生了故障，那么直到替换该原始设备并使用相同路径名之后才能从备份复原。在上次备份时可访问的所有块在您执行复原时均必须为可访问的。

符号链接简化了磁盘故障的恢复，并使您可迅速替换块所处的磁盘。您可以用其他设备替换故障设备，将新设备路径名链接至先前为故障设备创建的同一文件名，然后复原数据。无需等待原始设备修复。

3.6.2 指定存储空间和块的名称

块名称与存储空间名称遵循相同的规则。 如下所示，为存储空间或块指定明确的路径名：

- 如果要在 UNIX™ 上使用原始磁盘，那么必须使用已链接的路径名。（请参阅创建到原始设备的符号链接 (UNIX)。）
- 如果要为数据库服务器磁盘空间使用文件，那么路径名为完整的路径和文件名。

在您创建存储空间或添加块时请使用这些命名规则。文件名必须具有以下特征：

- 唯一，且不超过 128 字节
- 以字母或下划线开头
- 仅包含字母、数字、下划线或 $ 字符

除非名称用引号括起，否则名称不区分大小写。在缺省情况下，数据库服务器将该名称中的大写字符转换为小写。如果希望在名称中使用大写，请用引号将其括起，并将 DELIMIDENT 环境变量设置为 ON。

指定块的最大大小

在大多数平台上，最大块大小为 4 TB；但在其他平台上，最大块大小为 8 TB。

要确定平台所支持的块大小，请参阅机器说明文件。

指定块和存储空间的最大数量

您可在数据库服务器系统上指定存储空间的最大块数为 32,766 以及最大存储空间数为 32,766。

存储空间可以是数据库空间、BLOB 空间和智能大对象空间的任何组合。

考虑到对数据库服务器实例大小的所做的所有限制，实例的最大大小为 8 拍字节。

更改物理模式后备份

您必须对根数据库空间和修改过的存储空间执行 0 级备份，从而确保您可在执行以下操作时复原数据：

- 添加或删除镜像
- 删除逻辑日志文件
- 更改物理日志的大小或位置
- 更改您的存储管理器配置
- 添加、移动或删除数据库空间、BLOB 空间或智能大对象空间
- 对数据库空间、BLOB 空间或智能大对象空间添加、移动或删除块

重要： 添加新的逻辑日志时，不再需要为使用新逻辑日志而对根数据库空间和修改过的数据库空间执行 0 级备份。但必须执行 0 级备份以防止 1 级和 2 级备份失败。

您必须对修改过的存储空间执行 0 级备份，以确保执行以下操作时在转换到日志记录表类型之前可以复原未记录的数据：

- 当您将非日志记录数据库转换为日志记录数据库。
- 当您将 RAW 表转换为标准表

3.6.3 监视存储空间

您可以监视存储空间状态，并配置当存储空间变满时如何给予通知。

当存储空间或分区变满时，会通过联机消息日志文件显示消息。

您可以通过 STORAGE_FULL_ALARM 配置参数配置当存储空间变满时要触发的警报。您可以指定发送警报的频率和要发送警报的最低严重性级别。 缺省情况下，警报时间间隔是 600 秒，警报严重性级别为 3。有关 STORAGE_FULL_ALARM 配置参数和事件警报的更多信息，请参阅《GBase 8s 管理员参考》。

如果高可用性集群中的主服务器遇到了空间不足的情况，并且启用了 STORAGE_FULL_ALARM 配置参数，那么将触发事件警报，并在主服务器上返回错误状态，但是不会在任何辅助服务器上返回错误状态。这是预期的行为，因为主服务器遇到空间不足情况时，不会再将日志记录从主服务器发送到辅助服务器。在这种情况下，辅助服务器永远不会超过其存储限制，因此不会触发事件警报或返回错误状态。

您可以使用 GBase 8s 调度程序设置自动监视存储空间状态这一任务。 任务的属性定义调度程序收集的信息，并指定任务运行的频率。例如，可以将任务定义为一周五天、每隔一小时监视存储空间。有关更多信息，请参阅调度程序和创建任务。

3.6.4 管理数据库空间

这部分包含有关创建具有和不具有缺省页大小的标准和临时数据库空间的信息，创建数据库空间时在数据库空间中为表空间 **tblspace** 指定第一个和下一个扩展数据块大小的信息，以及向数据库空间或 BLOB 空间添加块的信息。

有关监视数据库空间的信息，请参阅监视存储空间。

创建使用缺省页大小的数据库空间

可以使用 gspaces 或 ON-Monitor 创建标准数据库空间和临时数据库空间。

有关创建具有非缺省页大小的数据库空间的信息，请参阅创建具有非缺省页大小的数据库空间。

任何新添加的数据库空间及其镜像（如果存在一个镜像）立即可用。如果您使用镜像，那么可在创建数据库空间时对其制作镜像。镜像可立即生效。

要使用 gspaces 创建标准数据库空间：

1. 在 UNIX™ 上，您必须以用户 **gbasedbt** 或 **root** 身份登录来创建数据库空间。
2. 请确保数据库服务器处于联机、管理或静默方式中。
3. 按分配磁盘空间中所述，为数据库空间分配磁盘空间。
4. 要创建数据库空间，请使用 gspaces -c -d 选项。

   KB 是 -s *size* 和 -o offset 选项的缺省单位。 要将 KB 转换为 MB，可将该单位数量乘以 1024（例如，10 MB = 10 * 1024 KB）。

   如果要创建具有非缺省页大小的数据库空间，请参阅创建具有非缺省页大小的数据库空间以获取有关附加 gspaces 选项的信息。

5. 如果不想为数据库空间中的表空间 **tblspace** 指定第一个和下一个扩展数据块大小，请转至 6。

   如果想要为数据库空间中的表空间 tblspace 指定第一个和下一个扩展数据块大小，表空间为表空间 tblspace 指定第一个和下一个扩展数据块大小中的其他信息。

6. 在创建数据库空间之后，必须对根数据库空间和新数据库空间执行 0 级备份。

以下示例显示了如何在 UNIX 上使用原始磁盘空间创建 10 MB 镜像数据库空间 **dbspce1**，对主块和镜像块的偏移量均为 5000 KB。

```
gspaces -c -d dbspce1 -p /dev/raw_dev1 -o 5000 -s 10240 -m /dev/raw_dev2 5000
```

有关使用 gspaces 创建数据库空间的更多信息，请参阅数据库空间和《GBase 8s 管理员参考》中有关 gspaces 的信息。

要使用 ON-Monitor 创建数据库空间 (UNIX)：

1. 选择数据库空间 > 创建选项。
2. 在字段**数据库空间名称**中输入新数据库空间的名称。
3. 如果您希望为初始数据库空间块创建镜像，请在**镜像**字段中输入 Y。否则，输入 N。
4. 如果您正在创建的数据库空间是临时数据库空间，请在**临时**字段输入 Y。 否则，输入 N。
5. 如果要为标准数据库空间指定页大小，请在**页大小**字段中输入大小（以 KB 计）。大小必须为根数据库空间页大小的倍数。有关指定页大小的更多信息，请参阅创建具有非缺省页大小的数据库空间。

   数据库空间内的所有表、索引以及其他对象都将使用指定大小的页。

6. 在主块部分的**完整路径名**字段中输入数据库空间初始主块的完整路径名。
7. 在**偏移量**字段中指定偏移量。
8. 在**大小**字段中输入块的大小（以 KB 计）。
9. 如果要对该数据库空间制作镜像，请在屏幕的镜像块部分输入完整路径名、大小和（可选）偏移量。

有关更多信息，请参阅《GBase 8s 管理员参考》中的 ON-Monitor 主题。

为表空间 tblspace 指定第一个和下一个扩展数据块大小

如果要减少表空间 **tblspace** 扩展数据块数，并降低发生必须将表空间 **tblspace** 扩展数据块放入非主块的情况的频率，可以指定第一个和下一个扩展数据块大小。（*主块*为数据库空间中的初始块。）

可选择性地指定第一个扩展数据块大小、下一个扩展数据块大小，以及第一个和/或下一个扩展数据块大小。如果不为表空间 **tblspace** 指定第一个或下一个扩展数据块大小，那么 GBase 8s 将使用现有的缺省扩展数据块大小。

可使用 TBLTBLFIRST 和 TBLTBLNEXT 配置参数为表空间 **tblspace**指定第一个或下一个扩展数据块大小，该表空间位于在初始化服务器时创建的根数据库空间中。

可使用 gspaces 实用程序为非根数据库表空间中的表空间 **tblspace** 指定第一个和下一个扩展数据块大小。

创建数据库空间时，可以仅指定第一个和下一个扩展数据块大小。创建数据库空间后，不能改变第一个和下一个扩展数据块大小的规范。另外，您还不能指定临时数据库空间、智能大对象空间、BLOB 空间或外部空间的扩展数据块大小。创建数据库空间之后，您将无法改变第一个和下一个扩展数据块大小的规格。

要指定第一个和下一个扩展数据块大小，请执行以下操作：

1. 确定在表空间 **tblspace** 中所需的总页数。

   页数等于表数 + 拆离索引数 + 可能位于数据库空间中的表分段数 + 表空间 tblspace 的一个页面。

2. 计算页数所需的 KB 数。

   此数字将取决于系统上的一个页面的 KB 数。

3. 通过考虑在创建数据库空间期间分配的表空间 **tblspace** 的所有扩展数据块的重要性，以及是否必须连续分配这些扩展数据块来确定系统上的空间管理需求。

   其中，最重要的就是第一个扩展数据块大小必须更大。如果不是很在意辅助块中存在不连 续扩展数据块，那么第一个和下一个扩展数据块大小可以小一些。

4. 如下所示，指定扩展数据块大小：
   - 如果空间需求适用于根数据库空间，那么在 TBLTBLFIRST 配置参数中指定第一个扩展数据块大小并在 TBLTBLNEXT 配置参数中指定下一个扩展数据块大小。 然后初始化数据库服务器实例。
   - 如果空间需求适用于非根数据库空间，那么通过使用 gspaces 实用程序在命令行上指示第一个和下一个扩展数据块大小从而创建数据库空间。

扩展数据块大小必须以 KB 计，并且必须是页大小的倍数。指定第一个和下一个扩展数据块大小时，请遵循以下指南：

扩展数据块的类型	最小大小	最大大小
非根数据库空间中的第一个扩展数据块	相当于 50 页（以 KB 指定）。这是系统缺省值。 例如，对于 2 KB 页面系统，最小长度为 100。	初始块的大小，减去任何系统对象（如保留页、数据库表空间，以及物理和逻辑日志）所需的空间。

扩展数据块的类型	最小大小	最大大小
根数据库空间中的第一个扩展数据块	相当于 250 页（以 KB 指定）。这是系统缺省值。	初始块的大小，减去任何系统对象（如保留页、数据库表空间，以及物理和逻辑日志）所需的空间。
下一个扩展数据块	为系统上磁盘页大小的 4 倍。 在任何类型的数据库空间上，缺省值均为 50 页。	最大块大小减去 3 页。

可使用以下 gspaces 实用程序和**-ef** 和 **-en** 选项为非根数据库表空间中的表空间 **tblspace** 指定第一个和下一个扩展数据块大小：

- 第一个扩展数据块大小：**-ef** *size_in_kbytes*
- 下一个扩展数据块大小：**-en** *size_in_kbytes*

例如，可以指定：

```
gspaces -c -d dbspace1 -p /usr/data/dbspace1 -o 0 -s 1000000 -e 2000 -n 1000
```

可使用 Gcheck -pt 和 gcheck -pT显示表空间 **tblspace** 的第一个和下一个扩展数据块大小。

如果正在使用数据复制并且在主数据库服务器上创建了数据库空间，那么第一个和下一个扩展数据块大小将通过 ADDCHK 日志记录传递到辅助数据库服务器。

有关 gspaces 实用程序、gcheck 命令以及为表空间 **tblspace** 指定第一个和下一个扩展数据块大小的更多信息，请参阅《GBase 8s 管理员参考》。

创建具有非缺省页大小的数据库空间

如果需要的密钥长度比缺省页大小可提供的更长，那么可以指定标准或临时数据库空间的页大小。

根数据库空间为缺省页大小。如果要指定页大小，该大小必须为缺省页大小的整数倍，且不能超过 16 KB。

对于具有足够存储空间的系统，较大页大小的性能优势包括：

- 减少 B 型树索引的深度（甚至对于较小的索引键）。
- 减少检查点时间（通常出现在使用较大页大小时）。

其他的性能优势还体现在您可以：

- 将当前跨多个页（大小为缺省页大小）的长行分组到同一页上。
- 定义临时表的不同页大小，这样临时表具有单独的缓冲池。

使用 BUFFERPOOL 配置参数可创建与数据库空间的页大小相对应的缓冲池。（您可能想要执行此操作来实现一种“专用缓冲池”。）

一个表可在一个数据库空间中，而该表的索引可在另一个数据库空间中。这些分区的页大小可以不同。

如果想要为数据库空间指定页大小，请指向以下任务：

1. 请运行 gadmin -BC 2 命令来启用大块方式。

   缺省情况下，首次初始化或重新启动 GBase 8s 时，将启动 GBase 8s，并启用大块方式。有关 gadmin 实用程序的信息，请参阅《*GBase 8s 管理员参考*》。

2. 创建对应于数据库空间页大小的缓冲池。可使用 glogadmin 实用程序或 BUFFERPOOL 配置参数。必须在创建数据库空间之前执行此操作。

   如果创建的数据库空间其页大小没有对应的缓冲池，那么 GBase 8s 将使用在 onconfig 配置文件中定义的缺省参数自动创建缓冲池。

   无法使多个缓冲池具体相同的页大小。

3. 创建数据库空间时，定义数据库空间的页大小。可使用 gspaces 实用程序或 ON-Monitor。有关更多信息，请参阅定义页大小。

例如，如果创建页大小为 6 KB 的数据库空间，那么必须创建大小为 6 KB 的缓冲池。如果没有为新的缓冲池指定页大小，那么 GBase 8s会将操作系统缺省页大小（在大多数 UNIX™ 平台上为 2 KB）用作缓冲池的缺省页大小。

提示： 如果使用非缺省页大小，可能需要增大物理日志的大小。如果对非缺省页执行许多更新，那么可能需要将物理日志大小增大 150% 到 200%。要调整物理日志，可能需要进行一些试验。可根据物理日志触发器检查点的填充频率来按需调整物理日志的大小。

为非缺省页大小创建缓冲池

创建缓冲池时，可使用 BUFFERPOOL 配置参数或 glogadmin 实用程序来指定有关缓冲区的信息（包括其大小）、指定缓冲池中 LRUS 的数量、指定缓冲池中缓冲区的数量，以及的 **lru_min_dirty** 和 **lru_max_dirty**值。

使用 BUFFERPOOL 配置参数指定此信息。

BUFFERPOOL 配置参数由 onconfig.std 文件中的两行组成。

在 UNIX™ 系统上，这些行是：

```
BUFFERPOOL default,buffers=10000,lrus=8,lru_min_dirty=50.00,lru_max_dirty=60.50
BUFFERPOOL size=2K,buffers=50000,lrus=8,lru_min_dirty=50,lru_max_dirty=60
```

第一行指定了使用的缺省值，这些值是当您创建的数据库空间的页大小没有对应在启动数据库服务器时创建的缓冲池时被使用的。缺省行之后的以下行指定了缓冲池的数据库服务器缺省值。 这些值以数据库服务器的缺省页大小为依据。

使用 gspaces 实用程序添加具有不同页大小的数据库空间时，或者使用 glogadmin 实用程序添加新的缓冲池时，将向 onconfig 文件的 BUFFERPOOL 配置参数添加一个新行。每个缓冲池的页大小必须为您操作系统缺省页大小的倍数。以下示例显示了添加到 onconfig 文件中的第三个 BUFFERPOOL 行：

```
BUFFERPOOL default,buffers=10000,lrus=8,lru_min_dirty=50.00,lru_max_dirty=60.50
BUFFERPOOL size=2K,buffers=50000,lrus=8,lru_min_dirty=50,lru_max_dirty=60
BUFFERPOOL size=6K,buffers=3000,lrus=8,lru_min_dirty=50,lru_max_dirty=60
```

BUFFERPOOL 行中的字段不区分大小写，所以您可以指定 **lrus**、**Lrus** 或 **LRUS**）。这些字段可按任何顺序显示。

如果没有为新的缓冲池指定页大小，那么 GBase 8s会将操作系统缺省页大小用作缓冲池的缺省页大小。

如果 **buffers** 的值为零 (0)，或者在 BUFFERPOOL 配置参数中缺少 **buffers** 的值，那么 GBase 8s 不会创建指定页大小的缓冲池。

重要： 使用BUFFERPOOL 配置参数输入的信息将取代先前使用不推荐的参数指定的任何信息。有关不推荐的参数的更多信息，请参阅附录，其中包含《GBase 8s 管理员参考》中已终止的配置参数的信息。

下表为您使用 BUFFERPOOL 配置参数或 glogadmin 实用程序指定的值提供了说明。

字段	解释	值范围
size	指定页面中后跟后缀 K 的 KB 数。	大小可在 2K 或 4K 至 16 K 之间变化。2K 为缺省值。
buffers	指定页大小缓冲区的数量。 这是数据库服务器用户线程代表客户机应用程序用于磁盘 I/O 的共享内存缓冲区的最大数目。数据库服务器所需的缓冲区数取决于应用程序。 例如，如果数据库服务器用了 90% 的时间访问了 15% 的应用程序数据，那么必须分配足够的缓冲区以保留该 15% 的数据。增加缓冲区数量可提高系统性能。 缓冲区空间所需的物理内存百分比取决于计算机上可用的内存量，以及用于其他应用程序的内存量。对于具有大量可用物理内存（4 千兆字节或以上）的系统，缓冲区空间可达到物理内存的 90%。对于可用物理内存量较小的系统，缓冲区空间的范围可以为物理内存的 20% 到 25%。 必须在设置缓冲区空间 (`buffers * system_page_size`) 后计算所有其他的共享内存参数。	对于 UNIX 上的 32 位平台 • 页大小等于 2048 字节：100 到 1,843,200 个缓冲区 (`1843200 = 1800 * 1024`) • 页大小等于 4096 字节：100 到 921,600 个缓冲区 (`921,600 = ((1800 * 1024)/4096) * 2048`) 对于 64 位平台：100 到 $2^{31}-1$ 个缓冲区（对于 64 位平台的实际值，请参阅发行说明。Solaris 上缓冲区的最 大数为 536,870,912。）
lrus	在共享内存缓冲池中为页大小指定 LRU（最近最少使用的）队列的数目。可调整 LRUS、**lru_min_dirty** 和 **lru_max_dirty** 值以控制共享内存缓冲区每隔多少时间清空到磁盘一次。 将 LRUS 设置得过高可能导致页清除程序活动过多。	1 到 128

字段	解释	值范围
lru_min_dirty	指定已修改的页占 LRU 队列的百分比，达到该百分比时页清除将不再是必需的。在某些情况下，页清除程序可能会超出该点继续清除。 用于清空检查点之间缓冲池的 LRU 值对于检查点性能不是特别重要。**lru_min_dirty** 值通常仅对于维持足够数量的干净页以供替换而言是必要的。通过设置 LRU 清空参数以将 **lru_min_dirty** 设置为 70 来启动。 有关更多信息，请参阅用于清空检查点之间缓冲池的 LRU 值。	0 到 100（允许小数值） 如果超过值的范围指定参数，那么缺省值将被设置为 50.00%。
lru_max_dirty	指定已修改页占 LRU 队列的百分比，达到该百分比时将清除队列。 用于清空检查点之间缓冲池的 LRU 值对于检查点性能不是特别重要。**lru_max_dirty** 值通常仅在维持足够数量的干净页以供替换时才显得必要。通过设置 LRU 清空参数以将 **lru_max_dirty** 设置为 80 来启动。 有关更多信息，请参阅用于清空检查点之间缓冲池的 LRU 值。	0 到 100（允许小数值） 如果超过值的范围指定参数，那么缺省值将被设置为 60.00%。

如果数据库服务器处于联机、静默或管理方式，那么您还可以使用 glogadmin 实用程序添加不同大小的新缓冲池。使用 glogadmin 实用程序时，指定的信息将自动传输到 onconfig 文件的，并且会使用 BUFFERPOOL 关键字指定新值。不能通过编辑onconfig.std 文件更改值。

当您使用 glogadmin 实用程序时，请如下指定信息：

```
glogadmin -b -g <size of buffer page in Kbytes> -n <number of buffers>
-r <number of LRUs> -x <max dirty (fractional value allowed)>
-m <minimum dirty (fractional value allowed)>
```

例如：

```
glogadmin -b -g 6 -n 3000 -r 2 -x 2.0 -m 1.0
```

这添加了 3000 个大小为 6K 字节的缓冲区，每个缓冲区都具体 2 个 LRUS 并且 **lru_max_dirty** 被设置为 2%，而且**lru_min_dirty** 被设置为 1%。

有关 glogadmin 实用程序的更多信息，请参阅《GBase 8s 管理员参考》。

建议： 将 PHYSBUFF 配置参数至少设置为 128 KB。如果数据库服务器已配置为使用 RTO_SERVER_RESTART，请将 PHYSBUFF 配置参数至少设置为 512 KB。将 PHYSBUFF 设置为较低的值可能会影响事务性能以及/或者导致服务器初始化期间出现性能警告。

LG_ADDBPOOL 日志记录和 **sysbufpool** 系统目录表包含有关每个缓冲池的信息。

当数据库服务器正在运行时添加的缓冲池将会进入虚拟内存而不是常驻内存中。根据正在使用的内存的可用性，仅在启动时在 onconfig 文件中指定的那些缓冲池条目会进入常驻内存。

调整现有缓冲池大小

如果必须调整现有缓冲池的大小，必须关闭数据库服务器。然后在 onconfig 文件中更改缓冲池的大小。

删除现有缓冲池

如果必须删除现有缓冲池，必须关闭数据库服务器。然后在 onconfig 文件中删除缓冲池。

定义页大小

可以使用 gspaces -k 选项或 ON-Monitor 以定义页大小。

如下所示设置页大小（以 KB 为单位）：

gspaces -c -d *DBspace* [-t] [-k *pagesize*] -p *path* -o *offset* -s *size* [-m *path offset*]

根数据库空间为缺省页大小。

如果指定页大小，页大小必须是缺省页大小的倍数，且不能超过 16 KB。

如果使用 ON-Monitor 创建数据库空间，请确保在**页大小**字段中输入页大小（以 KB 计）。

通过使用直接 I/O 提高熟文件数据库空间的性能

在 UNIX™ 系统上，可通过使用直接 I/O 提高用于数据库空间块的熟文件的性能。

直接 I/O 必须可用并且文件系统必须支持用于数据库空间块的页大小的直接 I/O。

可以使用 GBase 8s 将原始设备或熟文件用于数据库空间块。通常，熟文件会较慢，因为存在文件系统提供的额外开销和缓存。直接 I/O 绕过文件系统缓冲区的使用，因此对磁盘的读写效率更高。使用 DIRECT_IO 配置参数指定直接 I/O。如果文件系统支持用于数据库空间块的页大小的直接 I/O 并使用直接 I/O，那么熟文件的性能可能接近用于数据库空间块的原始设备的性能。

要通过使用直接 I/O 提高熟文件数据库空间的性能，请执行以下操作：

1. 验证您是否具有直接 I/O 且文件系统是否对用于数据库空间块的页大小支持直接 I/O。
2. 通过将 DIRECT_IO 配置参数设置为 1 启用直接 I/O。

如果有 AIX® 操作系统，那么还可以为 GBase 8s 启用并行 I/O，以在对使用熟文件的块执行读写时将其与直接 IO 一起使用。

将多个指定分段存储在单个数据库空间

对于使用基于表达式的分布方案或循环分布方案的分段表，可以创建可位于单个数据库空间中的指定分段。

对于使用基于表达式、时间间隔、列表或循环法的分布方案的分段表，可以创建可位于单个数据库空间中的指定分段。

将多个表或索引段存储到一个数据库空间中时的查询性能要优于将每个段存储到不同数据库空间中时的查询性能，并且简化了数据库空间的管理。

假设您要使用基于表达式的分布方案创建分段表，该分布方案中的每个表达式都会指定放置在特定分段中的数据集。您可决定使用某个数据库空间中某个月的 数据以及其他的 11 个数据库空间中的 11 个月的数据分割表中的数据。然而，如果要对所有年度数据仅使用一个数据库空间，那么可以创建指定分段，使每个月的数据存储在一个数据库空间中。

如果创建具有指定分段的分段表，**sysfragments** 系统目录中每行的**分区**列中都会包含分段名称。如果创建没有指定分段的分段表，那么数据库空间的名称位于**分区**列中。**sysfragments** 目录中的**标志**列告诉您分段存储方案中是否具有指定分段。

可以创建具有指定分段的表和索引，并且可以使用 PARTITION 关键字和分段名称创建、删除和修改指定分段。

要创建具有指定分段的分段表，请如以下示例中所示使用 SQL 语法：

```
CREATE TABLE tb1(a int)
    FRAGMENT BY EXPRESSION
      PARTITION part1 (a >=0 AND a < 5) IN dbspace1,
      PARTITION part2 (a >=5 AND a < 10) IN dbspace1
            ...
        ;
```

如果创建的表或索引分段包含指定分段，那么在使用 ALTER FRAGMENT 语句时必须使用包含指定分段名称的语法，如以下示例中所示：

```
ALTER FRAGMENT ON TABLE tb1 INIT FRAGMENT BY EXPRESSION
     PARTITION part_1 (a >=0 AND a < 5) IN dbspace1,
     PARTITION part_2 (a >=5 AND a < 10) IN dbspace1;
ALTER FRAGMENT ON INDEX ind1 INIT FRAGMENT BY EXPRESSION
     PARTITION part_1 (a >=0 AND a < 5) IN dbspace1,
     PARTITION part_2 (a >=5 AND a < 10) IN dbspace1;
```

按此示例中所示，可使用 PARTITION BY EXPRESSION 子句替代 CREATE TABLE、CREATE INDEX 和 ALTER FRAGMENT ON INDEX 语句中的 FRAGMENT BY EXPRESSION 子句：

```
ALTER FRAGMENT ON INDEX idx1 INIT PARTITION BY EXPRESSION
                PARTITION part1     (a <= 10) IN idxdbspc1,
                PARTITION part2     (a <= 20) IN idxdbspc1,
                PARTITION part3    (a <= 30) IN idxdbspc1;
```

使用 ALTER FRAGMENT 语法将不具有指定分段的分段表和索引更改为具有指定分段的表和索引。以下语法显示了如何可将具有多个数据库空间的分段表转换为具有指定分段的分段表：

```
CREATE TABLE t1 (c1 int) FRAGMENT BY EXPRESSION
      (c1=10) IN dbs1,
      (c1=20) IN dbs2;
ALTER FRAGMENT ON TABLE t1 MODIFY dbs2 TO PARTITION part_3 (c1=20)
 IN dbs1
```

以下语法显示了如何可将分段索引转换为包含指定分段的索引：

```
CREATE TABLE t1 (c1 int) FRAGMENT BY EXPRESSION
            (c1=10) IN dbs1, (c1=20) IN dbs2, (c1=30) IN dbs3
CREATE INDEX ind1 ON t1 (c1) FRAGMENT BY EXPRESSION
            (c1=10) IN dbs1, (c1=20) IN dbs2, (c1=30) IN dbs3
ALTER FRAGMENT ON INDEX ind1 INIT FRAGMENT BY EXPRESSION
            PARTITION part_1 (c1=10) IN dbs1, PARTITION part_2 (c1=20) IN dbs1,
            PARTITION part_3 (c1=30) IN dbs1,
```

请参阅《GBase 8s SQL 指南：语法》以获取更多的语法详细信息，包括 GRANT FRAGMENT 和 REVOKE FRAGMENT 语句中有关指定分段的信息以及有关使用 ALTER FRAGMENT 语句的 DROP、DETACH 和 MODIFY 子句的详细信息。

创建临时数据库空间

要指定分配临时文件的位置，请创建临时数据库空间。

要定义临时数据库空间，请执行以下操作：

1. 使用带 **-c -d -t** 选项的 gspaces 实用程序。

   有关更多信息，请参阅创建使用缺省页大小的数据库空间。

2. 使用 DBSPACETEMP 环境变量或配置参数可指定数据库服务器可用于临时存储器的数据库空间。

   DBSPACETEMP 配置参数可包含非缺省页大小的数据库空间。虽然可在 DBSPACETEMP 的参数列表中包含页大小不同的各个数据库空间，但数据库服务器只会使用页大小和所列出的第一个数据库空间的页大小相同的数据库空间。

   有关 DBSPACETEMP 的更多信息，请参阅 *GBase 8s 管理员参考* 中有关配置参数的主题。

3. 如果创建了多个临时数据库空间，那么这些数据库空间必须位于不同磁盘上以优化 I/O。

如果要创建临时数据库空间，那么必须通过设置 DBSPACETEMP 配置参数和/或 DBSPACETEMP 环境变量让数据库服务器知道存在新创建的临时数据库空间。数据库服务器直至您采取以下两个步骤之后才开始使用临时数据库空间：

- 设置 DBSPACETEMP 配置参数和/或 DBSPACETEMP 环境变量。
- 重新启动数据库服务器。

以下示例显示了如何创建名为 **temp_space** 的 5 MB 临时数据库空间，偏移量为 5000 KB：

```
gspaces -c -t -d temp_space -p /dev/raw_dev1 -o 5000 -s 5120
```

有关更多信息，请参阅临时数据库空间。

磁盘空间不足时应执行的操作

当您正在创建的数据库空间的初始块是熟文件（在 UNIX™ 上）时，数据库服务器会验证磁盘空间是否足够用于该初始块。如果块大小大于磁盘上的可用空间，将显示消息且不会创建任何数据库空间。但是，未除去数据库服务器为初始块创建的熟文件。它的大小表示在您创建数据库空间之前文件系统上剩余的空间。 除去该文件以回收该空间。

向数据库空间或 BLOB 空间添加块

当数据库空间、BLOB 空间或智能大对象空间将满或需要更多磁盘空间时添加块。

重要： 新添加的块及其相关联的镜像（如果存在一个镜像）立即可用。如果您正向已镜像的存储空间添加块，那么必须还添加镜像块。

要使用 gspaces 添加块，请执行以下操作：

1. 在 UNIX™ 上，必须以用户 **gbasedbt** 或 **root** 身份登录来添加块。
2. 请确保数据库服务器处于联机、管理或静默方式，或快速恢复方式的清除阶段。
3. 按分配磁盘空间中所述，为块分配磁盘空间。
4. 要添加块，请使用 gspaces 的 **-a** 选项。
   如果存储空间已镜像，那么必须为主块和镜像块都指定路径名。
   如果指定了不正确的路径名、偏移量或大小，那么数据库服务器不会创建块，并且会显示错误消息。另见磁盘空间不足时应执行的操作。
5. 在创建块之后，必须对根数据库空间和包含该块的数据库空间、BLOB 空间或智能大对象空间执行 0 级备份。

以下示例将 10 兆字节的镜像块添加至 **blobsp3**。对主块和镜像块都指定了 200 KB 的偏移量。如果您不添加镜像块，那么可省略 **-m** 选项。

```
gspaces -a blobsp3 -p /dev/raw_dev1 -o 200 -s 10240 -m /dev/raw_dev2 200
```

下一个示例以 5200 KB 的偏移量向数据库空间 **dbspc3** 添加 5 MB 的原始磁盘空间块。

```
gspaces -a dbspc3 \\.\e: -o 5200 -s 5120
```

还可以定义当应用程序需要更多存储空间时，GBase 8s 可用于自动扩展块大小的信息。如果有可扩展块，那么无需添加新块或耗费时间来尝试确定哪种存储空间（数据库空间、临时数据库空间、智能大对象空间、临时智能大对象空间或 BLOB 空间）将耗尽空间，以及何时将耗尽空间。

当启用 CHUNK_OVERLAP_PROTECTION 配置参数时，数据库服务器将确保新块不会与其他块重叠。如果服务器发现块重叠，新块添加操作将失败。

使用 ON-Monitor 添加块 (UNIX™)

可以使用 ON-Monitor 将块添加到数据库空间。

要向数据库空间添加块，请遵循以下指示信息：

1. 选择添加块 > 数据库空间选项。
2. 使用 Enter 键或方向键来选择将接收新块的 BLOB 空间或数据库空间，并按 CTRL-B 或 F3 键。
3. 下一屏幕指示是否对 BLOB 空间或数据库空间制作镜像。如果是，那么在**镜像**字段中输入 Y。
4. 如果您要向其中添加块的数据库空间是临时数据库空间，请在**临时**字段中输入 Y。
5. 如果您指示对数据库空间或 BLOB 空间制作镜像，那么主块和镜像块都必须要指定。
   在主块部分的**完整路径名**字段中输入新主块的完整路径名。
6. 在**偏移量**字段中指定偏移量。
7. 在**大小**字段中输入块的大小（以 KB 计）。
8. 如果要对该块制作镜像，请在屏幕的镜像块部分中输入完整路径名、大小和（可选）偏移量。

重命名数据库空间

如果您是用户 gbasedbt 或者具有 DBA 特权，并且数据库服务器处于静默方式（不是其他方式），那么可使用 **gspaces** 实用程序来重命名数据库空间。

要重命名数据库空间，请使用以下 gspaces 实用程序命令：

gspaces -ren *old_dbspace_name*-n *new_dbspace_name*

可重命名标准数据库空间和所有其他空间，包括 BLOB 空间、智能 BLOB 空间、临时空间和外部空间。然而，您无法重命名任何关键数据库空间，比如根数据库空间或包含物理日志的数据库空间。

可重命名数据库空间和智能大对象空间：

- 启用了 Enterprise Replication 时
- 启动数据复制时，在主数据库服务器上

无法在辅助数据库服务器上或者当辅助数据库服务器为 Enterprise Replication 配置的一部分时重命名数据库空间和智能大对象空间

重命名数据库空间操作仅更改数据库空间名称；它不会重新组织数据。

重命名数据库空间命名更新所有存储该名称位置处的数据库空间名称。这包含磁盘上 的保留页、系统目录、ONCONFIG 配置文件和内存内数据结构。

重要： 重命名数据库空间后，对重命名的根数据库空间和根数据库空间执行 0 级归档。有关信息，请参阅《GBase 8s 备份与复原指南》。

重命名数据库空间之后可能需要执行的其他操作

如果重命名数据库空间，必须重新写入和重新编译引用旧数据库空间名称的所有存储过程代码。 例如，如果您的存储过程包含 ALTER FRAGMENT 关键字并引用了数据库空间名称，那么必须重新写入和重新编译该存储过程。

如果您重命名在 DATASKIP 配置参数中指定的数据库空间，那么您必须在重命名数据库空间后手动更新 DATASKIP 配置参数。

3.6.5 管理 BLOB 空间

本节说明如何创建 BLOB 空间和确定 BLOB 页大小。数据库服务器将 TEXT 和 BYTE 数据存储在数据库空间或 BLOB 空间中，但 BLOB 空间更有效率。有关添加块的信息，请参阅向数据库空间或 BLOB 空间添加块。

有关监视 BLOB 空间的信息，请参阅监视存储空间

创建 BLOB 空间

可以使用 gspaces 或 ON-Monitor 创建 BLOB 空间。

创建 BLOB 空间之前：

1. 按分配磁盘空间中所述，为 BLOB 空间分配磁盘空间。
2. 确定对您环境最佳的 BLOB 页大小。
   有关指示信息，请参阅确定 BLOB 页大小。

指定最多为 128 个字节的 BLOB 空间名称。该名称必须是唯一的，并且必须以字母或下划线开头。您可以在名称中使用字母、数字、下划线和 $ 字符。

重要： 如果数据库服务器启用了镜像，那么您可在创建 BLOB 空间时对其制作镜像。镜像可立即生效。

要使用 gspaces 创建 BLOB 空间，请执行以下操作：

1. 要在 UNIX™ 上创建 BLOB 空间，您必须以用户 **gbasedbt** 或 **root** 身份登录。
2. 请确保数据库服务器处于联机、管理或静默方式，或快速恢复方式的清除阶段。
3. 要添加 BLOB 空间，请使用 gspaces -c -b 选项。
   a. 为 BLOB 空间指定明确的路径名。如果已对 BLOB 空间制作镜像，那么必须为主块和镜像块都指定路径名和大小。
   b. 使用 **-o** 选项为 BLOB 空间指定偏移量。
   c. 使用 **-s** 选项指定 BLOB 空间块的大小（以 KB 计）。
   d. 使用 **-g** 选项根据每个 BLOB 页的磁盘页数指定 BLOB 页大小。

   请参阅确定 BLOB 页大小。例如，如果数据库服务器实例的磁盘页大小为 2 KB，而您需要大小为 10 KB 的 BLOB 页，那么在该字段中输入 5。

   如果指定了不正确的路径名、偏移量或大小，那么数据库服务器不会创建 BLOB 空间，并且会显示错误消息。另见磁盘空间不足时应执行的操作。
4. 在创建 BLOB 空间之后，必须对根数据库空间和新 BLOB 空间执行 0 级备份。

以下示例显示了如何在数据库服务器页大小为 2 KB 时，创建 BLOB 页大小为 10 KB 的 10 MB 镜像 BLOB 空间 **blobsp3**。为主块和镜像块指定偏移量 200 KB。BLOB 空间从 UNIX 上的原始磁盘空间创建。

```
gspaces -c -b blobsp3 -g 5 -p /dev/raw_dev1 -o 200 -s 10240 -m /dev/raw_dev2 200
```

有关使用 gspaces 创建 BLOB 空间的参考信息，请参阅 *GBase 8s 管理员参考* 中有关 gspaces 实用程序的信息。

要使用 ON-Monitor 创建 BLOB 空间 (UNIX)，请执行以下操作：

1. 选择数据库空间 > BLOB 空间选项。
2. 在 **BLOB 空间名称**字段中输入新 BLOB 空间的名称。
3. 如果您希望为初始 BLOB 空间块创建镜像，请在**镜像**字段中输入 Y。 否则，输入 N。
4. 根据每个 BLOB 页的磁盘页数，在 **BLOB 页大小**字段中指定 BLOB 页大小。

   请参阅确定数据库服务器页大小。例如，如果数据库服务器实例的磁盘页大小为 2 KB，而您需要大小为 10 KB 的 BLOB 页，那么在该字段中输入 5。
5. 在主块部分的**完整路径名**字段中输入 BLOB 空间初始主块的完整路径名。
6. 在**偏移量**字段中指定偏移量。
7. 在**大小**字段中输入块的大小（以 KB 计）。
8. 如果要对该 BLOB 空间制作镜像，请在屏幕的镜像块部分中输入完整路径名、大小和（可选）偏移量。

准备 BLOB 空间以存储 TEXT 和 BYTE 数据

新创建的 BLOB 空间不能立即用于存储 TEXT 或 BYTE 数据。BLOB 空间日志记录和恢复需要在不同的逻辑日志文件中创建用于创建 BLOB 空间的语句，以及用于将 TEXT 和 BYTE 数据插入该 BLOB 空间的语句。对所有的 BLOB 空间均是如此要求，而无论数据库的日志记录状态。要满足该要求，请在创建 BLOB 空间后切换至下一逻辑日志文件。（有关指示信息，请参阅备份日志文件以释放 BLOB 页。）

确定 BLOB 页大小

在创建 BLOB 空间时，将最常出现的简单大对象的大小用作 BLOB 页的大小。换言之，选择浪费最少空间量的 BLOB 页大小。

如果一个表具有多个 TEXT 或 BYTE 列，并且各个对象的大小并不接近，请将每个列存储在不同的 BLOB 空间中，且每个列都用适当大小的 BLOB 页来存储。

确定数据库服务器页大小

在您指定 BLOB 页大小时，请根据数据库服务器基页大小来指定。

可以使用以下方法之一来确定您系统的数据库服务器页大小：

- 运行 gstat -b 实用程序以显示系统页大小，在输出的最后一行中显示其为缓冲区大小。

- 要查看 PAGE_PZERO 保留页的内容，请运行 gcheck -pr 实用程序。
- **仅限 UNIX™：**在 ON-Monitor 中，选择参数 > 共享内存或参数 > 初始化选项，以显示系统页大小。

获取 BLOB 空间存储统计信息

要帮助确定每个 BLOB 空间的最佳 BLOB 页大小，请使用以下数据库服务器实用程序命令：

- gcheck -pe
- gcheck -pB

gcheck -pe 命令提供有关存储在 BLOB 空间中的对象的背景信息：

- 有关每个有数据存储在 BLOB 空间块中的表的完整的所有权信息（显示为 *database:owner.table*）
- 每个表用于存储与其相关联的 TEXT 和 BYTE 数据的总页数
- BLOB 空间中总的可用页数和总的开销页数

gcheck -pB 命令列出以下有关每个表或数据库的信息：

- 表或数据库在每个 BLOB 空间中使用的 BLOB 页数
- 作为表或数据库的一部分而存储的每个简单大对象所用的 BLOB 页的平均填充度

3.6.6 管理智能大对象空间

本节描述如何创建标准的或临时的智能大对象空间、监视元数据或用户数据区域、向智能大对象空间添加块以及更改智能大对象的存储特征。

有关监视智能大对象空间的信息，请参阅监视存储空间。

创建智能大对象空间

使用 gspaces 实用程序创建智能大对象空间。

使用 gspaces 实用程序或 Server Administrator (ISA) 可创建智能大对象空间。

要使用 gspaces 创建智能大对象空间，请执行以下操作：

1. 要在 UNIX™ 上创建智能大对象空间，必须以用户 **gbasedbt** 或 **root** 身份登录。
2. 请确保数据库服务器处于联机、管理或静默方式，或者处于快速恢复方式的清除阶段。
3. 使用 gspaces -c -S 选项创建智能大对象空间。
   a. 使用 -p 选项指定路径名、使用 -o 选项指定偏移量，以及使用 -s 选项指定智能大对象空间大小。
   b. 如果要对智能大对象空间制作镜像，请使用 -m 选项指定镜像路径和偏移量。
   c. 如果您希望使用智能大对象空间的缺省存储特征，那么省略 -Df 选项。
   如果您希望指定不同的存储特征，请使用 -Df 选项。有关更多信息，请参阅智能大对象空间的存储特征。
   d. 智能大对象空间中的第一个块必须具有元数据区域。

您可以为智能大对象空间指定元数据区域，或让数据库服务器计算元数据区域的大小。有关更多信息，请参阅计算智能大对象空间元数据的大小。

4. 在创建智能大对象空间之后，必须对根数据库空间和新智能大对象空间执行 0 级备份。
5. 要在该智能大对象空间中开始存储智能大对象，请在 SBSPACENAME 配置参数中指定空间名称。
6. 使用 gstat -d、gstat -g smb s 以及 gcheck -cs、-cS、-ps 或 -pS 来显示有关智能大对象空间的信息。

有关更多信息，请参阅监视智能大对象空间。

这显示了如何创建 20 兆字节的镜像智能大对象空间 **sbsp4**。为主块和镜像块均指定 500 KB 的偏移量，还指定偏移量为 200 KB 且大小为 150 KB 的元数据。AVG_LO_SIZE -Df 标记指定 32 KB 的智能大对象平均期望大小。

```
gspaces -c -S sbsp4 -p /dev/rawdev1 -o 500 -s 20480 -m /dev/rawdev2 500
-Ms 150 -Mo 200 -Df "AVG_LO_SIZE=32"
```

有关创建智能大对象空间和智能大对象缺省选项的信息，请参阅《GBase 8s 管理员参考》中有关 gspaces 实用程序的信息。

要使用 ISA 创建智能大对象空间，请执行以下操作：

1. 使用 ISA 创建智能大对象空间。有关更多信息，请参阅 ISA 联机帮助。
2. 备份新智能大对象空间和根数据库空间。

计算智能大对象空间元数据的大小

智能大对象空间的第一个块必须有元数据区域。 在向智能大对象空间添加智能大对象和块时，元数据区域会增大。另外，数据库服务器保留 40% 的用户区域，在元数据区域空间耗尽的情况下使用。

为智能大对象空间正确估算元数据区域以确保智能大对象空间不会耗尽元数据空间，这一点很重要。您可以使用以下方法之一：

- 让数据库服务器为新智能大对象空间块计算元数据区域的大小。
- 明确指定元数据区域的大小。

向智能大对象空间添加块

可以将块添加到智能大对象空间或临时智能大对象空间。

您可以为块指定元数据区域、让数据库服务器计算元数据区域或者将块仅用于用户数据。

要使用 gspaces 将块添加到智能大对象空间，请执行以下操作：

1. 请确保数据库服务器处于联机、管理或静默方式，或者处于快速恢复方式的清除阶段。
2. 使用 gspaces -a 选项创建智能大对象空间块。

a. 使用 **-p** 选项指定路径名、使用 **-o** 选项指定偏移量，以及使用 **-s** 选项指定块大小。
b. 如果您希望对块制作镜像，请使用 **-m** 选项指定镜像路径和偏移量。
c. 要指定元数据空间的大小和偏移量，请使用 **-Mo** 和 **-Ms** 选项。
数据库服务器在新块上分配指定量的元数据区域、
d. 要让数据库服务器为新块计算元数据大小，那么省略 **-Mo** 和 **-Ms** 选项。
数据库服务器按用户数据区域大小划分智能大对象的估计平均大小。
e. 要将块仅用于用户数据，请指定 **-U** 选项。
如果您使用 -U 选项，那么数据库服务器不会在该块中分配元数据空间。取而代之，智能大对象空间使用一个其他块中的元数据区域。

3. 在将块添加到智能大对象空间后，数据库服务器将编写 CHRESERV 和 CHKADJUP 日志记录。
4. 对根数据库空间和智能大对象空间执行 0 级备份。
5. 使用 gstat -d 和 gcheck -pe 可监视智能大对象空间块中的可用空间量。

此示例向 **sbsp4** 添加 10 兆字节的镜像块。 对主块和镜像块都指定了 200 KB 的偏移量。如果您不添加镜像块，那么可省略 **-m** 选项。**-U** 选项指定新块仅包含用户数据。

```
gspaces -a sbsp4 -p /dev/rawdev1 -o 200 -s 10240 -m /dev/rawdev2 200 -U
```

还可以定义当应用程序需要更多存储空间时，GBase 8s 可用于自动扩展块大小的信息。如果有可扩展块，那么无需添加新块或耗费时间来尝试确定哪种存储空间（数据库空间、临时数据库空间、智能大对象空间、临时智能大对象空间或 BLOB 空间）将耗尽空间，以及何时将耗尽空间。

当启用 CHUNK_OVERLAP_PROTECTION 配置参数时，数据库服务器将确保新块不会与其他块重叠。如果服务器发现块重叠，新块添加操作将失败。

更改智能大对象的存储特征

使用 gspaces -ch 命令为智能大对象空间更改以下缺省存储特征：

- 扩展数据块大小
- 平均智能大对象大小
- 缓冲方式
- 上次访问时间
- 锁定方式
- 日志记录

创建临时智能大对象空间

有关确定智能大对象存储位置的背景信息和规则，请参阅临时智能大对象空间。您可将临时智能大对象存储在标准的或临时的智能大对象空间中。可以在临时智能大对象空间中添加或删除块。

要创建具有临时智能大对象的临时智能大对象空间，请执行以下操作：

1. 为临时智能大对象空间分配空间。

有关详细信息，请参阅分配磁盘空间。

有关 SBSPACETEMP 的信息，请参阅《GBase 8s 管理员参考》中有关配置参数的主题。

2. 如以下示例所示，创建临时智能大对象空间：

```
gspaces -c -S tempsbsp -t -p ./tempsbsp -o 0 -s 1000
```

3. 您可任意指定以下 gspaces 选项：
   a. 指定元数据区域和偏移量（**-Ms** 和 **-Mo**）。
   b. 指定存储特征 **(-Df)**。
      无法为临时智能大对象空间打开日志记录。
4. 将 SBSPACETEMP 配置参数设置为缺省临时智能大对象空间存储区域的名称。
   重新启动数据库服务器。
5. 使用 gstat -d 显示临时智能大对象空间。

   有关 gstat -d 输出示例的信息，请参阅《GBase 8s 管理员参考》中的 gstat 实用程序。

6. 在创建临时智能大对象时指定 LO_CREATE_TEMP 标志。

   使用 GBase 8s ESQL/C：

```
ifx_lo_specset_flags(lo_spec,LO_CREATE_TEMP);
```

有关创建智能大对象的信息，请参阅《GBase 8s ESQL/C 程序员手册》。

3.6.7 自动空间管理

可将服务器配置为需要更多空间时，自动添加更多存储空间。这样就可以更有效地使用空间，并确保在必要时分配空间，同时减少空间不足错误，以及减少手动监视空间和确定哪些存储空间将耗尽可用空间与何时耗尽空间所需的时间。将服务器配置为自动添加空间时，也可手动扩充空间或扩展块。

服务器扩展存储空间（数据库空间、临时数据库空间、智能大对象空间、临时智能大对象空间或 Blob 空间）时，服务器可向该存储空间添加块。如果存储空间是非镜像的数据库空间或临时数据库空间，服务器还可以扩展该存储空间中的块。

自动空间管理适用于参与 Enterprise Replication 的服务器或集群。要通过 Enterprise Replication 域传播 CDR_QDATA_SBSPACE 和 CDR_DBSPACE 配置参数，请使用 cdr define 命令。

要对空间的自动和手动管理进行配置，请运行 SQL 管理 API 命令来执行以下任务：

1. 在存储池中创建、修改和删除一个或多个条目。存储池中包含 GBase 8s 用于扩展存储空间的可用原始设备、熟文件和目录的条目。
2. 将块标记为可扩展。
3. 修改存储空间的创建和扩展大小（可选）。
4. 更改自动添加更多空间的阈值和等待时间（可选）。

5. 配置“监视低存储量”任务的频率（可选）。

如果存储池中包含条目，也可运行 SQL 管理 API 命令来执行以下操作：

- 不希望等待运行自动扩充空间的任务时，手动扩充存储空间或扩展块。
- 通过存储池条目手动创建存储空间，并将空存储空间中的空间返还给存储池。

如果不希望服务器自动扩充空间，可将 SP_AUTOEXPAND 配置参数设置为 0，以禁用块的自动创建或扩展。也可指定某个块不可扩展。

除了运行 SQL 管理 API 命令之外，也可使用 OpenAdmin Tool (OAT) 图形界面对空间的自动和手动管理进行配置，以及管理存储池条目。

提示：

在某些情况下，将数据库服务器配置为自动扩展现有存储空间后，该服务器可能不会自动扩展 DBSPACETEMP 配置参数中列出的临时数据库空间。如果使用临时数据库空间的操作（例如，索引构建或排序）耗尽空间，那么您将收到“空间不足”错误。

要解决此问题，必须手动向临时数据库空间添加块或者使用更大的临时数据库空间。

创建和管理存储池条目

可在存储池中添加、修改或删除条目（存储池是 GBase 8s 可在必要时用于自动向现有存储空间添加空间的可用原始设备、熟文件或目录的集合）。

存储池中的每一个条目都包含有关以下对象的信息：GBase 8s 实例可在必要时用于自动扩展现有数据库空间、临时数据库空间、智能大对象空间、临时智能大对象空间或 Blob 空间的目录、熟文件或原始设备。

创建存储池条目

要创建存储池条目，请运行带 storagepool add 自变量的 admin() 或 task() 函数，如下所示：

```
EXECUTE FUNCTION task("storagepool add", "path", "begin_offset",
"total_size", "chunk size", "priority");
```

指定以下信息：

- 需要更多存储空间时服务器可使用的文件、目录或设备的路径。
- GBase 8s 可开始分配空间的设备的偏移量（以 KB 计）。
- 此条目中 GBase 8s 可用的总空间。服务器可从该空间量中分配多个块。
- 可从设备、文件或目录分配的块的最小大小（以 KB 计）。可创建的最小块为 1000 K。因此，可指定的最小块大小为 1000 K。
- 优先级数字，从 1 到 3（1 = 高；2 = 中；3 = 低）。服务器会尝试先从高优先级的条目分配空间，然后再从较低优先级的条目分配空间。

存储池大小和偏移量的缺省单位为 KB。但也可以按照以下示例中显示的任意方式指定信息：

- "100000"

- "100000 K"
- "100 MB"
- "100 GB"
- "100 TB"

修改存储池条目

要修改存储池条目，请运行带 storagepool modify 自变量的 admin() 或 task() 函数，如下所示：

```
EXECUTE FUNCTION task("storagepool modify", "storage_pool_entry_id",
"new_total_size", "new_chunk size", "new_priority");
```

删除存储池条目

要删除存储池条目，请运行带 storagepool delete 自变量的 admin() 或 task() 函数，如下所示：

```
EXECUTE FUNCTION task("storagepool delete", "storage_pool_entry_id");
```

要删除所有存储池条目，请运行带 storagepool purge all 自变量的 admin() 或 task() 函数，如下所示：

```
EXECUTE FUNCTION task("storagepool purge all");
```

要删除已满的所有存储池条目，请运行带 storagepool purge full 自变量的 admin() 或 task() 函数，如下所示：

```
EXECUTE FUNCTION task("storagepool purge full");
```

要删除有错的存储池条目，请运行带 storagepool purge errors 自变量的 admin() 或 task() 函数，如下所示：

```
EXECUTE FUNCTION task("storagepool purge errors");
```

示例

以下命令添加开始偏移量为 0，总大小为 0，初始块大小为 20 兆字节且具有高优先级的目录，名称为 /region2/dbspaces。在该示例中，对于目录而言，只能接受偏移量为 0 和总大小为 0 这两个条目。

```
EXECUTE FUNCTION task("storagepool add", "/region2/dbspaces", "0", "0", "20000",
"1");
```

以下命令将存储池中第 8 个条目的总大小、块大小和优先级更改为 10 千兆字节、10 兆字节和中优先级。

```
EXECUTE FUNCTION task("storagepool modify", "8", "10 GB", "10000", "2");
```

以下命令删除条目标识为 7 的存储池条目：

```
EXECUTE FUNCTION task("storagepool delete", "7");
```

将块标记为可扩展或不可扩展

将块标记为可扩展，可以对块启用自动或手动扩展。将该标记更改为不可扩展，则可以防止自动或手动扩展块。

如果块标记为不可扩展：

- 块中只有少量或没有可用空间时，服务器不能自动扩展块。（但是，如果存储池中包含条目，那么服务器可通过向存储空间添加另一个块来扩展存储空间。）
- 不能手动扩展块的大小。

先决条件：可扩展块必须位于未镜像数据库空间或临时数据库空间中。

要将块标记为可扩展，请执行以下操作：

运行带 modify chunk extendable 自变量的 admin() 或 task() 函数，如下所示：

```
EXECUTE FUNCTION task("modify chunk extendable", "chunk number");
```

要将块标记为不可扩展，请执行以下操作：

运行带 modify chunk extendable off 自变量的 admin() 或 task() 函数，如下所示：

```
EXECUTE FUNCTION task("modify chunk extendable off", "chunk number");
```

以下命令指定可扩展第 12 个块：

```
EXECUTE FUNCTION task("modify chunk extendable", "12");
```

修改存储空间的创建或扩展大小

通过以下方法可以控制存储池条目的使用方式：修改与扩充存储空间相关的两个不同的数据库空间大小，即创建大小和扩展大小。

要修改存储空间的创建或扩展大小，请执行以下操作：

运行带 modify space sp_sizes 自变量的 admin() 或 task() 函数，如下所示：

```
EXECUTE FUNCTION task("modify space sp_sizes", "space_name",
 "new_create_size", "new_extend_size");
```

对于：

- new_create_size，请指定在指定的数据库空间、临时数据库空间、智能大对象空间、临时智能大对象空间或 Blob 空间中创建新块时，服务器可使用的最小大小。
- new_extend_size，请指定在指定的未镜像数据库空间或临时数据库空间中扩展块时，服务器可使用的最小大小。

使用数字（KB 数）或百分比（占总空间的百分比）指定大小。

以下命令将名为 dbspace3 的空间的创建大小和扩展大小分别设置为 60 兆字节和 10 兆字节：

```
EXECUTE FUNCTION task("modify space sp_sizes", "dbspace3", "60000", "10000");
```

以下命令将名为 logdbs 的空间的创建大小和扩展大小分别设置为 20% 和 1.5%：

```
EXECUTE FUNCTION task("modify space sp_sizes", "logdbs", "20", "1.5");
```

更改自动添加更多空间的阈值和等待时间

当存储空间已满时，GBase 8s 可以通过自动扩展或添加块来应对空间不足的情况，而您也可以配置服务器以在存储空间变满之前扩展或添加块。

为此，可为数据库空间、临时数据库空间、智能大对象空间、临时智能大对象空间或 Blob 空间中的最小可用 KB 量指定一个阈值。

定义的阈值用于触发扩充空间的任务。

也可使用 SP_WAITTIME 配置参数指定返回空间不足错误之前，线程等待空间扩充的最大秒数。

要更改阈值和等待时间，请执行以下操作：

1. 将 SP_THRESHOLD 配置参数中指定的阈值的值从 0（已禁用）更改为非零值。指定 1 到 50 的百分数或指定 1000 到块的最大大小（以 KB 计）。
2. 更改 SP_WAITTIME 配置参数的值，该配置参数指定返回空间不足错误之前，线程等待空间扩展的最大秒数。

配置监视低存储任务的频率

可更改 **mon_low_storage** 任务的频率，该任务定期扫描数据库空间列表，以查找低于 SP_THRESHOLD 配置参数指示的阈值的空间。 如果该任务找到了低于该阈值的空间，将尝试扩充找到的空间，方法是扩展可扩展块，或使用存储池来添加块。

mon_low_storage 任务的缺省频率为每小时一次，但也可以配置该任务，以增加或减小运行频率

先决条件：在 SP_THRESHOLD 配置参数中，为数据库空间、临时数据库空间、智能大对象空间、临时智能大对象空间或 Blob 空间中的最小可用 KB 量指定一个值。

要将 mon_low_storage 任务配置为以更高或更低频率运行，请执行以下操作：

运行以下 SQL 语句，其中 *minutes* 是两次运行之间的分钟数：

```
DATABASE sysadmin;
UPDATE ph_task set tk_frequency = INTERVAL (minutes)
 MINUTE TO MINUTE WHERE tk_name = “mon_low_storage”;
```

例如，要将任务配置为每 10 分钟运行一次，请运行以下 SQL 语句：

```
DATABASE sysadmin;
UPDATE ph_task set tk_frequency = INTERVAL (10) MINUTE TO MINUTE
 WHERE tk_name = “mon_low_storage”;
```

手动扩充空间或扩展可扩展块

必要时可手动扩充空间或扩展块，而无需等待 GBase 8s 自动扩充空间或扩展块。

先决条件：

- 只能扩展位于未镜像数据库空间或临时数据库空间中的块。
- 块必须先标记为可扩展，然后才能扩展。否则，必须运行带 modify chunk extendable 自变量的 admin() 或 task() 函数来指定该块可扩展。
- 如果不能通过扩展块来扩充空间，那么存储池中必须包含服务器可用于创建新块的活动条目。

要立即增加存储空间：

执行以下任一项操作：

- 通过运行带 modify space expand 自变量的 admin() 或 task() 函数手动扩充空间，如下所示：

  ```
  EXECUTE FUNCTION task("modify space expand", "space_name", "size");
  ```

  例如，以下命令将编号为 8 的空间扩展 1 千兆字节：

  ```
  EXECUTE FUNCTION task("modify space expand", "8", "1000000");
  ```

  服务器通过扩展空间中的块或添加新块来扩充空间。服务器可能对请求的大小向上取整，具体取决于存储空间的页大小以及扩展期间任何存储池条目所使用的已配置块大小。

- 通过运行带 modify chunk extend 自变量的 admin() 或 task() 函数手动扩展块，如下所示：

  ```
  EXECUTE FUNCTION task("modify chunk extend", "chunk_number",
  "extend_amount");
  ```

  例如，以下命令将编号为 12 的块扩展 5000 KB：

  ```
  EXECUTE FUNCTION task("modify chunk extend", "12", "5000");
  ```

  服务器可能对请求的大小向上取整，具体取决于存储空间的页大小。

对自动添加更多空间的最低程度配置和测试的示例

此示例显示可如何最低程度地配置和测试自动添加更多空间。可通过创建数据库空间、填充空间、向 GBase 8s 存储池添加条目和将表装入空间来执行此操作。空间填满之后，GBase 8s 将自动扩展该空间。

要最低程度地配置和测试自动添加更多空间，请执行以下操作：

1. 创建数据库空间。

   例如，创建名为 expandable_dbs 的数据库空间，并将名为 /my_directory/my_chunk 的熟文件的前 10000 KB 分配给初始块，如下所示：

   ```
   gspaces -c -d expandable_dbs -p /my_directory/my_chunk -o 0 -s 10000
   ```

2. 填充数据库空间。

   例如，在不装入数据行的情况下填充数据库空间。实际操作是创建表，然后为第一个扩展数据块分配一大组邻接的可用页，如下所示：

```
CREATE TABLE large_tab (col1 int) IN expandable_dbs EXTENT SIZE
10000000;
```

可通过使用 **gstat -d** 命令或 OpenAdmin Tool (OAT) 来监视块中的可用页。如果数据库空间已满，那么在尝试创建数据并将其装入到另一个新表时，将收到空间不足的错误。

3. 向 GBase 8s 存储池添加条目。

   例如，将 $GBASEDBTDIR/tmp 目录添加到存储池，如下所示：

```
DATABASE sysadmin;
EXECUTE FUNCTION task("storagepool add", "$GBASEDBTDIR/tmp",
 "0", "0", "10000", "2");
```

4. 在 SP_THRESHOLD 配置参数中，设置 GBase 8s 自动运行任务以扩充空间之前，存储空间中可包含的最小可用 KB 量的阈值。
5. 创建新表并将其装入数据库。

   现在，如果存储空间变满，不会再收到空间不足的错误，而是由 GBase 8s 自动在 $GBASEDBTDIR/tmp 文件中创建熟文件，并使用这个新的熟文件向 expandable_dbs 数据库添加块。继续填充此块时，服务器将自动对其进行扩展。向数据库空间添加新块之前，如果可能，服务器将始终扩展块。

6. 减小存储空间中的可用空间，以测试 SP_THRESHOLD 配置参数中的值。

   在数据库空间、临时数据库空间、智能大对象空间、临时智能大对象空间或 Blob 空间中分配足够的页面，以减少可用空间，使其低于 SP_THRESHOLD 指示的阈值。但是，请勿填满该空间。

   下次运行 **mon_low_storage** 任务时，您会发现该空间将自动扩展。

7. 创建空间不足的条件。

   在数据库空间、临时数据库空间、智能大对象空间、临时智能大对象空间或 Blob 空间中分配所有页。然后尝试分配更多页。必须确保此分配成功，并且确保不会收到空间不足的错误。

   GBase 8s 每次扩展或添加块以及将新块标记为可扩展时，都会向日志中写入消息。

   运行 gstat -d 命令可显示实例中的所有块。查找可扩展块，这种块带有 E 标志。该命令输出会显示服务器自动扩充了空间，方法是添加了新块或扩展了现有块的大小。

自动添加更多空间的配置示例

此示例显示可通过以下方法对自动添加更多空间进行充分配置：更改某些配置参数的设置，更改用于监视低存储的任务的频率，以及指定可扩展空间和块的信息。

要配置为自动添加更多存储空间：

1. 向存储池添加条目。

   例如，将 $GBASEDBTDIR/tmp 目录添加到存储池，如下所示：

```
DATABASE sysadmin;
EXECUTE FUNCTION task("storagepool add", "$GBASEDBTDIR/tmp",
 "0", "0", "10000", "2");
```

2. 将未镜像数据库空间和临时数据库空间中的某些块标记为可扩展，以便将来必要时，服务器可扩展这些块。

   例如，指定第 12 个块可扩展：

   ```
   EXECUTE FUNCTION task("modify chunk extendable", "12");
   ```

   也可将可扩展块的标记更改为不可扩展。 例如，指定编号为 10 的块不可扩展：

   ```
   EXECUTE FUNCTION task("modify chunk extendable off", "10");
   ```

3. 在 SP_THRESHOLD 配置参数中，设置 GBase 8s 自动运行任务以扩充空间之前，存储空间中可包含的最小可用 KB 量的阈值。指定以下任一项：
   - 1 到 50 的百分比值。
   - 1000 到块最大大小的值（以 KB 计）

   如果单个存储空间的填充程度超过了您定义的此阈值，并且该填充程度持续到下次运行空间监视任务（mon_low_storage）为止，那么服务器将通过扩展可扩展块或通过使用存储池添加块来尝试扩充该空间。

   例如，假设 SP_THRESHOLD 值为 5.5，服务器将该值视为 5.5%。如果某个空间中的可用页很少，可用空间百分比降到 5.5% 以下，并且这种低水平持续到下次运行 mon_low_storage 任务为止，那么该任务将尝试扩展此空间。如果 SP_THRESHOLD 设置为 50000，并且某个空间的可用空间低于 50000 KB，那么下次运行 mon_low_storage 时将扩展该空间。

4. **可选：**更改 **mon_low_storage** 任务的运行频率。该任务定期扫描数据库空间列表，以查找低于 SP_THRESHOLD 配置参数指示的阈值的空间。

   例如，要将任务配置为每 10 分钟运行一次，请运行以下 SQL 语句：

   ```
   DATABASE sysadmin;
   UPDATE ph_task set tk_frequency = INTERVAL (10) MINUTE TO MINUTE
    WHERE tk_name = “mon_low_storage”;
   ```

5. **可选：**更改 SP_WAITTIME 配置参数的值，该配置参数指定返回空间不足错误之前，线程等待空间扩展的最大秒数。
6. **可**选：更改与扩展存储空间相关的两种大小：
   - 扩展大小，这是扩展数据库空间或临时数据库空间中的块时使用的最小大小。
   - 创建大小，这是在不属于镜像空间的数据库空间、临时数据库空间、智能大对象空间、临时智能大对象空间或 Blob 空间中创建新块时使用的最小大小。

     例如，以下命令将编号为 3 的空间的创建大小和扩展大小分别设置为 60 兆字节和 10 兆字节：

     ```
     EXECUTE FUNCTION task("modify dbspace sp_sizes",
      "3", "60000", "10000");
     ```

对存储空间的自动扩充进行了配置之后，也可在必要时手动扩充空间或扩展空间中的块。

3.6.8 删除块

使用 gspaces 或 Server Administrator 从数据库空间删除块。

在删除块之前，请使用下表作为指导方针，确保数据库服务器处于正确方式。

块类型	联机方式下的数据库服务器	管理方式或静默方式下的数据库服务器	脱机方式下的数据库服务器
数据库空间块	是	是	否
临时数据库空间块	是	是	否
BLOB 空间块	否	是	否
智能大对象空间或临时智能大对象空间块	是	是	否

验证块是否为空

要用这些实用程序之一从数据库空间成功删除块，块不能包含任何数据。开销页以外的所有页都必须得到释放。

如果有任何页保持分配给未开销的实体，那么实用程序会返回以下错误：块不为空。

此外，当数据库空间由两个或更多的块组成并且附加块不包含用户数据时，如果这些附加块包含表空间 **tblspace**，那么它们无法被删除。

如果接收到块不为空消息，必须通过运行 gcheck -pe 列出扩展数据块内容，从而确定哪些表或其他实体仍在占用块中的空间。

通常，在您删除拥有这些页的表时就会除去这些页。然后重新输入实用程序命令。

使用 gspaces 从数据库空间删除块

以下示例在 UNIX™ 上从 **dbsp3** 删除块。 指定了 300 KB 的偏移量。

```
gspaces -d dbsp3 -p /dev/raw_dev1 -o 300
```

您不能用以上示例中的语法来删除数据库空间的初始块。取而代之，您必须删除数据库空间。使用 **gstat -d** 的 fchunk 列来确定数据库空间的初始块。有关 gstat 的更多信息，请参阅《GBase 8s 管理员参考》中有关 gspaces 实用程序的信息。

有关使用 gspaces 从数据库空间删除块的信息，请参阅《GBase 8s 管理员参考》。

从 BLOB 空间删除块

从 BLOB 空间删除块的过程与使用 gspaces 从数据库空间删除块中所述的从数据库空间删除块的过程相同，但是数据库服务器必须处于静默方式或管理方式。除了这一条件，还必须将任何出现对数据库空间的引用之处替换为您的 BLOB 空间名称。

使用 gspaces 从智能大对象空间删除块

以下示例在 UNIX™ 上从 **sbsp3** 删除块。 指定了 300 KB 的偏移量。当从智能大对象空间或临时智能大对象空间删除块时，数据库服务器必须处于联机管理方式或静默方式。

```
gspaces -d sbsp3 -p /dev/raw_dev1 -o 300
```

您不能用以上示例中的语法来删除智能大对象空间的初始块。取而代之，您必须删除智能大对象空间。使用 **gstat -d** 的 fchunk 列来确定哪个块是智能大对象空间的初始块。

-f（强制）选项

您可以使用 **gspaces** 的 -f 选项来删除其中未分配元数据的智能大对象空间块。 如果该块包含智能大对象空间的元数据，那么必须删除整个智能大对象空间。使用 **gstat -d** 的块部分来确定哪些智能大对象空间块包含元数据。

```
gspaces -d sbsp3 -f
```

警告： 如果您强制删除智能大对象空间，那么可能为表和智能大对象空间之间造成一致性方面的问题。

删除不带任何指针的智能大对象

每个智能大对象均有引用计数，即智能大对象的指针数。当引用计数大于 0 时，数据库服务器假定智能大对象正在使用中，就不会予以删除。

而引用计数为 0 的智能大对象很少会保留下来。您可使用 gspaces -cl 命令删除所有引用计数为 0 的智能大对象（如果它未被任何应用程序打开）。

有关使用 gspaces -cl 的信息，请参阅《GBase 8s 管理员参考》中有关 gspaces 实用程序的信息。

3.6.9 删除存储空间

使用 gspaces 或 ON-Monitor 可删除数据库空间、临时数据库空间、BLOB 空间、智能大对象空间、临时智能大对象空间或外部空间。

在 UNIX™ 上，您必须以 root 或 **gbasedbt**身份登录来删除存储空间。

仅当数据库服务器处于联机、管理或静默方式时，才能删除存储空间。

删除存储空间的准备工作

在删除数据库空间之前，您必须首先删除所有先前在数据库空间创建的数据库和表。您不能删除根数据库空间。

在删除 BLOB 空间之前，必须删除具有引用了该 BLOB 空间的 TEXT 或 BYTE 列的所有表。

运行 gcheck -pe 以验证没有表或日志文件位于数据库空间或 BLOB 空间中。

在删除某个智能大对象空间之前，必须删除具有引用了该智能大对象空间中存储对象的 CLOB 或 BLOB 列的所有表。对于智能大对象空间，无需删除指向智能大对象空间的列，但这些列必须为空；也就是说，所有智能大对象必须取消分配至智能大对象空间。

提示： 如果您在发生轻量级追加的数据库空间中删除表，这些轻量级追加可能会比您所期望的要慢。该问题的症状就是物理日志记录活动。如果轻量级追加比您所期望的要慢，请确保在轻量级追加之前或在轻量级追加期间没有从数据库空间中删除表。如果您已删除了表，请在执行轻量级追加之前用 gadmin -c 强制执行检查点。

重要： 删除块或数据库空间将触发阻塞检查点，这会强制所有数据库更新等待所有缓冲池清空到磁盘之后才执行。此更新阻塞在阻塞检查点期间所用时间可能比非阻塞检查点期间所用时间长得多，特别是缓冲池很大的情况下。

删除镜像存储空间

如果您删除已镜像的存储空间，那么镜像空间也被删除。

如果您希望仅删除存储空间镜像，请关闭镜像。（请参阅结束镜像过程。）该操作删除数据库空间、BLOB 空间或智能大对象空间镜像，并释放块以作其他用途。

使用 gspaces 删除存储空间

要使用 gspaces 删除存储空间，请按以下示例说明，使用 **-d** 选项。

以下示例删除名为 **dbspce5** 的数据库空间及其镜像。

```
gspaces -d dbspce5
```

以下示例删除名为 **blobsp3** 的数据库空间及其镜像。

```
gspaces -d blobsp3
```

如果您希望删除包含数据的智能大对象空间，那么将 **-d** 选项与 **-f** 选项一起使用。如果省略 **-f** 选项，您就无法删除包含数据的智能大对象空间。此示例删除称为 **sbspc4** 的智能大对象空间及其镜像。

```
gspaces -d sbspc4 -f
```

警告： 如果您使用 **-f** 选项，数据库服务器中的表可能包含指向已删除的智能大对象的无效指针。

有关使用 gspaces 删除存储空间的信息，请参阅《GBase 8s 管理员参考》中有关 gspaces 实用程序的信息。

使用 ON-Monitor 删除数据库空间或 BLOB 空间 (UNIX™)

可以使用 ON-Monitor 删除数据库空间或 BLOB 空间。

要使用 ON-Monitor 删除数据库空间或 BLOB 空间：

1. 选择数据库空间 > 删除选项。
2. 使用 Enter 键或方向键滚动至您希望删除的数据库空间或 BLOB 空间。
3. 按下 CTRL-B 或 F3。

要求您确认要删除数据库空间或 BLOB 空间。

删除存储空间后备份

如果您创建了名称与已删除存储空间相同的存储空间，请执行 0 级备份，以确保以后的存储不会将新存储空间与旧存储空间相混淆。有关更多信息，请参阅《*GBase 8s 备份与复原指南*》。

重要： 在删除数据库空间、BLOB 空间或智能大对象空间之后，新释放的块可用于重新分配至其他数据库空间、BLOB 空间或智能大对象空间。但是，在重新分配新释放的块之前，必须对根数据库空间和修改过的存储空间执行 0 级备份。如果未执行该备份，但随后必须执行复原，那么复原可能失败，因为备份保留页不是最新的。

3.6.10 从存储池创建空间或块

如果存储池中包含条目，可从存储池中的可用空间创建存储空间或块。

先决条件：存储池中必须包含条目（目录、熟文件或原始设备）。

要从存储池创建存储空间或块，请执行以下操作：

运行带以下自变量之一的 admin() 或 task() 函数从存储池创建空间。命令中所用元素取决于要创建的空间类型。

- EXECUTE FUNCTION task("create dbspace from storagepool", "space_name", "size", "page_size", "mirroring_flag", "first_extent", "next_extent");
- EXECUTE FUNCTION task("create tempdbspace from storagepool", "space_name", "size", "page_size");
- EXECUTE FUNCTION task("create blobspace from storagepool", "space_name", "size", "page_size", "mirroring_flag",);
- EXECUTE FUNCTION task("create sbspace from storagepool", "space_name", "size", "log_flag", "mirroring_flag",);
- EXECUTE FUNCTION task("create tempsbspace from storagepool", "space_name", "size",);
- EXECUTE FUNCTION task("create chunk from storagepool", "space_name", "size",);

示例

以下命令创建名为 blobspace1 的镜像 Blob 空间。新 Blob 空间的大小为 100 千兆字节，Blob 页大小为 100 页。

```
EXECUTE FUNCTION task("create blobspace from storagepool", "blobspace1", "100 GB", "100", "1");
```

以下命令向名为 logdbs 的数据库空间添加块。新块的大小为 200 兆字节。

```
EXECUTE FUNCTION task("create chunk from storagepool", "logdbs", "200 MB");
```

3.6.11 将空的空间返还给存储池

可将空块或空存储空间中的空间返还给存储池。

要将空的块、数据库空间、临时数据库空间、Blob 空间、智能大对象空间或临时智能大对象空间中的存储空间返还给存储池，请执行以下操作：

运行带以下自变量之一的 admin() 或 task() 函数将空间返还给存储池。命令中所用元素取决于要删除的对象类型。

- EXECUTE FUNCTION task("drop chunk to storagepool", "space_name", "chunk_path", "chunk_offset")
- EXECUTE FUNCTION task("drop dbspace to storagepool", "space_name");
- EXECUTE FUNCTION task("drop tempdbspace to storagepool", "space_name");
- EXECUTE FUNCTION task("drop blobspace to storagepool", "space_name");
- EXECUTE FUNCTION task("drop sbspace to storagepool", "space_name");
- EXECUTE FUNCTION task("drop tempsbspace to storagepool", "space_name");

示例

以下命令删除名为 blob4 的空 Blob 空间，并将释放的所有空间添加到存储池。

```
EXECUTE FUNCTION task("drop blobspace to storagepool", "blob4");
```

以下命令删除名为 health 的数据库空间中的空块，并将释放的所有空间添加到存储池。

```
EXECUTE FUNCTION task("drop chunk to storagepool", "health",
 "/health/rawdisk23", "100 KB");
```

3.6.12 管理外部空间

外部空间不需要分配磁盘空间。 使用 gspaces 实用程序创建和删除外部空间。 有关外部空间的更多信息，请参阅外部空间。

创建外部空间

用 gspaces 实用程序创建外部空间。但是首先必须具有有效数据源和用来访问该数据源的有效访问方法。虽然您可以创建不带有效访问方法或有效数据源的外部空间，但是从该外部空间检索数据的任何尝试都将生成错误。

要用 gspaces 创建外部空间，请按以下示例说明，使用 -c 选项。以下示例显示了如何创建与 UNIX™ 密码文件相关联的外部空间 **pass_space**。

```
gspaces -c -x pass_space -l /etc/passwd
```

指定最多为 128 字节的外部空间名称。该名称必须是唯一的，并以字母或下划线开始。您可以在名称中使用字母、数字、下划线和 **$** 字符。

重要： 以上示例假设您编写了提供正确访问文件 passwd 的函数的例程，并假设文件本身已存在。在创建了外部空间之后，就必须使用适当的命令以允许访问文件 passwd 中的数据。有关用户定义的访问方法的更多信息，请参阅*《GBase 8s虚拟表接口程序员指南》*。

有关使用 gspaces 创建外部空间的参考信息，请参阅《GBase 8s 管理员参考》中有关 gspaces 实用程序的信息。

删除外部空间

要使用 gspaces 删除外部空间，请按以下示例说明使用 **-d** 选项。外部空间如果不与现有表或索引相关联，就无法删除。

以下示例删除名为 **pass_space** 的外部空间。

```
gspaces -d pass_space
```

3.6.13 跳过不可访问的分段

分段存储的一个好处是可以在 I/O 操作期间跳过不可用的表分段。例如，即使当分段位于当前由于磁盘故障而脱机的块上时，查询也可继续。发生这种情况时，磁盘故障仅影响分段表中的部分数据。相较而言，如果未分段的表位于故障磁盘上，它们可能会变得完全不可访问。

如下所示控制此功能：

- 由数据库服务器管理员使用 DATASKIP 配置参数
- 由各个应用程序使用 SET DATASKIP 语句

DATASKIP 配置参数

可将 DATASKIP 参数设置为 OFF、ALL 或 ON dbspace_list。OFF 表示数据库服务器不跳过任何分段。如果分段不可用，查询将返回错误。ALL 指示跳过所有不可用的分段。ON dbspace_list 指示数据库服务器跳过任何位于指定数据库空间中的分段。

gspaces 的数据跳过功能

使用 gspaces 实用程序的 dataskip 功能来指定要跳过的数据库空间（当这些数据库空间不可用时）。 例如，以下命令设置 DATASKIP 参数，使数据库服务器跳过 dbspace1 和 dbspace3 中的分段，但不跳过 dbspace2 中的分段：

```
gspaces -f ON dbspace1 dbspace3
```

有关此 gspaces 选项的完整语法，请参阅《GBase 8s 管理员参考》中有关 gspaces 实用程序的信息。

使用 gstat 检查数据跳过状态

使用 gstat 实用程序列出当前受 dataskip 功能影响的数据库空间。**-f** 选项同时列出使用 DATASKIP 配置参数以及使用 gspaces 实用程序的 **-f** 选项设置的数据库空间。

运行 gstat -f 时，您会收到消息，通知您 DATASKIP 配置参数是对所有数据库空间设置为 on、对所有数据库空间设置为 off，还是对特定数据库空间设置为 on。

SQL 语句 SET DATASKIP

应用程序可以使用 SQL 语句 SET DATASKIP 来控制在某个分段不可用时是否跳过该分段。 应用程序必须仅在有限情况下包含此语句，因为它会使查询根据底层分段的可用性而返回不同结果。与配置参数 DATASKIP 类似，SET DATASKIP 语句接受那些向数据库服务器指示要跳过哪些分段的数据库空间列表。例如，假设某应用程序的编程人员将以下语句包含在应用程序的开始：

```
SET DATASKIP ON dbspace1, dbspace5
```

该语句使数据库服务器在以下两个条件都满足时就跳过 **dbspace1** 或 **dbspace5**：

- 应用程序尝试访问这两个数据库空间之一。
- 数据库服务器发现这两个数据库空间之一不可用。

如果数据库服务器发现 **dbspace1** 和 **dbspace5** 均不可用，它会跳过这两个数据库空间。

数据库服务器管理员可使用 SET DATASKIP 语句的 DEFAULT 设置来控制数据跳过功能。假设应用程序开发者将以下语句包含在应用程序中：

```
SET DATASKIP DEFAULT
```

如果在此 SQL 语句之后紧接着运行查询，那么数据库服务器会检查配置参数 DATASKIP 的值。数据库服务器管理员可鼓励用户使用此设置，以便指定当数据库服务器管理员注意到一个或多个数据库空间不可用时，要立即跳过的数据库空间。

数据跳过功能对事务的影响

如果打开 dataskip 功能，那么 SELECT 语句将始终执行。另外，如果表由循环分段且至少有一个分段联机，那么 INSERT 语句将始终成功执行。但是，当写入数据库的操作可能会影响数据库的完整性时，数据库服务器不会完成这类操作。以下操作失败：

- 当数据库服务器无法消除脱机的分段时，为所有的 UPDATE 和 DELETE 操作

  如果数据库服务器可以消除脱机的分段，那么更新或删除操作成功，但结果与 DATASKIP 设置无关。

- 当相应分段脱机时，根据基于表达式的分布方案对已分段的表的 INSERT 操作
- 当约束包含脱机的分段中的数据时，任何包含参阅约束检查的操作

  例如，如果应用程序删除具有子行的行，那么子行必须是可用的，以便删除。

- 当所提到的索引位于脱机的块中时，任何影响索引值的操作（例如，对已建立索引的列的更新）

确定何时使用数据跳过

要少用该功能且使用时要谨慎，因为该结果始终是不可靠的。可考虑在以下情况中使用该功能：

- 您可以接受事务完整性受影响。
- 您可以确定事务完整性未受影响。

后一任务可能很难且很耗时。

确定何时跳过选定的分段

在某种情况下，您可能希望数据库服务器跳过某些分段而不是其他分段。这通常会发生在以下情况中：

- 因为分段对查询结果没有重要作用，所以可以跳过这些分段。
- 某些分段脱机，并且您断定跳过这些分段并返回有限数量的数据将比取消查询要好。

当要跳过分段时，请使用 ON *dbspace-list* 设置来指定包含数据库服务器必须跳过的分段的数据库空间列表。

确定何时跳过所有分段

将 DATASKIP 配置参数设置为 ALL 将使数据库服务器跳过所有不可用的分段。请谨慎使用此选项。如果数据库空间变得不可用，那么在执行前未发出 SET DATASKIP OFF 语句的应用程序启动的所有查询均可能发生错误。

监视分段存储使用

数据库管理员可能会发现分段存储的以下方面对监视是有用的：

- 分段上的数据分发
- 分段上的 I/O 请求均衡
- 包含分段的块的状态

管理员可以监视表分段上的数据分发。如果分段存储的目标是提高管理响应时间，那么数据要均匀分布在分段上是很重要的。要监视分段存储磁盘使用，您必须监视数据库服务器表空间，因为分段的磁盘存储单位是表空间。（有关如何监视分段表的数据分发的信息，请参阅监视表空间和扩展数据块。）

管理员必须监视包含在分段中的数据的 I/O 请求队列。当 I/O 队列变得不平衡时，管理员必须使用 DBA 来调整分段存储策略。（有关如何监视块使用的说明，包括如何监视每个块的 I/O 队列，请参阅监视块。）

管理员必须为可用性而监视分段，并在包含一个或多个分段的数据库空间发生故障时采取适当的步骤。有关如何确定块是否脱机，请参阅监视块。

3.6.14 显示数据库

可以显示通过 SMI 表或 ON-Monitor 创建的数据库。

SMI 表

查询 **sysdatabases** 表以显示对应于数据库服务器管理的每个数据库的行。 有关此表中的列的描述，请参阅《GBase 8s 管理员参考》中有关 **sysmaster** 数据库的主题中的 **sysdatabases** 信息。

使用 GBase 8s Server Administrator

要使用 Server Administrator (ISA) 查询 **sysdatabases**，请遵循以下步骤：

1. 选择 SQL > 查询。
2. 在**数据库**列表中选择 **sysmaster** 数据。
3. 输入以下命令并单击提交：select * from sysdatabases;

使用 ON-Monitor 查找数据库状态 (UNIX™)

要使用 ON-Monitor 查扎每个数据库的当前状态，请选 择状态 > 数据库选项。

ON-Monitor 仅能显示最多 100 个数据库。若您的数据库服务器上有 100 多个数据库，请按上一节所述，使用 SMI 表来显示完整列表。

3.6.15 监视磁盘使用量

这些主题描述跟踪各种数据库服务器存储单元所使用磁盘空间的方法。

有关本节中提及的内部数据库服务器存储单元的背景信息，请参阅《GBase 8s 管理员参考》中有关磁盘结构和存储的章节。

监视块

可以监视块，获取以下信息：

- 块大小
- 可用页数
- 块中的表

可以使用这些信息跟踪块所使用的磁盘空间、监视块 I/O 活动以及检查是否有分段存储。

gstat -d 实用程序

gstat -d 实用程序列出所有数据库空间、BLOB 空间和智能大对象空间以及以下有关这些空间中的块的信息。

- 块地址
- 块号及相关联的数据库空间号
- 设备中的偏移量（以页计）
- 块大小（以页计）
- 块中的可用页数
- 物理设备的路径名

如果您在带有 BLOB 空间块的某实例上发出 gstat -d 命令，那么所显示的可用页数是过时的。在 free 值之前的波浪号 (~) 指示该数值是大约值。在备份其中执行了删除的逻辑日志

且释放 BLOB 页之后，gstat -d 命令才会将 BLOB 页注册为可用页。因此，如果删除了 25 个简单大对象并立即运行 gstat -d，gstat 输出中将不会包含新释放的空间。

要获取 BLOB 空间块中准确的可用 BLOB 页数，可发出 gstat -d update 命令。有关详细信息，请参阅 gstat -d update 选项。

在 **gstat -d update** 输出中，**块**部分的**标志**列提供以下信息：

- 块是主块还是镜像块
- 块是联机、脱机、正在恢复还是新块

有关 gstat -d 输出的示例，请参阅《GBase 8s 管理员参考》中有关 gstat 实用程序的信息。

重要： 在关闭镜像之后以及镜像可活动之前，必须对根数据库空间和修改过的数据库空间执行 0 级备份。

gstat -d update 选项

gstat -d update 选项显示与 gstat -d 所显示信息相同的信息以及每个 BLOB 空间块的准确的可用 BLOB 页数。

gstat -D 选项

gstat -D 选项显示与 gstat -d 所显示信息相同的信息，另外还显示从块中读取的页数（在**页读取**字段中）。

使用 gstat -g iof 命令监视块 I/O 活动

可以使用 gstat -g iof 命令监视分段表的不同分段上的 I/O 请求的分布。

gstat -g iof 命令显示：

- 从每个块的读取次数和向每个块写入的次数。
- 单个操作中断的 I/0（按服务级别）
- 操作类型
- 操作发生的次数
- 完成操作所用平均时间

如果某个块具有数量与之不成比例的 I/O 活动，那么该块可能会成为系统的瓶颈。

有关 gstat -g iof 输出的示例，请参阅《GBase 8s 管理员参考》中有关 gstat 实用程序的信息。

gcheck -pr 命令

数据库服务器将块信息存储在保留页 PAGE_1PCHUNK 和 PAGE_2PCHUNK 中。

要列出保留页的内容，请运行 gcheck -pr。以下示例显示 gcheck -pr 的样本输出。此输出基本上与 gstat -d 的输出相同；但如果块信息自上一个检查点之后更改过，那么 gcheck -pr 输出中不会包含这些更改。

```
Validating PAGE_1DBSP & PAGE_2DBSP...
Using dbspace page PAGE_2DBSP.
```

```
        DBspace number          1
        DBspace name            rootdbs
        Flags                   0x20001         No mirror chunks
        Number of chunks        2
        First chunk             1
        Date/Time created       07/28/2008 14:46:55
        Partition table page number     14
        Logical Log Unique Id   0
        Logical Log Position    0
        Oldest Logical Log Unique Id    0
        Last Logical Log Unique Id      0
        Dbspace archive status          No archives have occurred
.
Validating PAGE_1PCHUNK & PAGE_2PCHUNK...
Using primary chunk page PAGE_2PCHUNK.
        Chunk number            1
        Flags                   0x40            Chunk is online
        Chunk path              /home/server/root_chunk
        Chunk offset            0 (p)
        Chunk size              75000 (p)
        Number of free pages    40502
        DBSpace number          1
```

.

gcheck -pe 命令

要获取块中信息的物理布局，请运行 gcheck -pe。将列出数据库空间、BLOB 空间和智能大对象空间。以下示例显示 gcheck -pe 的样本输出。

显示以下信息：

- 数据库空间的名称、所有者和创建日期
- 块大小（以页计）、已用页数和可用页数
- 块中所有表的列表，包含初始页数和页中的表长度

按顺序列出块中的表。该输出对于确定块分段存储是有用的。如果数据库服务器无法在块中分配扩展数据块（尽管有足够的可用页数），那么该块可能会被错误地分段。

```
DBSpace Usage Report:   rootdbs         Owner:  GBase 8s    Created: 08/08/2006


Chunk   Pathname                                Size        Used        Free
```

```
  1        /home/server/root_chunk                75000      19420      55580


Description                                          Offset      Size
---------------------------------------------      ----------------------------
RESERVED PAGES                                          0          12
CHUNK FREELIST PAGE                                    12           1
rootdbs:'gbasedbt'.TBLSpace                                        13
250
PHYSICAL LOG                                          263        1000
FREE                                                 1263        1500
LOGICAL LOG: Log file 2                              2763        1500
LOGICAL LOG: Log file 3                              4263        1500
...
sysmaster:'gbasedbt'.sysdatabases                   10263           4
sysmaster:'gbasedbt'.systables                      10267           8

...

Chunk    Pathname                                    Size       Used       Free
     2    /home/server/dbspace1                      5000         53       4947


Description                                          Offset       Size
---------------------------------------------      ---------------------------------
RESERVED PAGES                                          0           2
CHUNK FREELIST PAGE                                     2           1
dbspace1:'gbasedbt'.TBLSpace                            3          50
FREE                                                   53        4947
```

Server Administrator

可以使用 Server Administrator (ISA) 命令来执行以下任务：

- 检查保留页。
- 检查存储空间。
- 添加数据库空间、临时数据库空间、BLOB 空间、临时智能大对象空间和智能大对象空间。
- 显示块并将块添加至存储空间。
- 检查 dataskip 状态。
- 显示并添加外部的空间。
- 显示您数据库中的页数、已分配空间的百分比和已使用的空间。
- 覆盖 ONDBSPACEDOWN。

使用 ON-Monitor 监视空间和块 (UNIX™)

可以使用 ON-Monitor 来创建和监视空间和块。

您可以使用 ON-Monitor 命令来执行以下任务。

状态 > 空间
    显示有关存储空间和块的状态信息
数据库空间 > 创建
    创建数据库空间

数据库空间 > BLOB 空间
创建 BLOB 空间
数据库空间 > 镜像
为存储空间添加或删除镜像
数据库空间 > 信息
显示有关存储空间的信息
数据库空间 > 添加块
向存储空间添加块
数据库空间 > 数据跳过
启动或停止 dataskip
数据库空间 > 块
向数据库空间或 BLOB 空间添加块
数据库空间 > 删除
删除数据库空间或 BLOB 空间
数据库空间 > 状态
更改块的镜像状态

SMI 表

查询 **syschunks** 表以获取块状态。以下是相关列。

chknum
数据库空间中的块数
dbsnum
数据库空间数
chksize
总的块大小（以页为单位）
nfree
可用页数
is_offline
块是否脱机
is_recovering
块是否正在恢复
mis_offline
镜像块是否脱机
镜像块是否正在恢复

syschkio 表包含以下列。

pagesread
从块中读取的页数
pageswritten
向块中写入的页数

监视表空间和扩展数据块

监视 tblspaces 和扩展数据块按数据库、表或表分段确定磁盘使用。 当您正使用表分段存储并希望确保表数据和表索引数据适当分布在分段上之时，监视表的磁盘使用就特别重要。

运行 gcheck -pt 以获取扩展数据块信息。gcheck -pT 选项返回所有来自 gcheck -pt 选项的信息以及有关页和索引使用的其他信息。

SMI 表

查询 **systabnames** 表以获取有关每个表空间的信息。**systabnames** 表包含指示相应表、数据库和每个表空间所有者的列。

查询 **sysextents** 表以获取有关每个扩展数据块的信息。**sysextents** 表包含指示扩展数据块所属的数据库和表的列，以及扩展数据块的物理地址和大小的列。

监视 BLOB 空间中的简单大对象

监视 BLOB 空间以确定可用空间以及 BLOB 页大小是否为最佳大小。

gstat -O 选项

gstat -O 选项显示有关登台区域 BLOB 空间和 Optical Subsystem 内存高速缓存的信息。所显示的总计随会话而逐一累计。仅当运行 gstat -z 时，数据库服务器才会将总计重置为 0。

有关 gstat -O 输出的示例，请参阅《GBase 8s 管理员参考》中有关 gstat 实用程序的信息。

显示的第一部分描述以下系统高速缓存总计信息。

列
　描述

size
　在 OPCACHEMAX 配置参数中指定的大小

alloc
　数据库服务器分配给高速缓存的 1 KB 大小的块数

avail
　未使用的 alloc 部分（以 KB 计）

number
　数据库服务器成功放入高速缓存而未溢出的简单大对象数目

kbytes
　数据库服务器放入高速缓存而未溢出的简单大对象的 KB 数

number
　数据库服务器写入登台区域 BLOB 空间的简单大对象数目

kbytes
　数据库服务器写入登台区域 BLOB 空间的简单大对象的 KB 数

虽然 size 输出指示了在配置参数 OPCACHEMAX 中指定的内存量，但是数据库服务器直至需要时才会将内存分配至 OPCACHEMAX。因此，alloc 输出仅反映已处理的最大简单大对象的 1 KB 大小的块数。当 alloc 和 avail 输出相等时，高速缓存为空。

显示的第二部分描述以下用户高速缓存总计信息。

列
　描述

SID
　用户的会话标识

user
　客户机的用户标识

size
　在 GBASEDBTOPCACHE 环境变量中指定的大小（如果已设置）
　如果您未设置 GBASEDBTOPCACHE 环境变量，那么数据库服务器使用您在配置参数 OPCACHEMAX 中指定的大小。

number
数据库服务器放入高速缓存而未溢出的简单大对象数目

kbytes
数据库服务器放入高速缓存而未溢出的简单大对象的 KB 数

number
数据库服务器写入登台区域 BLOB 空间的简单大对象数目

kbytes
数据库服务器写入登台区域 BLOB 空间的简单大对象的大小（以 KB 计）

使用 gcheck -pB 确定 BLOB 页填充度

gcheck -pB 命令显示描述 Blob 页平均充满度的统计信息。如果您发现大量简单大对象的统计信息中显示的充满度百分比很低，那么更改 Blob 空间中的 Blob 页的大小会提高数据库服务器的性能。

使用数据库名称或表名作为参数运行 gcheck -pB。以下示例检索存储在 **stores_demo** 数据库的 **sriram.catalog** 表中的所有简单大对象的存储信息：

```
gcheck -pB stores_demo:sriram.catalog
```

使用 gcheck -pe 监视 BLOB 空间使用情况

gcheck -pe 命令提供了有关 BLOB 空间使用情况的信息：

- 存储 TEXT 和 BYTE 数据的表名（按块）
- 所使用的磁盘页（非 BLOB 页）数（按表）
- 剩余的可用磁盘页数（按块）
- 所用的开销页数（按块）

以下示例显示样本 gcheck -pe 输出。

```
BLOBSpace Usage Report:    fstblob                Owner:    GBase 8s    Created: 03/01/08

 Chunk: 3     /home/server/blob_chunk                Size        Used        Free
                                                     4000         304        3696

          Disk usage for Chunk 3                                Total Pages
          ------------------------------------------------      -----------------------
          OVERHEAD                                                    8
          stores_demo:chrisw.catalog                                296
          FREE                                                     3696
```

使用 gcheck -pT 监视数据库空间中的简单大对象

使用 gcheck -pT 可监视数据库空间来确定 TEXT 和 BYTE 数据使用的数据库空间的页数。

此命令采用数据库名称或表名作为参数。对于数据库中的每个表，或对于特定表，数据库服务器显示常规表空间报告。

常规报告之后是扩展数据块中页使用的详细分类（按页类型）。请参阅 **Type** 列以获取有关 TEXT 和 BYTE 数据的信息。

数据库服务器可以将一个以上的简单大对象存储在同一 BLOB 页上。因此，可对表空间中存储 TEXT 或 BYTE 数据的页数计数，但无法估计表中简单大对象的数目。

以下示例显示样本输出。

```
TBLSpace Usage Report for mydemo:chrisw.catalog

     Type              Pages        Empty      Semi-Full       Full      Very-Full
     ----------------  -----------  ---------  ----------  ----------  -----------
     Free                    7
     Bit-Map                 1
     Index                   2
     Data (Home)             9
     Data (Remainder)        0          0           0           0           0
     Tblspace BLOBs          5          0           0           1           4
                        -----------
     Total Pages            24

     Unused Space Summary

          Unused data bytes in Home pages                 3564
          Unused data bytes in Remainder pages               0
          Unused bytes in Tblspace Blob pages             1430

Index Usage Report for index    111_16 on mydemo:chrisw.catalog

                          Average      Average
     Level      Total     No. Keys   Free Bytes
     --------   --------  ---------- --------------
      1          1         74          1058
     --------   --------  ---------- --------------
     Total       1         74          1058

Index Usage Report for index    111_18 on mydemo:chrisw.catalog

                          Average      Average
     Level    Total      No. Keys   Free Bytes
    --------  --------   ---------- --------------
      1         1         74          984
    --------  --------   ---------- --------------
     Total      1         74          984
```

监视智能大对象空间

要在智能大对象空间中监视的最重要的区域之一是元数据页使用。在创建智能大对象空间时，您指定元数据区域的大小。而且，每次向智能大对象空间添加块时，您可以指定要添加至块的元数据空间。

如果您尝试插入新的智能大对象，但没有可用的元数据空间，就会接收到一个错误。管理员必须监视元数据空间可用性以防止此情况发生。

使用以下命令监视智能大对象空间。

命令	描述
gstat -g smb s	显示系统中所有智能大对象空间的存储属性： • 智能大对象空间名称、标志、所有者 • 日志记录状态 • 平均智能大对象大小 • 第一个扩展数据块大小、下一个扩展数据块大小和最小扩展数据块大小 • 最大 I/O 存取时间 • 锁定方式
gstat -g smb c	显示以下有关每个智能大对象空间的信息： • 块数和智能大对象空间名称 • 块大小和路径名 • 用户数据页和可用用户数据页的总数 • 每个用户数据和元数据区域中的页的位置和数量
gcheck -ce gcheck -pe	显示以下有关智能大对象空间使用的信息： • 存储智能大对象数据的表名（按块） • 所使用的磁盘页（非智能大对象页）数（按表） • 剩余的可用用户数据页数（按块） • 剩余的保留用户数据页数（按块） • 所使用的元数据页数（按块） 输出提供以下总计： • 所有用户数据区域和元数据区域的已用页的总数。系统向用户数据区域和元数据区域的总计添加了 53 页以用于保留区域。 • 在元数据区域中剩余的可用页数 • 在所有用户数据区域中剩余的可用页数
gstat -d	显示以下有关每个智能大对象空间中的块的信息： • 在元数据区域和在用户数据区域中的每个智能大对象空间块中的可用智能大对象页数 • 在元数据区域和在用户数据区域中的每个智能大对象空间块中的总智能大对象页数
gcheck -cs gcheck -ps	验证并显示有关智能大对象空间的元数据区域的信息。
gcheck -cS	显示有关智能大对象空间的智能大对象扩展数据块和用户数据区域的信息。

命令	描述
gcheck -pS	显示有关智能大对象空间的智能大对象扩展数据块、用户数据区域和元数据区域的信息。

gstat -d 选项

使用 gstat -d 选项显示以下有关每个智能大对象空间中块的信息：

- 在元数据区域和在用户数据区域中的每个智能大对象空间块中的可用智能大对象页数
- 在元数据区域和在用户数据区域中的每个智能大对象空间块中的总智能大对象页数

有关 gstat -d 输出的示例，请参阅《GBase 8s 管理员参考》中有关 gstat 实用程序的信息。

要查明已使用空间的总量，请运行 gcheck -pe 命令。有关更多信息，请参阅 gcheck -ce 和 gcheck -pe 选项。

在备份执行删除的逻辑登录且释放智能大对象页之后，gstat -d 命令才会将智能大对象页注册为可用页。因此，如果删除了 25 个智能大对象并立即运行 gstat -d，gstat 输出中将不会包含新释放的空间。

gcheck -ce 和 gcheck -pe 选项

运行 gcheck -ce 可显示用户数据区域中每个智能大对象空间块的大小、已使用空间的总量和可用空间量。gcheck -pe 选项显示与 gcheck -ce 所显示信息相同的信息，另外还显示块使用的详细列表。首先列出数据库空间，然后列出智能大对象空间。**-pe** 输出提供以下有关智能大对象空间使用的信息：

- 存储智能大对象数据的表名（按块）
- 所使用的磁盘页（非智能大对象页）数（按表）
- 剩余的可用用户数据页数（按块）
- 所使用的元数据页数（按块）

输出提供以下总计：

- 用户数据区域、元数据区域和保留区域的已用页的总数。

  系统向用户数据区域和元数据区域的总计添加了额外的 53 页以用于保留区域。

- 在元数据区域中剩余的可用页数
- 在用户数据区域中剩余的可用页数

提示： gcheck -pe 选项以数据库服务器页的形式提供了有关智能大对象空间的使用情况的信息，而非智能大对象页的形式。

以下示例显示样本输出。在该示例中，智能大对象空间 **s9_sbspc** 的已用总页数为 214、元数据区域中的可用页数为 60 以及用户数据区域的可用页数为 726。

```
Chunk Pathname                                Size        Used        Free
```

```
        2 /ix/ids9.2/./s9_sbspc                    1000          940           60

      Description                                    Offset       Size
      ------------------------------------------------          ----------   ----------
      RESERVED PAGES                                  0           2
      CHUNK FREELIST PAGE                             2           1
      s9_sbspc:'gbasedbt'.TBLSpace                    3           50
      SBLOBSpace LO [2,2,1]                           53          8
      SBLOBSpace LO [2,2,2]                           61          1
...
SBLOBSpace LO [2,2,79]                                168         1
      SBLOBSpace FREE USER DATA                       169         305
      s9_sbspc:'gbasedbt'.sbspace_desc                474         4
      s9_sbspc:'gbasedbt'.chunk_adjunc                478         4
      s9_sbspc:'gbasedbt'.LO_hdr_partn                482         8
      s9_sbspc:'gbasedbt'.LO_ud_free                  490         5
      s9_sbspc:'gbasedbt'.LO_hdr_partn                495         24
      FREE                                            519         60
      SBLOBSpace FREE USER DATA                       579         421

      Total Used:                                     214
      Total SBLOBSpace FREE META DATA:                60
      Total SBLOBSpace FREE USER DATA:                726
```

可以将 CHECK EXTENTS 用作与 gcheck -ce 等同的 SQL 管理 API *命令*。有关使用 SQL API 命令的信息，请参阅使用 SQL 管理 API 执行远程管理和《GBase 8s 管理员参考》。

gcheck -cs 选项

gcheck -cs 和 gcheck -Cs 选项验证智能大对象空间的元数据区域。下面显示了 **s9_sbspc** 的 **-cs** 输出的一个示例。如果不在命令行上指定智能大对象空间名称，那么gcheck 会检查和显示所有智能大对象空间的元数据。

使用 gcheck -cs 输出来查看在元数据区域中留下的空间量。如果该区域已满，那么为元数据区域分配另一具有足够空间的块。要查明元数据区域中的已用页数，可将 **Used** 列中的数值进行总计。 要查明元数据区域中的可用页数，可将 **Free** 列中的数值进行总计。

例如，根据下图中显示的字段值，**s9_sbspc** 的元数据区域中的已用总页数为 33 个 2 KB 大小的页（或 66 KB）。元数据区域包含的可用总页数为 62（或 124 KB）。

```
Validating space 's9_sbspc' ...

SBLOBspace Metadata Partition                 Partnum      Used      Free
s9_sbspc:'gbasedbt'.TBLSpace                  0x200001       6        44
s9_sbspc:'gbasedbt'.sbspace_desc              0x200002       2         2
s9_sbspc:'gbasedbt'.chunk_adjunc              0x200003       2         2
s9_sbspc:'gbasedbt'.LO_hdr_partn              0x200004      21        11
s9_sbspc:'gbasedbt'.LO_ud_free                0x200005       2         3
```

gcheck -ps 选项

gcheck -ps 选项验证并显示有关智能大对象空间分区中元数据区域的信息。下面显示了 **s9_sbspc** 的 **-ps** 输出的一个示例。如果您未在命令行中指定智能大对象空间名称，那么 gcheck 验证并显示所有存储空间的表空间信息。

要监视可用元数据空间量，请运行以下命令：

```
gcheck -ps spacename
```

-ps 输出包含有关元数据区域中的锁定粒度、**partnum**、已分配和已使用的页数、扩展数据块大小以及行数。使用 gcheck -ps 输出来查看在元数据区域中留下的空间量。如果该区域已满，那么为元数据区域分配另一具有足够空间的块。

如果您为包含存储了智能大对象的表的数据库空间运行 gcheck -ps，那么可查明表中的行数。

```
Validating space 's9_sbspc' ...

TBLSpace Report for
    TBLspace Flags                2801        Page Locking
                                              TBLspace use 4 bit bit-maps
                                              Permanent System TBLspace

    Partition partnum             0x200001
    Number of rows                92
    Number of special columns     0
    Number of keys 0
    Number of extents             1
    Current serial value          1
    First extent size             50
    Next extent size              50
    Number of pages allocated     50
    Number of pages used          6
    Number of data pages          0
    Number of rows                0
    Partition lockid              2097153
    Optical Cluster Partnum       -1
    Current SERIAL8 value         1
    Current REFID value           1
    Created                       Thu May 24 14:14:33 2007
```

监视元数据和用户数据区域

数据库服务器保留 40% 的用户数据区域作为*保留区域*。数据库服务器将该保留空间用于元数据或用户数据。元数据区域在向该智能大对象空间添加智能大对象时被耗尽。当数据库服务器耗尽元数据或用户数据空间时，它会将一块保留空间移至相应区域。

当所有的保留区域均用尽时，即使用户数据区域包含可用空间，数据库服务器也无法移动空间至元数据区域。

1. 当您将智能大对象添加到智能大对象空间时，请使用 gcheck -pe 或 gstat -g smb c 监视元数据区域、用户数据区域和保留区域中的空间。

   有关示例，请参阅 gcheck -ce 和 gcheck -pe 选项。

   数据库服务器打印有关从保留区域分配至元数据区域的页数消息。

2. 在智能大对象空间耗尽元数据和保留区域中空间之前向智能大对象空间添加另一个块。

   有关更多信息，请参阅向智能大对象空间添加块。

3. 数据库服务器在将空间从保留区域移至元数据或用户数据区域时编写 FREE_RE 和 CHKADJUP 日志记录。

有关更多信息，请参阅计算智能大对象空间元数据的大小。

3.6.16 存储优化

数据压缩和合并方法可将数据所用磁盘空间降到最低。

下表描述可用于减小数据所用磁盘空间量的方法。

表 1. 存储优化方法

存储优化方法	用途	何时使用
压缩数据	压缩表或分段行中的数据，从而减小所需磁盘空间量 压缩数据之后，还可以合并表或分段中剩余的可用空间，并将这些可用空间返还给数据库空间。	希望减少表中的数据大小时
重新打包数据	合并表和分段中的可用空间	压缩数据之后，或希望单独合并表或分段中的可用空间时
收缩数据	将可用空间返还给数据库空间	压缩或重新打包数据之后，或希望单独将可用空间返还给数据库空间时
为表扩展数据块取消分段	使邻接的合并扩展数据块中的数据行靠得更紧密	经常更新的表在多个非邻接扩展数据块之间变得分散时

可自动执行以上任意一个或所有方法。

可在 OpenAdmin Tool (OAT) 中执行这些方法，也可以使用 SQL 管理 API 函数进行编程来执行这些方法。

自动优化数据存储

可通过更新 **auto_crsd** 调度程序任务，为表和扩展数据块配置自动压缩、收缩、重新打包和取消分段。

必须以用户 **gbasedbt** 或其他授权用户身份连接 **sysadmin** 数据库。

要启用和配置自动存储优化，请执行以下操作：

1. 通过在 **ph_task** 表上使用 UPDATE 语句将 **tk_enable** 列的值设置为 T，从而启用 **auto_crsd** 调度程序任务。例如，以下语句启用 **auto_crsd** 任务：

```
UPDATE ph_task
  SET tk_enable = 'T'
  WHERE tk_name = 'auto_crsd';
```

2. 可选：通过在 **ph_threshold** 表上使用 UPDATE 语句来将阈值的 **value** 列设置为 F，从而禁用单个操作：
   - AUTOCOMPRESS_ENABLED：控制压缩
   - AUTOREPACK_ENABLED：控制重新打包
   - AUTOSHRINK_ENABLED：控制收缩
   - AUTODEFRAG_ENABLED：控制取消分段

   例如，以下语句只禁用 **auto_crsd** 任务的取消分段存储操作：

```
UPDATE ph_threshold
  SET value = 'F'
  WHERE name = 'AUTODEFRAG_ENABLED';
```

3. 可选：通过在 **ph_threshold** 表上使用 UPDATE 语句来更改阈值的 **value** 列的值，从而更改单个操作的阈值：
   - AUTOCOMPRESS_ROWS：压缩的阈值是未压缩行数。缺省阈值为 10 000 行。在未压缩行数超过 10 000 时，将压缩表。
   - AUTOREPACK_SPACE：重新打包的阈值为非邻接空间的百分比。缺省值为 90%。在表占用的空间中 90% 以上的空间非连续时，将重新打包该表。
   - AUTOSHRINK_UNUSED：表或分段的收缩阈值是已分配的未用空间的百分比。缺省值为 50%。在 50% 以上的已分配空间未被使用时，将收缩表或分段。
   - AUTODEFRAG_EXTENTS：表或分段扩展数据块的取消分段阈值是扩展数据块的数量。缺省值为 100。在扩展数据块的数量超过 100 时，将对表或分段执行取消分段操作。

   例如，以下语句将压缩阈值更改为 5000 行：

```
UPDATE ph_threshold
  SET value = '5000'
  WHERE name = 'AUTOCOMPRESS_ROWS';
```

或者，除了运行已调度的自动压缩任务之外，还可以通过 SQL 管理 API create dictionary 命令或初始 compress 命令来启用自动压缩。对表或表分段运行 SQL 管理 API create dictionary 命令或初始 compress 命令时，将启用后续数据装入（包含 2000 行或更多行数据）的自动压缩。

您还可以使用 CREATE TABLE 语句的 COMPRESSED 选项来支持在将数据装入表或表分段时对大量行内数据执行自动压缩。CREATE TABLE 语句的 COMPRESSED 选项不支持对数据库空间或索引中的简单大对象执行自动压缩。

对分区取消分段

可以通过对分区取消分段以将非邻接的扩展数据块合并，从而提高性能。

随着时间推移，经常更新的表可能变为分段表，从而当服务器每次访问表时，导致性能下降。对表取消分段可使数据行更紧密的集合在一起，并避免分区标题页溢出问题。对索引取消分段会让条目更集中，这会提高存取表信息的速度。

要确定表、索引或分区有多少扩展数据块，可运行 gcheck -pt and pT 命令。

要为表、索引或分区取消分段，请运行带 defragment 自变量或 defragment partnum 自变量的 SQL 管理 API task() 或 admin() 函数，并指定要取消分段的表名、索引或分区号。

限制和注意事项

对分区取消分段之前，请查看以下重要注意事项：

- 提交取消分段请求之后，无法停止该请求。
- 不能为以下对象取消分段：
    - 伪表，如虚拟表接口 (VTI) 表
    - 包含虚拟索引接口 (VII) 索引的表
    - 包含 B 型树函数索引的表
    - 临时表
    - 排序文件
    - 光盘 BLOB 文件
- 不得对要取消分段的表或分区发出冲突操作。必须先完成第一个操作，再启动第二个操作。如果第一个操作仍在运行，对第二个操作的请求将返回错误。以下列表中包含冲突操作示例：
    - 一个分区一次只能运行一个取消分段请求。
    - 一个数据库空间一次只能运行一个取消分段请求。
    - 如果表或分区上正在运行 DROP TABLE 或 ALTER FRAGMENT 之类的 DDL 语句，那么不能为表取消分段。
    - 不能为正在截断的表取消分段。
    - 不能为正在压缩或解压缩的表取消分段。
    - 不能为正在运行联机索引构建的表取消分段。
    - 不能为设置了互斥存取的表取消分段。

如果完成取消分段请求时发生问题，将向联机日志文件发送错误消息。

数据压缩

可以压缩和解压缩表与分段中的行数据和/或数据库空间中的简单大对象。可以压缩表与表分段中的数据，以减少占用的磁盘空间量。还可以合并表或分段中的可用空间，然后将这些可用空间返还给数据库空间。 压缩数据之前，可估计可以节省的磁盘空间量。

压缩数据、整合数据以及返还可用空间具有以下益处：

- 显著节省磁盘存储空间
- 减少压缩的分段的磁盘使用量

- 显著节省逻辑日志使用量，这样在完成压缩操作后可节省额外空间并可以防止高吞吐量 OLTP 的瓶颈。
- 页面读取更少，因为更多行可以置于一个页面中
- 缓冲池更小，因为更多数据置于同一大小的池中
- I/O 活动减少，因为：
    - 与未压缩行相比，更多压缩行置于一个页面中
    - 压缩行的插入、更新和删除操作的日志记录更小
- 可以压缩按时间分段数据的不常访问的较旧分段，而让频繁访问的更多近期数据处于未压缩的格式
- 可以释放表不再需要的空间
- 备份与复原更快

I/O 绑定表（例如，高速缓存命中率低的表）非常适合压缩。在 OLTP 环境中，压缩 I/O 绑定表可以提高性能。

但是，如果应用程序运行时具有很高的缓冲区高速缓存命中率，并且高性能比空间使用量更重要，那么可能不希望压缩数据，因为压缩可能会稍微降低性能。

查询可以访问压缩表中的数据。

由于与未压缩数据相比，压缩的数据占用更少页面且每页中有更多行，所以查询优化器可能在压缩后选择不同的计划。

如果使用 Enterprise Replication (ER)，那么压缩一个复制服务器上的数据不会影响其他任何复制服务器上的数据。

如果使用高可用性数据复制 (HDR)，源表中的压缩数据在目标表中也处于压缩状态。不能在 HDR 辅助服务器、RS 辅助服务器或 SD 辅助服务器上执行压缩操作，因为 HDR 目标服务器必须具有与源服务器相同的数据和物理布局。

不能使用 gload 和 gunload 实用程序将压缩后的数据从一个数据库移至另一个数据库。在使用 gload 和 gunload 实用程序之前，您必须将压缩表和分段中的数据解压缩。

可使用 OpenAdmin Tool (OAT) 执行与压缩相关的操作，也可通过运行包含带有压缩参数的 SQL 管理 API 命令的 SQL 语句来执行这些操作。 对表或表分段运行 SQL 管理 API create dictionary 命令或初始 compress 命令时，将启用后续数据装入（包含 2000 行或更多行数据）的自动压缩。

在 GBase 8s V8.8 中，必须运行启用压缩的 SQL 管理 API 命令才可以对表或分段进行压缩。 如果启用了压缩，那么必须遵循以下 GBase 8s 还原过程，以更改回没有数据压缩功能的服务器版本，并且必须解压缩或删除任何压缩表和分段。

取代压缩的主要方法是购买更多物理存储器。取代减少 IO 绑定工作负载内的瓶颈的主要方法是购买更多物理内存，以便扩展缓冲池。

可压缩的数据

可以压缩行中的数据以及数据库空间中的简单大对象。但是，您可能并不希望压缩可压缩的所有类型的数据。

具有经常重复的长模式的表或表分段数据非常适合压缩。特定类型的数据（如文本）可能比其他类型的数据（如数字数据）更适合压缩，因为文本之类的数据类型可能包含更长、更频繁重复的模式。

但是，不能仅根据数据类型来预测压缩率。例如：

- 不同语言或字符集的文本的压缩率可能不同，即使文本存储在 CHAR 或 VARCHAR 列中也是如此。
- 大部分由零构成的数字数据压缩率可能很高，而可变数字数据较多则压缩率可能不高。
- 具有大量空格的数据压缩率较高
- 已使用其他某种算法压缩的数据和已加密的数据的压缩率可能不高。例如，行中的图像和声音采样可能已经压缩，因此再次压缩这些数据不会再额外节省任何空间。

GBase 8s 可压缩任意数据类型的组合，因为它将所有数据视为未结构化的字节序列来压缩。因此，服务器可以压缩跨多个列的模式，如城市、州和邮政编码的组合。（服务器对字节序列的解压缩顺序与数据压缩之前存在的顺序相同。）

可以压缩：

- 数据行的内容，包括跨多页的行的任何剩余片段，以及包含在逻辑日志记录中的行的映像。
- 数据库空间中的简单大对象（例如，XML 文件、PDF 文件或其他包含文本的文档）

压缩仅应用于数据行的内容，包括跨多页的行的剩余片段，以及包含在逻辑日志记录中的行的映像。

不能压缩的数据

不能压缩某些类型的表和分段中的行数据。不能压缩索引中的数据，也不能压缩某些类型的表和分段中的数据。

不能压缩以下对象的行中的数据：

- **sysmaster**、**sysutils**、**sysuser**、**syscdr** 和 **syscdcv1** 数据库中的表或分段
- 目录
- 临时表
- 虚拟表接口表
- 表空间 **tblspace**（这些是隐藏分段，每个数据库空间一个隐藏分段。每个表包含有关数据库空间内所有分段的元数据。）
- 内部分区表
- 字典表，每个数据库空间一个字典表（这些表包含该数据库空间内已压缩的分段或表的压缩字典，以及有关这些字典的元数据。）
- 索引

- 在其中正在执行联机索引构建的表

也不能压缩 BLOB 空间中的简单大对象。

不会将压缩应用于索引数据、存储在行外的 LOB 数据，或其他任何形式的非行数据。

加密的数据、已经通过其他算法压缩的数据，以及没有长重复模式的数据压缩效果不佳或不压缩。 请尽量不要将包含压缩效果不佳的数据的列放在具有常用模式的列之间，以防止跨列模式的可能损坏。

如果存储 XML 数据时将第一部分放在行中，将其余部分放在该行外，那么压缩将仅应用于存储在行中的数据。

仅当压缩行的映像小于未压缩映像时，GBase 8s 才会压缩行的映像。即便压缩后的行仅比其未压缩映像稍小，但节省的少量空间也可以让服务器在页中放入更多行。

压缩估算

压缩表或表分段之前，可估算压缩数据后将可节约的空间量。显示的比率根据行数据的样本估算。实际节约的空间比率可能稍有不同。

GBase 8s 通过以下方法估算压缩率：对行数据随机采样（使用与字典构建相同的采样算法），然后计算以下项的大小总和：

- 未压缩行映像
- 使用新压缩字典（由估算压缩命令临时创建的压缩字典）得到的压缩行映像
- 使用存在的现有字典得到的压缩行映像（如果没有现有字典，该值将与未压缩行映像的大小之和相同。）

获得的空间实际节约比率可能由于以下原因而有所差异：

- 发生了较小的采样错误。
- 估算是基于行的原始压缩能力。

  例如，服务器通常尝试将整个行放入一页中。因此，如果每个未压缩行几乎填满一整页，并且压缩率低于 50%，那么每个压缩行仍将填满大半页，而即使在压缩之后，服务器仍倾向于将每行放在单独的一页上。在这种情况下，尽管估算的压缩率可能为 45%，实际节约的空间则可能为 0%。

  每个未压缩行填充页的一半稍多。每个未压缩行将占用一整页，因为两个整行的总大小已超出一页。 例如，估算的压缩率可能为 5%，但是该压缩率可能刚够将每个行收缩到小于半页。所以，压缩之后，两行可放入一页，实际节约的空间可能为 50%。

实际获得的压缩可能与估算不一样，因为 GBase 8s 在一页中存储的行数不能超过 255。所以小行或大页会减少压缩可获得的节约总量。 例如，如果压缩前一页中有 200 行，无论压缩后行有多小，最大有效压缩率均接近 20%，因为压缩后一页中只能有 255 行。

如果使用的页面大于最小页大小，那么可通过切换到更小的页来增加实现的压缩空间节约量，从而：

- 不再达到 255 行的限制。

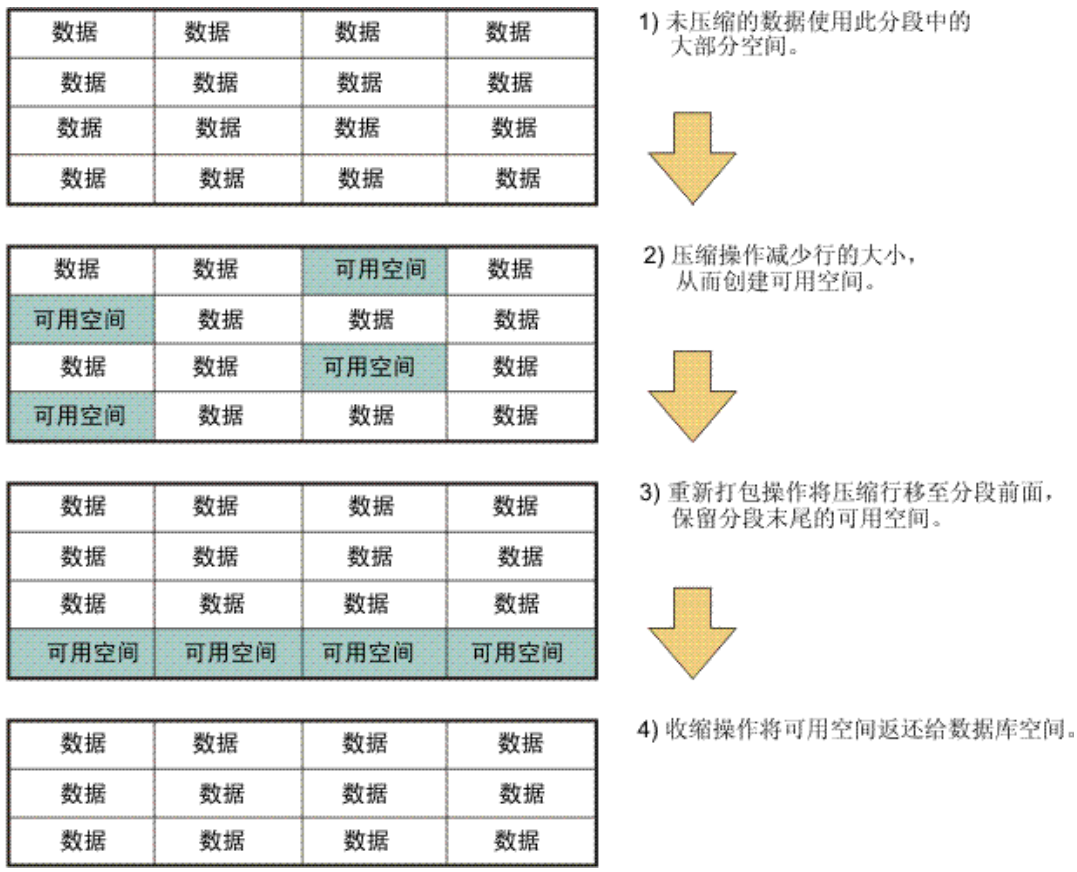
- 如果仍然达到了该限制，页中的未使用空间将更少。

如果压缩操作包含重新打包操作、收缩操作或重新打包并收缩操作，那么可节约更多（或更少）空间。仅当一页中的压缩行数超过了未压缩行数，重新打包操作才能节约更多空间。如果重新打包操作释放出空间，那么收缩操作可以在数据库空间级别节约空间。

压缩数据和存储优化的图示

本主题中的图示显示使用分段中大部分空间的未压缩数据、压缩数据时创建的可用空间、执行重新打包操作之后移动到分段末尾的可用空间，以及执行收缩操作之后留在分段中的数据。

图: 压缩和存储优化处理期间分段中的数据



压缩和解压缩数据

此方案显示如何运行 SQL 管理 API 命令来管理压缩和存储优化。

在该方案中，用户 **mario** 所有的数据库 **music** 内有一个表 **rock**。

先决条件:

- 表的每个分段中必须至少有 2000 行，而不仅是整个表总共只有 2000 行。
- 您必须可以连接到 **sysadmin** 数据库（缺省情况下只有用户 **gbasedbt** 可连接），并且必须是 DBSA。

- 如果希望运行的任何工作负载（包括但不限于压缩操作）耗用日志文件的速度超过每 30 秒一个文件，那么必须将日志配置为比当前大小更大。
- 如果希望运行的任何工作负载（包括但不限于这些压缩操作）耗用日志文件的速度超过每 30 秒一个文件，请将日志配置为比当前大小更大。compress、repack、repack_offline、uncompress 和 uncompress_offline 操作可能会耗用大量日志。
- 如果在使用 GBase 8s，必须先启用压缩，才能压缩数据。要启用压缩，请运行以下命令：

  ```
  EXECUTE FUNCTION task("enable compression");
  ```

  无需启用压缩，即可使用 estimate_compression 自变量估算压缩数据可节约的空间量。如果要使用 shrink、repack 或 repack_offline 自变量在不压缩任何行数据的情况下从表中释放空间，也无需启用压缩。

要同时压缩行数据以及数据库空间中的简单大对象：

要压缩和解压缩行数据，请执行以下操作：

1. 不确定是否要压缩 **rock** 表，那么可以运行以下命令来检查压缩该表可节约的空间量：

   ```
   EXECUTE FUNCTION task("table estimate_compression", "rock", "music",
   mario");
   ```

   复审生成的报告，其中指示可为 **rock** 表当前使用的空间节约 75%。您决定压缩表。

2. 压缩数据之前，希望创建压缩字典，其中包含 GBase 8s 用于压缩 **rock** 表中的数据的信息。运行以下命令

   ```
   EXECUTE FUNCTION task("table create_dictionary", "rock", "music", "mario");
   ```

   如果不作为单独的步骤来创建压缩字典，GBase 8s 将在您压缩数据时自动创建字典。

3. 您决定要压缩 **rock** 表中的数据以及数据库空间中的简单大对象，合并这些数据，然后将可用空间返还给数据库空间。 可以运行以下命令：

   ```
   EXECUTE FUNCTION task("table compress repack shrink", "rock", "music",
   "mario");
   ```

   如果决定仅压缩行数据或仅压缩数据库空间中的简单大对象，请通过将 rows 或 blobs 插入到单词 compress 之后或字符串 compress repack shrink 之后来调整命令。例如：

   - 要仅压缩行数据，请指定：

     ```
     EXECUTE FUNCTION task("table compress rows","rock","music","mario");
     ```

   - 要仅压缩行数据，然后重新打包和收缩数据，请指定：

     ```
     EXECUTE FUNCTION task("table compress repack shrink rows",
     "rock","music","mario";
     ```

   - 要仅压缩数据库空间中的简单大对象，请指定：

     ```
     EXECUTE FUNCTION task("table compress blobs","rock","music","mario");
     ```

   压缩现有行和简单大对象之后，GBase 8s 会合并表末尾留下的可用空间，然后从表中除去这些可用空间，从而将该空间返还给数据库空间。

如果简单大对象或行 /* 在压缩后并不会变小，那么数据库服务器不会对其进行压缩。

4. 现在假设必须解压缩数据。可以运行以下命令：
```
EXECUTE FUNCTION task("table uncompress", "rock", "music", "mario");
```
5. 您希望除去压缩字典。
   - 验证 Enterprise Replication 是否不需要该字典。

     如果 Enterprise Replication 确实需要这些已解压缩或已删除的表和分段的压缩字典，请勿除去这些字典。
   - 归档其中包含带有压缩字典的表或分段的数据库空间。
   - 运行以下命令：
```
EXECUTE FUNCTION task("table purge_dictionary", "rock", "music", "mario");
```

按照压缩和解压缩行中数据以及数据库空间中简单大对象的相同方式来压缩和解压缩表分段中的数据以及数据库空间中的简单大对象，但是不同之处在于，运行的命令具有以下格式：

```
EXECUTE FUNCTION task("fragment compression_arguments", "partnum_list");
```

如果要在不执行压缩或重新压缩的情况下合并可用空间或返还可用空间，可运行一个命令，指示服务器重新打包、收缩或重新打包并收缩。

启用压缩

在 GBase 8s V8.8 中，必须先运行用于启用压缩的 SQL 管理 API admin() 或 task()命令，才能对实例中的表或表分段执行首次压缩或解压缩操作。

在 GBase 8s V8.8 中，压缩将自动启用。

您必须可以连接到 **sysadmin** 数据库（缺省情况下只有用户 **gbasedbt** 可连接），并且必须是 DBSA。

但是，在启用压缩之前，可以估算压缩率、合并可用空间（重新打包）和将可用空间返还给表或分段（收缩）。

要启用压缩，请执行以下操作：

运行带 enable compression 自变量的 admin() 或 task() 函数。

例如，指定：

```
EXECUTE FUNCTION task(“enable compression”);
```

启用压缩之后，可创建压缩字典，或者也可以压缩表的行或者分段表的特定分段或所有分段的行。

估算压缩率

如果压缩表或者分段表的特定分段或所有分段，可估算可节约的空间百分比。该命令将显示可用于确定是否要压缩或重新压缩行数据的信息。

先决条件：您必须可以连接到 **sysadmin** 数据库（缺省情况下只有用户 **gbasedbt** 可连接），并且必须是 DBSA。

用于估算压缩率的命令总是同时估算新压缩率和当前压缩率。

有关压缩率和估算的一般信息，请参阅压缩率和压缩估算。

要估算压缩的优点，请执行以下操作：

运行带 estimate_compression 自变量的 admin() 或 task() 函数。

例如，为表使用以下语法：

```
EXECUTE FUNCTION task("table estimate_compression", "table_name",
        "database_name", "owner_name");
```

对于分段，请使用以下语法：

```
EXECUTE FUNCTION task("fragment estimate_compression""partnum_list");
```

以下示例显示了这样的一个命令：指示 GBase 8s 估算所有者为“wong”的“store123”数据库内名为“cash_transaction”的表的压缩优点。

```
EXECUTE FUNCTION task("table estimate_compression", "cash_transaction",
        "store123", "wong");
```

估算压缩操作显示可实现的估算压缩率、当前压缩率、对获得或失去的百分比的估算、每个分段的分区号，以及表的全名，包括数据库、所有者和表名。如果表未压缩，那么当前比率为 0.0%。

在以下示例中，已经压缩了第一个分段。未压缩第二个分段。如果重新压缩第一个分段，可以节约的空间会增加 0.4%。如果压缩第二个分段，可以增加 75.7%。

```
est      curr     change   partnum     coloff      table
-------- -------- -------- ----------- ------- ------------------------------------
75.7%    75.3%    +0.4     0x00200003  -1      store3:wg.cash_transaction
75.7%    0.0%     +75.7    0x00300002  -1      store3:wg.cash_transaction


 est      curr     change   partnum          table
 -------- -------- -------- ---------------- ---------------------------------------------
 75.7%    75.3%    +0.4     0x00200003       store123:wong.cash_transaction
 75.7%    0.0%     +75.7    0x00300002       store123:wong.cash_transaction
```

表和分段的压缩估算输出看起来几乎相同，不同之处在于表的输出始终显示表中的所有分段，而分段的输出仅显示指定分段的信息。

创建压缩字典

可根据现有行创建压缩字典，以供 GBase 8s 压缩表或表分段中的数据时使用。创建字典之后，GBase 8s 将使用该字典来压缩新插入或更新的行。

如果没有压缩字典，您也可以在运行压缩命令时创建。这两个命令的唯一差别是压缩命令还将压缩表或分段中的现有数据。

如果数据已装入表中，并且您要创建字典以用于在装入数据时压缩新数据，那么可以为该表创建压缩字典。

该表或分段必须至少包含 2000 行数据，然后数据库服务器才能创建压缩字典。如果创建字典时，表或分段没有足够的数据用于字典，那么该表或分段将设置为自动压缩。然后，在该表或分段中装入了足够的数据后，数据库服务器将自动创建该字典。

有关压缩字典的一般信息，请参阅压缩字典。

先决条件：

- 您必须可以连接到 **sysadmin** 数据库（缺省情况下只有用户 **gbasedbt** 可连接），并且必须是 DBSA。
- 如果要为分段创建压缩字典，分段中必须至少包含 2,000 行。如果要为表创建压缩字典，该表的每个分段必须至少包含 2,000 行。

要创建压缩字典，请执行以下操作：

运行带 table create_dictionary 或 fragment create_dictionary 自变量的 admin() 或 task() 函数。

例如：

- 为表指定信息，如下所示：

  ```
  EXECUTE FUNCTION task(“table create_dictionary”, “table_name”, “database_name”, “owner_name”);
  ```

  必须指定表名。数据库和所有者名称可选。如果不指定数据库或所有者名称，GBase 8s 将使用当前数据库和所有者名称。

- 为分段指定信息，如下所示：

  ```
  EXECUTE FUNCTION task(“fragment create_dictionary”, partnum_list”);
  ```

  partnum_list 列出了分区号，以空格分隔。

以下示例显示了这样的一个命令：指示 GBase 8s 为所有者为“shakar”的“music”数据库内名为“classical”的表创建压缩字典。

```
EXECUTE FUNCTION task("table create_dictionary","classical","music","shakar");
```

要在创建压缩字典之后压缩现有表或分段行中的数据，必须运行压缩命令。

只有在解压缩表或分段之后，才能删除压缩字典。

合并表中的可用空间

压缩表或分段时，可以合并（重新打包）表和分段中的可用空间，或单独合并可用空间，而不进行压缩。

可以使用 repack 或 repack_offline 自变量，以联机或脱机方式执行重新打包操作。repack_offline 操作与 repack 操作相同，但当 GBase 8s 在表或分段上持有互斥锁定期间执行操作时例外。 此操作在完成之前，会阻止对数据的所有其他访问。

如果执行 repack 操作时在表或分段中进行了轻量级追加，那么 repack 操作不会在表或分段的末尾完成空间合并。repack 操作无法完成的原因是，在已执行了 repack 操作的位置中添加了新的扩展数据块，因此空间无法返还给数据库空间。要完成 repack 进程，必须在完成轻量级追加活动之后，再运行一次 repack 操作。第二次运行的 repack 操作将在第一次 repack 操作工作的基础上构建。

完成 repack_offline 操作之前删除或禁用索引可以缩短服务器完成此操作所用时间量。之后，可以重新创建或重新启用索引，最好是利用 PDQ。先删除或禁用索引，再重新创建或启用索引，要比不这样做而直接完成 repack_offline 操作的速度更快。

先决条件：

- 您必须可以连接到 **sysadmin** 数据库（缺省情况下只有用户 **gbasedbt** 可连接），并且必须是 DBSA。
- 如果希望运行的任何工作负载（包括但不限于 repack 或 repack_offline 操作）耗用日志文件的速度超过每 30 秒一个文件，请将日志配置为比当前大小更大。

要合并表中的可用空间，请执行以下操作：

1. 运行使用 table repack、table repack_offline、fragment repack 或 fragment repack_offline 命令自变量的 admin() 或 task() 函数。

   例如，为表指定：

   ```
   EXECUTE FUNCTION task("table repack", "table_name","database_name",
           "owner_name");
   ```

   必须指定表名。数据库和所有者名称可选。 如果不指定数据库或所有者名称，GBase 8s 将使用当前数据库和所有者名称。

   例如，为分段指定：

   ```
   EXECUTE FUNCTION task("fragment repack_offline", "partnum_list");
   ```

   partnum_list 是属于同一个表的分区号的空格分隔列表。

2. （可选）扩展自变量以在以下任意组合中包含 compress 和 shrink：
   - compress repack
   - compress repack shrink
   - repack shrink

以下示例显示了这样的一个命令：指示 GBase 8s 合并所有者为“bob”的一个“music”数据库中名称为“opera”的表中的可用空间。

```
EXECUTE FUNCTION task("table repack","opera","music","bob");
```

以下示例显示了这样的一个命令：指示 GBase 8s 以脱机方式合并所有者为“bob”的一个“music”数据库中名称为“folk”的表中的可用空间。

```
EXECUTE FUNCTION task("table repack_offline","folk","music","janna");
```

以下示例显示这样的一个命令：指示 GBase 8s 合并可用空间，并将空间返还给数据库空间中分区号为 14680071 的分段。

```
EXECUTE FUNCTION task("fragment repack shrink", "14680071");
```

可以取消使用 compress 自变量的命令，例如在 DB-Access 中输入 CTRL-C。之前中断命令之后，可重新发出带 repack 或 repack_offline 自变量的命令。（压缩和重新打包操作将记录，但是在较小的部分中运行。）

将可用空间返还给数据库空间

压缩、重新打包或压缩并重新打包表或分段时，可将可用空间返还给数据库空间（收缩空间）；也可在不压缩或重新打包的情况下单独返还可用空间。 返还可用空间将减小分段或表的总大小。

可在不影响表的分配策略的情况下，安全收缩整个表。例如，如果有一个分段表，一周的每一天都有一个分段，并且还有许多预分配的分段供将来使用，那么可在不影响该分配策略的情况下收缩该表。如果该表为空，GBase 8s 将把该表收缩为创建该表时指定的初始扩展数据块大小。

启动收缩操作时，GBase 8s 将缩小扩展数据块，如下所示：

- 将除第一个扩展数据块之外的其他所有扩展数据块尽量缩小。
- 如果整个表都在第一个扩展数据块中（例如，因为该表是空表），GBase 8s 将不把第一个扩展数据块收缩为小于使用 CREATE TABLE 语句创建该表时指定的扩展数据块大小。

可以使用 ALTER TABLE 语句的 MODIFY EXTENT SIZE 子句来减小当前扩展数据块大小。执行此操作之后，可重新运行收缩操作，以便将第一个扩展数据块收缩为新扩展数据块大小。

先决条件：您必须可以连接到 **sysadmin** 数据库（缺省情况下只有用户 **gbasedbt** 可连接），并且必须是 DBSA。

要将可用空间返还给数据库空间：

1. 运行带 table shrink 或 fragment shrink 自变量的 admin() 或 task() 函数。

   例如，为表指定：

   ```
   EXECUTE FUNCTION admin("table shrink", "table_name", "database_name",
           "owner_name");
   ```

   必须指定表名。数据库和所有者名称可选。 如果不指定数据库或所有者名称，GBase 8s 将使用当前数据库和所有者名称。

   例如，为分段指定：

   ```
   EXECUTE FUNCTION task("fragment shrink", "partnum_list");
   ```

   partnum_list 是属于同一个表的分区号的空格分隔列表。

2. 可以选择扩展自变量以在以下任意组合中包含 compress 和 repack：
   - compress repack shrink
   - compress shrink
   - repack shrink

以下示例显示了这样的一个命令：指示 GBase 8s 收缩所有者为“bob”的“music”数据库内名为“opera”的表。

```
EXECUTE FUNCTION task("table shrink","opera","music","bob");
```

以下示例显示了这样的一个命令：指示 GBase 8s 重新打包和收缩分区号为 14680071 的分段。

```
EXECUTE FUNCTION task("fragment repack shrink," "14680071");
```

解压缩数据

可解压缩之前压缩的表和分段。解压缩表或分段将停用对新的插入和更新操作的压缩、解压缩所有已压缩的行、停用压缩字典，并对不再适合其原始页面的行分配新页面。

可使用 uncompress 或 uncompress_offline 自变量，以联机或脱机方式解压缩。 脱机解压缩操作与解压缩的操作相同，不同之处是该操作在执行时对分段保持互斥锁定，以便在完成操作之前，避免其他对分段数据进行的所有访问。

完成脱机解压缩操作之前删除或禁用索引可以减少服务器完成此操作需要的时间量。之后，可以重新创建或重新启用索引，最好是利用 PDQ。先删除或禁用索引，再重新创建或启用，这种做法比不这样做而直接完成脱机重新打包或脱机解压缩操作的速度更快。

先决条件：

- 您必须可以连接到 **sysadmin** 数据库（缺省情况下只有用户 **gbasedbt** 可连接），并且必须是 DBSA。
- 必须压缩表或分段。
- 如果希望运行的任何工作负载（包括但不限于 uncompress 或 uncompress_offline 操作）耗用日志文件的速度超过每 30 秒一个文件，请将日志配置为比当前大小更大。

要解压缩表或分段中的数据，请执行以下操作：

运行带 table uncompress、table uncompress_offline、fragment uncompress 或 fragment uncompress_offline 命令自变量的 admin() 或 task() 函数。

为表指定信息，如下所示：

```
EXECUTE FUNCTION task(“table uncompress”, “table_name”,“database_name”,
        “owner_name”);
```

或

```
EXECUTE FUNCTION admin(“table uncompress_offline”, “table_name”,
        “database_name”, “owner_name”);
```

必须指定表名。数据库和所有者名称可选。如果不指定数据库或所有者名称，GBase 8s 将使用当前数据库和所有者名称。

为分段指定信息，如下所示：

```
EXECUTE FUNCTION task(“fragment uncompress”, “partnum_list”);
```

或

```
EXECUTE FUNCTION task(“fragment uncompress_offline”, “partnum_list”);
```

示例

以下示例显示了这样的一个命令：指示 GBase 8s 解压缩所有者为“mario”的“music”数据库内名为“rock”的表。

```
EXECUTE FUNCTION task("table uncompress","rock","music","mario");
```

以下示例显示了这样的一个命令：指示 GBase 8s 以脱机方式解压缩分区号为 14680071 的分段。

```
EXECUTE FUNCTION task("fragment uncompress_offline," "14680071");
```

如果已解压缩表，GBase 8s 将把该表的字典标记为不活动。GBase 8s 不会删除字典，因为 Enterprise Replication 函数会将字典用于较旧的日志。可删除不再需要的字典。

您可以取消带 uncompress 自变量的命令，如在 DB-Access 中按 CTRL-C。

之前中断命令之后，可重新发出带 uncompress 和 uncompress_offline 自变量的命令。（压缩、重新打包和解压缩操作将记录，但是在较小的部分中运行。）

删除压缩字典

可删除特定表或分段的不活动压缩字典，删除所有不活动压缩字典，或删除达到指定日期的所有不活动压缩字典。删除为表和分段创建的任何压缩字典之前，必须解压缩表和分段，以使字典处于不活动状态。

请勿除去 Enterprise Replication 需要的压缩字典。

不能删除为索引创建的压缩字典。删除索引时，数据库服务器会除去这些压缩字典。

先决条件：

- 您必须可以连接到 **sysadmin** 数据库（缺省情况下只有用户 **gbasedbt** 可连接），并且必须是 DBSA。
- 删除关联字典之前，请解压缩或删除表或分段。只能删除压缩表或分段不再使用的压缩字典。
- 确保 Enterprise Replication 函数未将压缩字典用于较旧的日志。
- 归档包含具有压缩字典的表或分段的任何数据库空间，即使表或分段中的数据已解压缩并且字典不再处于活动状态。

要删除不再需要的一个或多个压缩字典，请执行以下操作：

运行带以下自变量的 admin() 或 task() 函数：

- table purge_dictionary 或 fragment purge_dictionary 命令自变量，用于删除特定的不活动字典。
- compression purge_dictionary 命令自变量，用于删除所有字典。
- compression purge_dictionary 命令自变量和日期，用于删除该日期及之前创建的所有字典。

任何可根据您的语言环境转换为 DATE 数据类型格式的日期均可使用。 例如，可以指定 01/31/2012、01/31/12 或 Jan 31, 2012。

示例 1：运行以下命令可除去所有者为“arlette”的“music”数据库中表“latin”的不活动字典。

```
EXECUTE FUNCTION task(“table purge_dictionary”, “latin”, “music”,”arlette”);
```

示例 2：运行以下命令可除去 2011 年 3 月 8 日及之前创建的所有字典：

```
EXECUTE FUNCTION task(“compression purge_dictionary”, "03/08/11");
```

压缩表时，数据库服务器会创建一个内部分区用于保存压缩字典。解压缩表和清除字典时，数据库服务器会除去该字典。 但是，内部分区不会除去。如果为保存字典而创建的内部分区位于第二个块上，那么可能会有一个块只包含内部分区。 如果发生此情况，将无法删除该块。可以备份数据库空间，然后将其删除。

移动压缩数据

可使用 High-Performance Loader (HPL) 或其他任何 GBase 8s 实用程序（gunload 和 gload 实用程序除外）来移动压缩数据。

不能使用 gunload 和 gload 实用程序将压缩后的数据从一个数据库移动到另一个。在使用 gunload 和 gload 实用程序之前，您必须将压缩表和分段中的数据解压缩。

dbexport 实用程序用于解压缩压缩的数据。因此，如果数据库包含带有压缩数据的表或分段，那么必须在使用 **dbimport** 实用程序导入数据之后重新启用压缩和重新压缩。

dbexport 实用程序用于解压缩压缩的数据。因此，如果数据库包含带有压缩数据的表或分段，那么必须在使用 **dbimport** 实用程序导入数据之后重新压缩。

B 型树索引压缩和存储优化

可以压缩拆离的 B 型树索引。您还可以合并索引中的可用空间，然后可将索引末尾的可用空间返还给数据库空间。压缩数据之前，可估计可以节省的磁盘空间量。

压缩索引、合并数据并返还可用空间会带来以下益处：

- 可将更多数据放入缓冲区高速缓存
- 释放内存用于其他用途

使用 CREATE INDEX 命令创建新索引时，可以对其进行压缩。

可以压缩位于分段或非分段表上的现有拆离 B 型树索引。使用 SQL 管理 API 命令来压缩现有索引。

您不能压缩以下类型的索引：

- 不是 B 型树索引的索引
- 连接的 B 型树索引
- 虚拟 B 型树索引
- 键数少于 2000（这是可压缩索引需要的最小键数）的索引

先决条件：要能够进行压缩，索引必须至少有 2000 个键。

压缩操作仅压缩索引的叶子（底层）。

不能将已压缩的索引解压缩。如果不再需要已压缩的索引，可以删除该索引，然后将其重新创建为未压缩的索引。

压缩率

压缩率取决于压缩的数据。GBase 8s 使用的压缩算法是基于字典的算法，该算法对找到的最常用的数据模式执行操作，方法是在构建该字典时采样的数据中按长度加权。

如果典型数据分发与创建字典时采样的数据存在偏差，压缩率可能降低。

最大压缩率可能为 90%。这是因为任何顺序的字节的最大可能压缩率是通过以下方法得出的：将每组 15 个字节替换为一个 12 位的符号数，从而生成占原始映像大小 10% 的压缩映像。但是，并不容易获得 90% 的压缩率，因为 GBase 8s 会向每个压缩映像添加一个字节的元数据。

压缩字典

对于每个压缩分段和每个压缩的非分段表，都存在一个单独的压缩字典。对于每个压缩分段、每个压缩的非分段表、数据库空间中的每个压缩简单大对象和每个压缩索引分区，都存在一个单独的压缩字典。每个压缩字典是由分段或表数据中经常出现的模式和替换这些模式的符号数构成的库。

压缩字典是使用从至少包含 2,000 行的分段或非分段表随机采样的数据构建的。如果分段或表中包含的行数不足 2,000，那么 GBase 8s 不会构建压缩字典。

压缩字典最多可存储 3,840 个模式，每个模式的长度可以为 2 到 15 个字节。（超过 7 个字节的模式将减少字典可包含的模式的总数。）这些模式中的每一个都通过压缩行中的 12 位符号数来表示。要进行压缩，输入行映像中的一系列字节必须精确匹配字典中的某个完整模式。如果行与字典匹配的模式不足，那么可能无法压缩，因为不完全匹配的输入行中的每个字节在压缩映像中都将替换为 12 位（1.5 个字节）。

GBase 8s 尝试捕获最佳的可压缩模式（模式频率乘以长度）。数据通过以下方法压缩：将出现的模式替换为字典中的相应符号数，并将出现的不匹配任何模式的字节替换为特别保留的符号数。

数据库空间中表或分段的所有字典都存储在该数据库空间中一个隐藏的字典表内。**sysmaster** 数据库中的 **syscompdicts_full** 表和 **syscompdicts** 视图提供有关压缩字典的信息。

通常需要大约 100 KB 的空间来存储压缩分段或表的压缩字典。因此，太小的表不适合压缩，因为可能无法通过压缩行来收回足够抵销压缩字典存储花费的空间。

此外，GBase 8s 不能将单个行压缩到长度小于 4 个字节。这是因为服务器必须留出一些空间，以免行映像稍后增长到超过了页面可容纳的程度。因此，不得尝试压缩带有包含 4 个或更少字节的行的分段或非分段表。

可查看的压缩信息

可以使用 GBase 8s 实用程序、**sysmaster** 数据库表和 **sysmaster** 视图来显示压缩统计信息、有关压缩字典的信息，以及压缩字典。

表 1. 用于显示压缩信息的实用程序、**sysmaster** 表和视图

实用程序、表或视图	描述
gcheck -pT 选项	在输出的“Compressed Data Summary”部分中显示有关任何压缩项的统计信息。如果未压缩任何项，那么输出中不会显示“Compressed Data Summary”部分。 例如，对于行数据，gcheck -pT 显示表或表分段中任何压缩行的数量，以及压缩的表或表分段行的百分比。
glogdump -c 选项	使用压缩字典来扩展压缩的数据，并显示压缩日志记录的未压缩内容。
gstat -g dsk 选项	显示有关当前运行的压缩操作的进度的信息。
gstat -g ppd 选项	显示有关当前打开的压缩分段（也称为分区）存在的活动压缩字典的信息。此选项显示的信息与 **sysmaster** 数据库中的 **syscompdicts** 视图显示的信息相同。
sysmaster 数据库中的 **syscompdicts_full** 表	显示有关压缩字典和压缩字典二进制对象的元数据。 只有用户 gbasedbt 可访问此表。
sysmaster 数据库中的 **syscompdicts** 视图	与 **syscompdicts_full** 表显示的信息相同，不过出于安全性原因，其中不包含 **dict_dictionary** 列，该列中包含压缩字典二进制对象。

可以使用 UNLOAD 语句，将压缩字典从 **syscompdicts_full** 表卸载到压缩字典文件，如下所示：

```
UNLOAD TO 'compression_dictionary_file'
    SELECT * FROM sysmaster:syscompdicts_full;
```

有关 **glogdump** 实用程序、**gstat -g dsk** 选项、**gstat -g ppd** 选项、**gcheck -pT** 选项、**syscompdicts_full** 表和 **syscompdicts** 视图的更多信息，请参阅《GBase 8s 管理员参考》。

3.6.17 将数据装入表

您可以按以下方法将数据 装入现有的表中。

装入数据的方法	TEXT 或 BYTE 数据	CLOB 或 BLOB 数据	引用

装入数据的方法	TEXT 或 BYTE 数据	CLOB 或 BLOB 数据	引用
DB-Access LOAD 语句	是	是	《GBase 8s SQL 指南：语法》中的 LOAD 语句
dbload 实用程序	是	是	
dbimport 实用程序	是	是	
GBase 8s ESQL/C 程序	是	是	《GBase 8s ESQL/C 程序员手册》
使用 EXTERNAL 源表插入 MERGE	是	是	《GBase 8s SQL 指南：语法》
gload 实用程序	否	否	
onpladm 实用程序	是，精致方式	是，精致方式	

重要： 数据库服务器在数据已装入数据库后，不包含任何压缩 TEXT 和 BYTE 数据的机制。

3.7 使用外部表移动数据

可使用外部表装入和卸载数据库数据。

您可以发出一系列用于执行以下功能的 SQL 语句：

- 将运作数据高效传输到其他系统或从其他系统传入
- 以 GBase 8s 内部数据格式跨平台传输数据文件
- 使用数据库服务器在定界 ASCII、固定 ASCII 与 GBase 8s 内部（原始）表示法之间转换数据
- 使用 SQL INSERT 和 SELECT 语句来指定数据库表中新列的数据映射
- 提供并行标准 INSERT 操作，即可在不删除索引的情况下装入数据
- 使用命名管道来支持在存储设备中装入和卸载数据，这些存储设备包括磁带机和直接网络连接
- 维护运行期间的装入和卸载统计信息的记录
- 执行快速（高速）和高级（数据检查）传输

可使用 DB-Access 发出 SQL 语句，或将其嵌入 ESQL/C 程序中。

3.7.1 外部表

外部表是不受 GBase 8s 数据库服务器管理的数据文件。外部表的定义包括数据格式化类型、外部数据描述字段和全局参数。

为了将外部数据映射到内部数据，数据库服务器将外部数据视为外部表。将外部数据视为表是一种很有用的方法，可用于将数据移入或移出数据库，并指定数据库的转换。

数据库服务器运行装入任务时，将从外部源读取数据，并执行创建行所需的转换，然后将行插入表中。数据库服务器将错误写入拒绝文件。

如果不能转换外部表中的数据，可指定数据库服务器将记录写入拒绝文件，并且包含失败的原因。为此，请在 CREATE EXTERNAL TABLE 语句中指定 REJECTFILE 关键字。

数据库服务器提供了若干不同的转换机制，这些转换机制在数据库服务器内部执行，从而在执行转换任务期间提供最大性能。数据库服务器优化固定格式和定界格式的 ASCII 与 GBase 8s 数据表示法之间的数据转换。

要执行定制转换，可创建一个过滤器程序，用于将转换后的数据写入命名管道。数据库服务器然后以一种常用格式从命名管道读取其输入。

3.7.2 定义外部表

要定义外部表，需要使用 SQL 语句来描述数据文件，定义表，然后指定要装入或卸载的数据。

要设置装入和卸载任务，需要发出以下一系列 SQL 语句：

- CREATE EXTERNAL TABLE，用于描述要装入或卸载的数据文件
- CREATE TABLE，用于定义要装入的表
- INSERT...SELECT，用于执行装入和卸载

以下步骤概述装入过程：

1. CREATE EXTERNAL TABLE 语句描述各个外部文件的位置和外部数据的格式，其位置可以是磁盘或来自管道（磁带机或直接网络连接）。以下示例描述的是 CREATE EXTERNAL TABLE 语句：

```
CREATE EXTERNAL TABLE emp_ext
( name CHAR(18) EXTERNAL CHAR(18),
hiredate DATE EXTERNAL CHAR(10),
address VARCHAR(40) EXTERNAL CHAR(40),
empno INTEGER EXTERNAL CHAR(6) )
USING (
    FORMAT 'FIXED',
     DATAFILES
        ("DISK:/work2/mydir/emp.fix")
     );
```

2. CREATE TABLE 语句定义要装入的表。以下样本 CREATE TABLE 语句定义 **employee** 表：

```
CREATE TABLE employee
    FRAGMENT BY ROUND ROBIN IN dbspaces;
```

3. INSERT...SELECT 语句用于映射外部数据与数据库表之间的移动。 以下样本 INSERT 语句从外部表装入 **employee** 表：

```
INSERT INTO employee SELECT * FROM emp_ext
```

要点： 如果指定多个 *INSERT...SELECT 语句*来卸载数据，那么每个后续 INSERT 语句将覆盖数据文件。请对数据文件使用绝对路径。

将数据装入数据库时，SELECT 子句的 FROM 表部分是 CREATE EXTERNAL 语句定义的外部表。将数据卸载到外部文件时，SELECT 子句控制从数据库检索数据。

与 TEMP 表不同，外部表在删除之前会在目录中保留定义。创建外部表时，可保存数据的外部描述，以便复用。将表卸载为 GBase 8s 内部数据表示法后，此操作特别有用，因为稍后可使用相同的外部表描述来重新装入这些数据。

外部表定义中包含定义外部数据文件中的数据所需的全部信息，如下所示：

- 外部数据中的字段的描述。
- DATAFILES 子句。

  该子句指定：
  - 数据文件是位于磁盘上还是命名管道上。
  - 文件的路径名。
- FORMAT 子句。

  该子句指定外部数据文件中的数据格式化类型。数据库服务器转换若干数据格式的外部数据，这些格式包括定界和固定 ASCII，以及 GBase 8s 内部格式。
- 影响数据格式的任何全局参数。

如果将外部表直接映射到定界格式的内部数据库表中，可使用 CREATE EXTERNAL TABLE 语句定义列，并添加子句 SAMEAS *内部表*，而无需显式枚举列。

3.7.3 将列映射到其他列

如果数据文件应该让字段按照其他顺序排列（例如，**empno**、**name**、**address**、**hiredate**），可使用 INSERT 语句映射列。首先创建包含列的表，而这些列的排列顺序是其在外部文件中发现时的排列顺序。

```
CREATE EXTERNAL TABLE emp_ext
    (
    f01 INTEGER,
    f02 CHAR(18),
    f03 VARCHAR(40),
    f04 DATE
    )
USING (
    DATAFILES ("DISK:/work2/mydir/emp.dat"),
    REJECTFILE "/work2/mydir/emp.rej"
    );
INSERT INTO employee (empno, name, address, hiredate)
    SELECT * FROM emp_ext;
```

通过这种方法将对插入列进行映射，以匹配外部表的字段顺序。

另一种对列进行重新排序的方法是使用 SELECT 子句来匹配数据库表的顺序。

```
INSERT INTO employee
    SELECT f02, f04, f03, f01 FROM emp_ext;
```

3.7.4 从命名管道装入数据和将数据卸载到命名管道

可使用命名管道（也称为先进先出，FIFO）数据文件，来从非标准设备（如磁带机）装入和卸载到非标准设备。

与普通的操作系统文件不同，命名管道没有 2 GB 大小限制。操作系统打开并检查命名管道的文件末尾，其方法与普通文件不同。

使用命名管道装入数据

可使用命名管道从外部表装入数据。

要使用命名管道从外部表装入数据，请执行以下步骤：

1. 在 SQL 中 CREATE EXTERNAL TABLE 语句的 DATAFILES 子句内，指定命名管道。
2. 创建在 DATAFILES 子句中指定的命名管道。

   使用操作系统命令创建命名管道。

   使用带 **-p** 选项的 mknod UNIX™ 命令创建命名管道。要避免在 UNIX 上出现妨碍管道打开的问题，请为管道读程序和管道写程序启动单独的 UNIX 进程，或使用 O_NDELAY 标志集来打开管道。
3. 使用读取命名管道的程序打开命名管道。
4. 执行 SQL 中的 INSERT 语句。

   ```
   INSERT INTO employee SELECT * FROM emp_ext;
   ```

重要： 如果在执行 INSERT 语句之前不创建和打开命名管道，INSERT 将成功执行，但是不装入任何行。

FIFO 虚拟处理器

数据库服务器使用 FIFO 虚拟处理器 (VP) 来读写命名管道上的外部表。

缺省 FIFO 虚拟处理器数为 1。

数据库服务器为您在 CREATE EXTERNAL TABLE 语句的 DATAFILES 子句中指定的每个命名管道使用一个 FIFO VP。例如，假设您使用以下 SQL 语句定义外部表：

```
CREATE EXTERNAL TABLE ext_items
    SAMEAS items
    USING (
        DATAFILES("PIPE:/tmp/pipe1",
                  "PIPE:/tmp/pipe2",
                  "PIPE:/tmp/pipe3"
```

```
    ));
```

如果对 FIFO VP 使用缺省值 1，数据库在读取完 **pipe1** 中的所有数据之前不会从 **pipe2** 读取，在读取完 **pipe2** 中的所有数据之前不会从 **pipe3** 读取。

使用命名管道卸载数据

可使用命名管道将数据从数据库卸载到外部表。

要使用命名管道将数据卸载到外部表，请执行以下步骤：

1. 在 SQL 的 CREATE EXTERNAL TABLE 语句或 SELECT INTO EXTERNAL 语句的 DATAFILES 子句中，指定命名管道。

   ```
   DATAFILES ("PIPE:/usr/local/TAPE")
   ```

2. 创建在 DATAFILES 子句中指定的命名管道。使用操作系统命令创建命名管道。
3. 使用写入命名管道的程序打开命名管道。
4. 将数据卸载到命名管道。

   ```
   CREATE EXTERNAL TABLE emp_ext
   ( name CHAR(18) EXTERNAL CHAR(20),
          hiredate DATE EXTERNAL CHAR(10),
          address VARCHAR(40) EXTERNAL CHAR(40),
          empno INTEGER EXTERNAL CHAR(6) )
       USING (
           FORMAT 'FIXED',
           DATAFILES
   ("PIPE:/usr/local/TAPE")
             );

   INSERT INTO emp_ext SELECT * FROM employee;
   ```

重要： 如果在执行 SELECT 或 INSERT 语句之前不创建和打开命名管道，卸载将失败，并生成 *ENXIO* 错误消息（没有这样的设备或地址）。

使用 PIPE 选项将数据从一个实例复制到另一个

可使用命名管道将数据从一个 GBase 8s 实例复制到另一个，而无需将数据写入中间文件。

可使用命名管道将数据从一个 GBase 8s 实例卸载，然后装入另一个实例，而无需将数据写入中间文件。也可使用命名管道将数据从一个表复制到同一个 GBase 8s 实例上的另一个表。 在以下示例中，数据将从一个实例上的源表复制到第二个实例上的目标表。

必须首先根据所用硬件平台使用以下命令之一创建命名管道。在本示例中，命名管道称为 pipe1。

```
% mkfifo /work/pipe1

% mknod /work/pipe1
```

执行以下步骤，将数据从源实例上的表复制到同一台计算机上目标实例中的表。

1. 在源实例上创建源表。在本示例中，源表称为 source_data_table：

```
CREATE TABLE source_data_table
(
  empid      CHAR(5),
  empname    VARCHAR(40),
  empaddr    VARCHAR(100)
);
```

2. 在源实例上创建外部表。在本示例中，外部表称为 ext_table：

```
CREATE EXTERNAL TABLE ext_table
(
  empid      CHAR(5),
  empname    VARCHAR(40),
  empaddr    VARCHAR(100)
)
USING
(DATAFILES
  (
    'PIPE:/work/pipe1'
  )
);
```

3. 在目标实例上创建目标表。在本示例中，目标表称为 destin_data_table：

```
CREATE TABLE destin_data_table
(
  empid      CHAR(5),
  empname    VARCHAR(40),
  empaddr    VARCHAR(100)
);
```

4. 在目标实例上创建外部表。在本示例中，外部表称为 ext_table：

```
CREATE EXTERNAL TABLE ext_table
(
  empid      CHAR(5),
  empname    VARCHAR(40),
  empaddr    VARCHAR(100)
)
USING
(DATAFILES
  (
    'PIPE:/work/pipe1_1'
  )
);
```

5. 从 UNIX™ shell 运行以下命令。该命令将数据从 /work/pipe1 重定向到 /work/pipe1_1

```
cat /work/pipe1 > /work/pipe1_1
```

6. 在目标实例上运行以下命令，以将数据从命名管道定向到目标表：

```
INSERT INTO destin_data_table SELECT * FROM ext_table;
```

7. 在源实例上运行以下命令，以将数据假脱机到命名管道：

```
INSERT INTO ext_table SELECT * FROM source_data_table;
```

可使用多个管道，方法是在 DATAFILES 子句中插入多个 PIPE 语句，并为每个语句创建一个命名管道。

3.7.5 监视装入或卸载操作

可监视外部表的装入或卸载操作的状态。

您可能希望在以下情况监视装入或卸载操作：

- 如果希望经常装入和卸载同一个表，以构建数据集市或数据仓库，那么监视作业的进度可估算类似作业的时间，供将来使用。
- 如果从命名管道装入或卸载，那么监视 I/O 队列可确定是否有足够数量的 FIFO 虚拟处理器。

监视常用装入和卸载操作

可使用 gstat -g iof 命令查找要查看的文件中的全局文件描述符 (gfd)。然后使用 gstat -g sql 命令来监视装入和卸载操作。

以下示例显示了样本 gstat -g iof 命令输出。

```
AIO global files:

Gfd  path name     bytes read   page reads   bytes write   page writes   io/s
3    rootdbs       1918976      937          145061888     70831         36.5


     op type       count        avg. time
     seeks         0            N/A
     reads         937          0.0010
     writes        4088         0.0335
     kaio_reads    0            N/A
     kaio_writes   0            N/A
```

要确定装入或卸载操作是否可使用并行执行，请在执行 INSERT 语句之前执行 SET EXPLAIN ON 语句。SET EXPLAIN 输出显示以下计数：

- 优化器为 INSERT 语句选择的并行 SQL 运算符的数量
- 每个 SQL 运算符要处理的行数

要监视装入操作，请运行 gstat -g sql 来获取会话标识。

监视 FIFO 虚拟处理器

可使用 gstat 命令监视 FIFO VP 的有效使用。

可使用 gstat -g ioq 选项显示正在等待执行 I/O 请求的每个 FIFO 队列的长度。以下示例显示样本输出。

```
AIO I/O queues:
q name/id    len  maxlen  totalops  dskread  dskwrite  dskcopy
  fifo    0    0       0         0        0         0        0
  adt     0    0       0         0        0         0        0
msc       0    0       1       153        0         0        0
  aio     0    0       9      3499     1013        77        0
  pio     0    0       2         3        0         2        0
  lio     0    0       2      2159        0      2158        0
  gfd     3    0      16     39860       38     39822        0
  gfd     4    0      16     39854       32     39822        0
  gfd     5    0       1         2        2         0        0
  gfd     6    0       1         2        2         0        0
...
  gfd    19    0       1         2        2         0        0
```

以上示例中样本输出内的 **q name** 字段显示了队列的类型，如 **fifo** 表示 FIFO VP，或 **aio** 表示 AIO VP。 如果 **q name** 字段显示 **gfd** 或 **gfdwq**，说明这是其全局文件描述符与输出的 **id** 字段匹配的文件的队列。磁盘文件在一个队列中同时包含读写请求。每个磁盘文件在 gstat -g ioq 输出中显示一行。 管道有单独的读写队列。每个管道在输出中显示两行：**gfd** 针对读请求，**gfdwq** 针对写请求。

len 或 **maxlen** 字段的值最高为 4（对于装入）或 4 * number_of_writer_threads（对于卸载）。**xuwrite** 运算符控制写程序线程的数量。

请使用 **totalops** 字段中的值，而不是 **len** 或 **maxlen** 字段中的值，监视对文件或管道执行的读或写请求的数量。**totalops** 字段表示从该文件中读取了 34 KB 的数据，或将 34 KB 的数据写入了该文件。如果 **totalops** 不增加，说明已停止对文件或管道执行读或写操作（因为 FIFO VP 正忙）。

要提高性能，请使用 gadmin -p 命令增加更多 FIFO VP。FIFO VP 的缺省数量为 1。在该样本输出中，FIFO 队列内不包含任何数据。例如，如果通常定义两个以上的管道来装入或卸载，请使用以下样本 gadmin 命令增加 FIFO VP 的数量：

```
gadmin -p +2 FIFO
```

也可以使用 gadmin -p 命令除去 FIFO VP。但是，不能将 FIFO VP 的数量设置为低于 1。

有关更多信息，请参阅《GBase 8s 管理员参考》。

3.7.6 高可用性集群环境中的外部表

可按照几乎与主服务器上使用的相同方式，在辅助服务器上使用外部表。

可在主服务器和辅助服务器上执行以下操作：

- 将数据从数据库表卸载到外部表：

```
INSERT INTO external_table SELECT * FROM base_table WHERE ...
```

- 将数据从外部表装入到数据库表：

```
INSERT INTO base_table SELECT * FROM external_table WHERE ...
```

在 SDS、RSS 或 HDR 辅助服务器上装入数据的速度比在主服务器上装入数据的速度慢。

辅助服务器上不支持 CREATE EXTERNAL TABLE 语句和 SELECT ... INTO EXTERNAL ... 语句。

将数据从数据库表卸载到外部表时，将在辅助服务器上创建数据文件，但是不会在主服务器上创建。在辅助服务器上创建的外部表数据文件不会自动传输到主服务器，反之在主服务器上创建的外部表数据文件也不会自动传输到辅助服务器。

在主服务器上创建外部表时，仅外部表的模式会复制到辅助服务器，而不会复制数据文件。

要在主服务器与辅助服务器之间同步外部表，可以将外部表文件从主服务器复制到辅助服务器，或使用以下步骤：

1. 在主服务器上：
   a. 使用与外部表相同的模式创建临时表。
   b. 填充临时表：

```
INSERT INTO dummy_table SELECT * FROM external_table
```

2. 在辅助服务器上：

   使用以下命令填充外部表：

```
INSERT INTO external_table SELECT * FROM dummy_table
```

3.7.7 外部表的系统目录条目

可查询系统目录表，以确定外部表的状态。

每次创建外部表时，GBase 8s 都会更新 **sysexternal** 和 **sysextdfiles** 系统目录表。指定外部格式类型 (**fmttype**) FIXED 时，将更新 **sysextcols** 系统目录表。

表 1. 外部表系统目录条目

表名	描述
sysexternal	存储有关每个外部表的信息。
sysextdfiles	存储有关外部表数据文件的信息
sysextcols	存储有关类型为 FIXED 的外部表的信息

请参阅《GBase 8s SQL 指南：参考》以获取更多信息。

创建外部表时，将在 **systables** 系统目录中插入行；但是，除非在创建外部表时指定了 NUMROWS 子句，否则 **nrows**（行数）和 **npused**（所用数据页数）列可能无法精确反映外部表所用行数和数据页数。

创建外部表时如果没有为 NUMROWS 子句指定值，GBase 8s 就无法确定外部表中的行数，因为数据位于数据库外的数据文件中。GBase 8s 通过插入较大值 (MAXINT – 1) 来更新 **systables** 系统目录中的 **nrows** 列，并根据 **nrows** 的值计算所用数据页数。优化器稍后将使用 **npused** 和 **nrows** 中存储的值来确定最有效的执行计划。尽管无需精确指定 NUMROWS 子句，但是指定得越准确，**nrows** 和 **npused** 的值就越准确。

3.7.8 使用外部表时的性能注意事项

如果要使用 SQL 命令处理 ASCII 文件中的数据，或将数据从外部数据文件装入 RAW 数据库表，您可以使用外部表。

将信息装入数据库的方法有几种，包括：

- LOAD FROM ... INSERT INTO... DB-Access 命令
- dbimport 实用程序
- High-Performance Loader 实用程序
- 外部表

High Performance Loader 实用程序可在将外部数据装入到带索引的数据库表时提供最佳性能。

外部表可在将数据装入到不带索引的 RAW 表时提供最佳性能。

注： 装入数据前锁定外部表将提高装入性能

3.7.9 管理外部表装入和卸载操作产生的错误

可管理外部表装入和卸载操作期间发生的错误。

这些主题描述如何使用拒绝文件和错误消息来管理错误，以及如何恢复已装入到数据库的数据。

拒绝文件

装入期间具有转换错误的行将写入执行转换的服务器上的拒绝文件中。

CREATE EXTERNAL TABLE 语句中的 REJECTFILE 关键字用于确定指定给拒绝文件的名称。

可不使用拒绝文件，而是使用 CREATE EXTERNAL TABLE 语句中的 MAXERRORS 关键字来指定数据库服务器停止装入数据之前允许的错误数。(如果不设置 MAXERRORS 关键字，数据库将无视错误数，处理所有数据。)

数据库服务器将在开始执行装入时除去拒绝文件（如果有）。仅当装入期间发生错误时，才会重新创建和写入拒绝文件。

拒绝文件条目是包含以逗号分隔的以下字段的单行：

file name, record, reason-code, field-name: bad-line

file name
输入文件的名称

record
输入文件中的记录号，在该文件中检测到错误

reason-code
错误的描述

field-name
外部字段名称（行中的第一个错误即在此处出现）或 <none>（如果拒绝不特别针对特定列）

bad-line
坏行本身（仅对于定界或固定 ASCII 文件）

装入操作以 ASCII 格式写入 *file name*、*record*、*field-name* 和 *reason-code*。

bad-line 信息由输入文件的类型决定：

- 对于定界文件或固定文本文件，整个坏行将直接复制到拒绝文件中。但是如果定界格式表具有 TEXT 或 BYTE 列，那么拒绝文件中将不包含任何坏数据。装入操作将仅为每个拒绝的行生成一个标题。
- 对于 GBase 8s 内部数据文件，坏行不会放入拒绝文件中，因为您无法编辑文件中的二进制表示。但是，拒绝文件中仍然会报告 *file name*、*record*、*reason-code* 和 *field-name*，这样您可以隔离该问题。

以下类型的错误可能导致行被拒绝。

CONSTRAINT *constraint name*
违反了该约束。

CONVERT_ERR
有字段遇到转换错误。

MISSING_DELIMITER
找不到定界符。

MISSING_RECORDEND
找不到记录结尾。

NOT NULL
field-name 中找到了空值。

ROW_TOO_LONG
输入记录长度超过了 64 KB。
输入记录长度超过了 2 GB。

外部表错误消息

与外部表有关的大多数错误消息都在范围 -26151 到 -26199 之间。

其他消息为 -615、-999、-23852 和 -23855。在这些消息中，*n macro* 和 *r macro* 指从替换字符 %r(*first..last*) 生成的值。有关错误消息列表使用 finderr 实用程序。有关违例表错误消息的信息，请参阅《GBase 8s 管理员参考》。

外部表的表类型可恢复性

装入数据时，数据库服务器将检查表的可恢复性级别。

- 如果表类型为 RAW，数据库服务器可使用轻量级追加（快速）方式装入数据和处理检查约束。如果装入期间数据库服务器崩溃，将不回滚装入的数据，而表可能保持未知状态。
- 如果表类型为 STATIC，那么数据库服务器根本不能装入数据。
- 只有高级方式支持数据可恢复性。高级方式使用记录的常规插入。要在快速方式装入失败之后恢复数据，请还原为最近的 0 级备份。对于此级别的可恢复性，表类型必须为 STANDARD。

有关复原表类型的信息，请参阅《GBase 8s 备份与复原指南》。

4 日志记录和日志管理

4.1 日志记录

这些主题描述了 GBase 8s 数据库的日志记录，并解决了以下问题：

- 哪些数据库服务器进程需要日志记录？
- 什么是事务日志记录？
- 记录哪种数据库服务器活动？
- 什么是数据库日志记录状态？

- 谁可以设置或更改数据库日志记录状态？

单个数据库服务器实例所管理的所有数据库均将其日志记录存储在同一逻辑日志中，而无论它们是否使用事务日志记录。大多数数据库用户可能会关心事务日志记录是否已缓冲，或表是否使用日志记录。

如果您希望更改数据库日志记录状态，请参阅日志记录状态或方式的设置或更改。

4.1.1 需要日志记录的数据库服务器进程

在 GBase 8s 运行时（处理事务、跟踪数据存储和确保数据一致性），GBase 8s 会自动为所执行的一些操作生成*逻辑日志记录*。大多数时候，数据库服务器不会进一步使用逻辑日志记录。但是，当数据库服务器需要回滚事务，例如在系统发生故障后运行快速恢复时，逻辑日志记录将会很关键。逻辑日志记录是数据恢复机制的核心。

数据库服务器将逻辑日志记录存储在*逻辑日志*中。逻辑日志由*逻辑日志文件*组成，数据库服务器在磁盘上管理它们，直至这些文件已安全地转为脱机（*备份*）。数据库服务器管理员将保留已备份的逻辑日志文件，直至在数据复原期间需要用到这些文件，或直至管理员决定不再需要这些记录用于复原。请参阅逻辑日志获取有关逻辑日志的更多信息。

逻辑日志记录本身长度可变。这就增加了可写入逻辑日志缓冲区中页的逻辑日志记录数。但数据库服务器常常在页满之前清空逻辑日志缓冲区。有关逻辑日志记录格式的更多信息，请参阅《GBase 8s 管理员参考》中有关解释逻辑日志记录的主题。

数据库服务器在执行各种恢复数据和确保数据一致性的函数时使用逻辑日志记录，如下所示：

事务回滚
如果数据库正在使用事务日志记录，并且事务一定要回滚，那么数据库服务器使用逻辑日志记录来还原事务期间所作更改。有关更多信息，请参阅事务日志记录。

快速恢复
如果数据库服务器以不受控方式关闭，那么数据库服务器使用逻辑日志记录来恢复自未将最旧的更新清空到磁盘以来发生的所有事务，以及回滚任何未落实的事务。（当共享内存中和磁盘上的所有数据均相同时，它们*在物理上是一致的*。）如果要将整个数据库服务器返回至最近逻辑日志记录之时的逻辑一致性状态，数据库服务器会在快速恢复中使用逻辑日志记录。 （有关更多信息，请参阅检查点之后的快速恢复。）

数据复原
数据库服务器使用最近存储空间和逻辑日志备份来重新创建数据库服务器系统，直到最近备份的逻辑日志记录点为止。逻辑复原将应用上一次备份存储空间以来的所有日志记录。

延迟检查
如果事务使用 SET CONSTRAINTS 语句将检查设置为 DEFERRED，那么数据库服务器直至落实事务后才检查约束。如果在落实事务之时发生约束错误，那么数据库服务器使用逻辑日志记录回滚事务。有关更多信息，请参阅《GBase 8s SQL 指南：语法》中的『设置数据库对象方式』。

级联删除
对参阅约束的级联删除使用逻辑日志记录可确保如果在删除子行之前删除父行时系统发生故障，事务可回滚。有关主键和外键约束的信息，请参阅《GBase 8s SQL

指南：教程》。

分布式事务

分布式事务所涉及的每个数据库服务器将保留事务的逻辑日志记录。即使在执行该事务的其中一个数据库服务器上发生故障，该过程也可确保数据完整性和一致性。有关更多信息，请参阅两阶段落实和逻辑日志记录。

数据复制

Data Replication 使用逻辑日志记录在主数据库服务器和辅助数据库服务器上维护一致数据，从而使其中一个数据库服务器在另一数据库服务器发生故障时迅速用作备份数据库服务器。有关更多详细信息，请参阅数据复制的工作原理。

Enterprise Replication

必须对 Enterprise Replication 使用数据库日志记录，因为它从逻辑日志记录中复制数据。

4.1.2 事务日志记录

当数据库中的 SQL 数据操作语句生成逻辑日志记录时，人们就认为数据库或表具有或使用事务日志记录。

数据库日志记录*状态*指示数据库是否使用事务日志记录。*日志缓冲方式*指示数据库使用已缓冲还是未缓冲日志记录、或是符合 ANSI 标准的日志记录。有关更多信息，请参阅数据库日志记录状态和管理数据库日志记录方式。

在创建数据库时，指定数据库是否使用*事务日志记录*，如果使用，将使用哪种日志缓冲机制。在创建数据库之后，您可以（例如）关闭数据库日志记录或切换到已缓冲日志记录。即使您关闭了所有数据库的事务日志记录，数据库服务器也始终会记录一些事件。有关更多信息，请参阅 始终记录的活动和X/Open DTP 环境中的数据库日志记录。

您可以使用数据库中的日志记录表或非日志记录表。创建表的用户将指定表类型。即使您使用非日志记录表，数据库服务器也始终会记录一些事件。有关更多信息，请参阅 GBase 8s 的表类型。

4.1.3 SQL 语句和数据库服务器活动的日志记录

在数据库服务器中可能有以下三类记录活动：

- 始终记录的活动
- 使用事务日志记录为数据库记录的活动
- 未记录的活动

始终记录的活动

有些数据库操作始终生成逻辑日志记录（即使您关闭事务日志记录或使用非日志记录表）。

以下操作始终为永久表而记录：

- 某些 SQL 语句，其中包括 SQL 数据定义语句。
- 存储空间备份
- 检查点

- 对数据库服务器配置的管理更改（如添加块或数据库空间）
- 为表分配新扩展数据块
- 对数据库的日志记录状态的更改
- 智能大对象操作：
  - 创建
  - 删除
  - 分配与取消分配扩展数据块
  - 截断
  - 组合与分割块可用列表页
  - 更改 LO 头和 LO 引用计数
- 智能大对象空间元数据
- BLOB 空间

下表列出了生成操作的语句，即使关闭了事务日志记录也会记录这些操作。

- ALTER ACCESS_METHOD
- ALTER FRAGMENT
- ALTER FUNCTION
- ALTER INDEX
- ALTER PROCEDURE
- ALTER ROUTINE
- ALTER SECURITY LABEL COMPONENT
- ALTER SEQUENCE
- ALTER TABLE
- ALTER TRUSTED CONTEXT
- ALTER USER
- CLOSE DATABASE
- CREATE ACCESS_METHOD
- CREATE AGGREGATE
- CREATE CAST
- CREATE DATABASE
- CREATE DISTINCT TYPE
- CREATE EXTERNAL TABLE
- CREATE FUNCTION
- CREATE FUNCTION FROM
- CREATE INDEX
- CREATE OPAQUE TYPE
- CREATE OPCLASS
- CREATE PROCEDURE
- CREATE PROCEDURE FROM
- CREATE ROLE
- CREATE ROUTINE FROM
- CREATE ROW TYPE
- CREATE SCHEMA

- CREATE SECURITY LABEL
- CREATE SECURITY LABEL COMPONENT
- CREATE SECURITY POLICY
- CREATE SEQUENCE
- CREATE SYNONYM
- CREATE TABLE
- CREATE TEMP TABLE
- CREATE TRIGGER
- CREATE TRUSTED CONTEXT
- CREATE USER
- CREATE VIEW
- CREATE XADATASOURCE
- CREATE XADATASOURCE TYPE
- DROP ACCESS_METHOD
- DROP AGGREGATE
- DROP CAST
- DROP DATABASE
- DROP FUNCTION
- DROP INDEX
- DROP OPCLASS
- DROP PROCEDURE
- DROP ROLE
- DROP ROUTINE
- DROP ROW TYPE
- DROP SECURITY
- DROP SEQUENCE
- DROP SYNONYM
- DROP TABLE
- DROP TRIGGER
- DROP TRUSTED CONTEXT
- DROP TYPE
- DROP USER
- DROP VIEW
- DROP XADATASOURCE
- DROP XADATASOURCE TYPE
- GRANT
- GRANT FRAGMENT
- RENAME COLUMN
- RENAME DATABASE
- RENAME INDEX
- RENAME SECURITY
- RENAME SEQUENCE

- RENAME TABLE
- RENAME TRUSTED CONTEXT
- RENAME USER
- REVOKE
- REVOKE FRAGMENT
- TRUNCATE
- UPDATE STATISTICS
- SAVE EXTERNAL DIRECTIVES
- SET CONSTRAINTS
- SET Database Object Mode
- SET INDEXES
- SET TRIGGERS
- START VIOLATIONS TABLE
- STOP VIOLATIONS
- ALTER ACCESS_METHOD
- ALTER FRAGMENT
- ALTER FUNCTION
- ALTER INDEX
- ALTER PROCEDURE
- ALTER ROUTINE
- ALTER SECURITY LABEL COMPONENT
- ALTER SEQUENCE
- ALTER TABLE
- ALTER TRUSTED CONTEXT
- ALTER USER
- CLOSE DATABASE
- CREATE ACCESS_METHOD
- CREATE AGGREGATE
- CREATE CAST
- CREATE DATABASE
- CREATE DISTINCT TYPE
- CREATE EXTERNAL TABLE
- CREATE FUNCTION
- CREATE FUNCTION FROM
- CREATE INDEX
- CREATE OPAQUE TYPE
- CREATE OPCLASS
- CREATE PROCEDURE
- CREATE PROCEDURE FROM
- CREATE ROLE
- CREATE ROUTINE FROM
- CREATE ROW TYPE

- CREATE SCHEMA
- CREATE SECURITY LABEL
- CREATE SECURITY LABEL COMPONENT
- CREATE SECURITY POLICY
- CREATE SEQUENCE
- CREATE SYNONYM
- CREATE TABLE
- CREATE TEMP TABLE
- CREATE TRIGGER
- CREATE TRUSTED CONTEXT
- CREATE USER
- CREATE VIEW
- CREATE XADATASOURCE
- CREATE XADATASOURCE TYPE
- DROP ACCESS_METHOD
- DROP AGGREGATE
- DROP CAST
- DROP DATABASE
- DROP FUNCTION
- DROP INDEX
- DROP OPCLASS
- DROP PROCEDURE
- DROP ROLE
- DROP ROUTINE
- DROP ROW TYPE
- DROP SECURITY
- DROP SEQUENCE
- DROP SYNONYM
- DROP TABLE
- DROP TRIGGER
- DROP TRUSTED CONTEXT
- DROP TYPE
- DROP USER
- DROP VIEW
- DROP XADATASOURCE
- DROP XADATASOURCE TYPE
- GRANT
- GRANT FRAGMENT
- RENAME COLUMN
- RENAME DATABASE
- RENAME INDEX
- RENAME SECURITY

- RENAME SEQUENCE
- RENAME TABLE
- RENAME TRUSTED CONTEXT
- RENAME USER
- REVOKE
- REVOKE FRAGMENT
- TRUNCATE
- UPDATE STATISTICS
- SAVE EXTERNAL DIRECTIVES
- SET CONSTRAINTS
- SET Database Object Mode
- SET INDEXES
- SET TRIGGERS
- START VIOLATIONS TABLE
- STOP VIOLATIONS TABLE

使用事务日志记录为数据库记录的活动

如果数据库使用事务日志记录，那么以下 SQL 语句将生成一个或多个日志记录。如果这些语句回滚，那么回滚也会生成日志记录。

- DELETE
- FLUSH
- INSERT
- LOAD
- MERGE
- PUT
- SELECT INTO TEMP
- UNLOAD
- UPDATE

以下 SQL 语句在特殊情况下才生成日志。

表 1. 在特殊情况下生成日志的 SQL 语句。

SQL 语句	语句生成的日志记录
BEGIN WORK	除非数据库使用事务日志记录，否则返回错误。如果事务进行一些其他日志记录工作，那么产生一条日志记录。
COMMIT WORK	除非数据库使用事务日志记录，否则返回错误。如果事务进行一些其他日志记录工作，那么产生一条日志记录。
ROLLBACK WORK	除非数据库使用事务日志记录，否则返回错误。如果事务进行一些其他日志记录工作，那么产生一条日志记录。
EXECUTE	此语句是否生成日志记录取决于正在运行的命令。
EXECUTE FUNCTION	此语句是否生成日志记录取决于正在执行的函数。

SQL 语句	语句生成的日志记录
EXECUTE IMMEDIATE	此语句是否生成日志记录取决于正在运行的命令。
EXECUTE PROCEDURE	此语句是否生成日志记录取决于正在执行的过程。

未记录的活动

以下 SQL 语句无论数据库日志记录方式如何，均不生成日志记录。

- ALLOCATE COLLECTION
- ALLOCATE DESCRIPTOR
- ALLOCATE ROW
- CLOSE
- CONNECT
- DATABASE
- DEALLOCATE COLLECTION
- DEALLOCATE DESCRIPTOR
- DEALLOCATE ROW
- DECLARE
- DESCRIBE
- DISCONNECT
- FETCH
- FREE
- GET DESCRIPTOR
- GET DIAGNOSTICS
- INFO
- LOCK TABLE
- OPEN
- OUTPUT
- PREPARE
- RELEASE SAVEPOINT
- SAVEPOINT
- SELECT
- SET AUTOFREE
- SET COLLATION
- SET CONNECTION
- SET DATASKIP
- SET DEBUG FILE
- SET DEFERRED_PREPARE
- SET DESCRIPTOR
- SET ENCRYPTION PASSWORD
- SET ISOLATION

- SET LOCK MODE
- SET LOG
- SET OPTIMIZATION
- SET PDQPRIORITY
- SET ROLE
- SET SESSION AUTHORIZATION
- SET STATEMENT CACHE
- SET TRANSACTION
- SET Transaction Mode
- SET USER PASSWORD
- UNLOCK TABLE
- WHENEVER
- SET ENVIRONMENT
- SET EXPLAIN

对于临时数据库空间中的临时表，不记录任何内容，甚至始终记录的活动中所列的 SQL 语句也不记录。如果您在 DBSPACETEMP 中包含了临时（非日志记录）数据库空间，那么数据库服务器会将非日志记录表先放在这些临时数据库空间中。有关更多信息，请参阅临时表。

4.1.4 数据库日志记录状态

您必须对数据库使用事务日志记录以利用需要日志记录的数据库服务器进程中所列的所有功能。

数据库服务器管理的每个数据库均有日志记录状态。日志记录状态指示数据库是否使用事务日志记录以及（如果使用）数据库运用哪一种日志缓冲机制。要查明数据库的事务日志记录状态，请按监视数据库的日志记录方式中的说明，使用数据库服务器实用程序。数据库日志记录状态将指示下列日志记录类型中的任意一种：

- 未缓冲的事务日志记录
- 已缓冲的事务日志记录
- 符合 ANSI 的事务日志记录
- 无日志记录

所有逻辑日志记录在数据库服务器将其写入磁盘上的逻辑日志之前，均经过共享内存中的逻辑日志缓冲区。但是，数据库服务器清仓逻辑日志缓冲区的时刻对于已缓冲的事务日志记录和未缓冲的事务日志记录是不同的。

未缓冲的事务日志记录

如果事务是对使用未缓冲日志记录的数据库进行的，那么逻辑日志缓冲区中的记录可保证在落实处理期间写入磁盘。当应用程序在 COMMIT 语句之后（对于分布式事务，在 PREPARE 语句之前）重获控制权时，逻辑日志记录在磁盘上。一旦落实缓冲区中的任何事务（即，将落实记录写入逻辑日志缓冲区），数据库服务器就立即清仓记录。

当数据库服务器清仓缓冲区时，仅将已使用的页写入磁盘。已使用的页包含那些仅部分已满的页，但这样就浪费了一些空间。由于这个原因，相比在同一数据库服务器上的所有数据库均使用已缓冲的日志记录这种情况而言，磁盘上的逻辑日志文件会更快地填满。

未缓冲的日志记录对于大多数数据库而言是最好的选择，因为它保证所有已落实的事务可得以恢复。在发生故障的情况下，仅丢失在发生故障时未落实的事务。但是，有了未缓冲的日志记录，数据库服务器会更频繁地将逻辑日志缓冲区清仓到磁盘，而缓冲区将包含更多部分已满的页，因此它比已缓冲的日志记录会更快地填充逻辑日志。

已缓冲的事务日志记录

如果事务是对使用已缓冲日志记录的数据库进行的，那么记录尽可能久地保留（*已缓冲*）在逻辑日志缓冲区中。直至发生以下情况之一，这些记录才会从共享内存中的逻辑日志缓冲区刷新到磁盘上的逻辑日志：

- 缓冲区已满。
- 具有未缓冲日志记录的数据库上的落实清空了缓冲区。
- 出现检查点。
- 连接关闭。

如果您使用已缓冲日志记录并且发生了故障，那么不能期望数据库服务器恢复那些在发生故障时位于逻辑日志缓冲区中的事务。因此，可能会丢失一些已落实的事务。作为对该风险的补偿，变更期间的性能会稍有提高。只要您在发生故障的情况下可重新创建更新，那么已缓冲日志记录最适合用于频繁更新的数据库（这时更新速度很重要）。您可调整逻辑日志缓冲区的大小以便为您的系统在性能和因系统故障而丢失事务的风险之间找到可接受的平衡。

符合 ANSI 的事务日志记录

符合 ANSI 标准的数据库日志记录状态表明数据库所有者使用 MODE ANSI 关键字。符合 ANSI 标准的数据库对事务处理强制执行 ANSI 规则，始终使用未缓冲的事务日志记录。您无法更改符合 ANSI 标准的数据库的缓冲状态。

无数据库日志记录

如果您关闭了数据库的日志记录，那么将不记录事务，但会记录其他操作。有关更多信息，请参阅 始终记录的活动。通常，当您要装入数据或仅运行查询时，将关闭数据库的日志记录。

如果您对恢复源感到满意，可以决定不对数据库使用事务日志记录，从而减少数据库服务器处理量。例如，如果要将许多行从可恢复的源（如磁带或 ASCII 文件）装入数据库，那么可能不需要事务日志记录，并且在不使用事务日志记录的情况下装入会进行得更快。但是，如果其他用户在数据库中是活动的，那么您直到重新启动日志记录（这必须等待 0 级备份）后才会拥有其事务的逻辑日志记录。

具有不同日志缓冲状态的数据库

数据库服务器上的所有数据库使用相同的逻辑日志和相同的逻辑日志缓冲区。因此，对具有不同逻辑缓冲状态的各数据库的事务可写入相同的逻辑日志缓冲区。在这种情况下，如果事务是对具有已缓冲日志记录的数据库和对具有未缓冲日志记录的数据库而存在的，那么数据库服务器在缓冲区已满时或在具有未缓冲日志记录的数据库的事务完成时清空缓冲区。

X/Open DTP 环境中的数据库日志记录

X/Open 分布式事务处理 (DTP) 环境中的数据库必须使用*未缓冲日志记录*。未缓冲日志记录确保数据库服务器逻辑日志始终处于一致状态，并且可与事务管理器同步。如果在 X/Open DTP 环境中打开用已缓冲日志记录创建的数据库，数据库状态会自动更改为未缓冲日志记录。数据库服务器支持符合 ANSI 标准和不符合 ANSI 标准的数据库。有关更多信息，请参阅事务管理器。

4.1.5 日志记录状态或方式的设置或更改

用 CREATE DATABASE 语句创建数据库的用户为该数据库设立日志记录状态或缓冲方式。有关 CREATE DATABASE 语句的更多信息，请参阅《GBase 8s SQL 指南：语法》。

如果 CREATE DATABASE 语句不指定日志记录状态，那么创建无日志记录的数据库。

只有数据库服务器管理员可以更改日志记录状态。管理数据库日志记录方式描述了该主题。普通用户不能更改数据库日志记录状态。

如果数据库不使用日志记录，那么无需考虑是已缓冲还是未缓冲日志记录更为适当。如果您为数据库指定了日志记录但没有为其指定缓冲方式，那么缺省为未缓冲日志记录。

用户*可以*在*会话期间*从未缓冲日志记录切换到已缓冲（但不符合 ANSI 标准）日志记录，也可以从已缓冲日志记录切换到未缓冲日志记录。应用程序中的 SET LOG 语句将执行该切换。有关 SET LOG 语句的更多信息，请参阅《GBase 8s SQL 指南：语法》。

4.2 管理数据库日志记录方式

可以监视和修改数据库日志记录方式。

本节中的主题提供了有关以下内容的信息：

- 了解数据库日志记录方式
- 使用 gdblogmode 修改数据库日志记录方式
- 使用 gtape 修改数据库日志记录方式
- 使用 ON-Monitor 修改数据库日志记录方式
- 监视事务日志记录

作为数据库服务器管理员，您可以如下所示更改数据库的日志记录方式：

- 将事务日志记录从已缓冲更改为未缓冲。
- 将事务日志记录从未缓冲更改为已缓冲。
- 使数据库符合 ANSI 标准。
- 向数据库添加事务日志记录（已缓冲或未缓冲）。
- 结束数据库的事务日志记录。

有关数据库日志记录方式、何时使用事务日志记录以及何时缓冲事务日志记录的信息，请参阅日志记录。要查明数据库的当前日志记录方式，请参阅监视数据库的日志记录方式。

有关使用 SQL 管理 API 命令（而不是一些 gdblogmode 和 gtape 命令）的信息，请参阅使用 SQL 管理 API 执行远程管理和《GBase 8s 管理员参考》。

4.2.1 更改数据库日志记录方式

可以使用 gdblogmode 或 gtape 来添加或更改日志记录。然后使用 ON-Bar 或 gtape 来备份数据。在使用 ON-Bar 或 gtape 时，数据库服务器必须处于联机、管理或静默方式。

可以使用 gdblogmode、gtape 或 Server Administrator (ISA) 来添加或更改日志记录。然后使用 ON-Bar 或 gtape 来备份数据。在使用 ON-Bar 或 gtape 时，数据库服务器必须处于联机、管理或静默方式。

有关 ON-Bar 和 gtape 的信息，请参阅《GBase 8s 备份与复原指南》。

下表显示数据库服务器管理员能如何更改数据库记录方式。某些日志记录方式更改是立即发生的，而另外一些更改需要 0 级备份。

表 1. 日志记录方式转换

转换源：	转换为无日志记录	转换为未缓冲日志记录	转换为已缓冲日志记录	转换为符合 ANSI 标准
无日志记录	不适用	（受影响存储空间的）0 级备份	（受影响存储空间的）0 级备份	（受影响存储空间的）0 级备份
未缓冲日志记录	是	不适用	是	是
缓冲的日志记录	是	是	不适用	是
符合 ANSI 标准	非法	非法	非法	不适用

更改数据库日志记录方式有以下影响：

- 数据库服务器在更改日志记录状态时对数据库加上互斥锁定以防止其他用户访问该数据库，而当更改完成时释放该锁定。

- 如果在日志记录方式更改期间发生故障，那么在复原数据库服务器数据后，请检查 **sysmaster** 数据库的 **sysdatabases** 表内标志中的日志记录方式。有关更多信息，请参阅监视数据库的日志记录方式。然后重试更改日志记录方式。
- 如果在日志记录方式更改期间发生故障，那么在复原数据库服务器数据后，请检查 ISA 中的日志记录方式或 **sysmaster** 数据库中 **sysdatabases** 表中的标志。有关更多信息，请参阅监视数据库的日志记录方式。然后重试更改日志记录方式。
- 在选择了已缓冲或未缓冲日志记录之后，应用程序就可以使用 SQL 语句 SET LOG 从一种日志记录方式更改为另一种日志记录方式。此更改在会话期间会一直持续。有关 SET LOG 的信息，请参阅 *GBase 8s SQL 指南：语法*。
- 如果您向数据库添加日志记录，那么直至数据库的所有存储空间的下一次 0 级备份才完成该更改。

4.2.2 使用 gdblogmode 修改数据库日志记录方式

您可以使用 gdblogmode 实用程序更改一个或多个数据库的日志记录方式。如果您向数据库添加日志记录，那么必须在更改生效之前对包含该数据库的数据库空间创建 0 级备份。有关更多信息，请参阅《GBase 8s 管理员参考》中有关使用 gdblogmode 的主题。

使用 gdblogmode 更改缓冲方式

要在名为 **stores_demo** 的数据库上将缓冲方式从已缓冲日志记录更改为未缓冲日志记录，请运行以下命令：

```
gdblogmode unbuf stores_demo
```

要在名为 **stores_demo** 的数据库上将缓冲方式从未缓冲日志记录更改为已缓冲日志记录，请运行以下命令：

```
gdblogmode buf stores_demo
```

使用 gdblogmode 取消日志记录方式更改

要在发生下一次 0 级备份之前取消日志记录方式更改请求，请运行以下命令：

```
gdblogmode cancel stores_demo
```

不能取消那些立即执行的日志记录更改。

使用 gdblogmode 结束日志记录

要结束名为 dbfile 的文件中所列的两个数据库的日志记录，请运行以下命令：

```
gdblogmode nolog -f dbfile
```

使用 gdblogmode 使数据库符合 ANSI 标准

要使用 **gdblogmode** 使名为 stores_demo 的数据库成为符合 ANSI 标准的数据库，请运行以下命令：

```
gdblogmode ansi stores_demo
```

更改符合 ANSI 标准的数据库的日志记录方式

在将数据库创建或转换为 ANSI 方式之后，就不能轻易将其更改为任何其他日志记录方式。如果您意外地将数据库转换为 ANSI 方式，请遵循以下步骤来更改日志记录方式：

要更改日志记录方式，请执行以下操作：

1. 要卸载数据，可使用 dbexport 或任何其他迁移实用程序。

   dbexport 实用程序创建 schema 文件。

2. 要使用已缓冲日志记录重新创建数据库并装入数据，请使用 dbimport -l buffered 命令。

   要使用未缓冲日志记录重新创建数据库并装入数据，请使用 dbimport -l 命令。

4.2.3 使用 gtape 修改数据库日志记录方式

如果您将 gtape 用作备份工具，那么可使用 gtape 来更改数据库的日志记录方式。

使用 gtape 打开事务日志记录

在您修改数据库日志记录方式之前，请阅读更改数据库日志记录方式。

在创建 0 级备份的同时用 gtape 向数据库添加日志记录。

例如，要使用 **gtape** 向名为 stores_demo 的数据库添加已缓冲日志记录，请运行以下命令：

```
gtape -s -B stores_demo
```

要使用 **gtape** 向名为 stores_demo 的数据库添加未缓冲日志记录，请运行以下命令：

```
gtape -s -U stores_demo
```

除了打开事务日志记录以外，这些命令还创建整个系统存储空间备份。当 gtape 提示您提供备份级别时，请指定 0 级备份。

提示： 对于 gtape，您必须执行所有存储空间的 0 级备份。

使用 gtape 结束日志记录

要使用 **gtape** 为名为 stores_demo 的数据库结束日志记录，请运行以下命令：

```
gtape -N stores_demo
```

使用 gtape 更改缓冲方式

要使用 **gtape** 在名为 stores_demo 的数据库上将缓冲方式从已缓冲日志记录更改为未缓冲日志记录，但不创建存储空间备份，请运行以下命令：

```
gtape -U stores_demo
```

要使用 **gtape** 在名为 stores_demo 的数据库上将缓冲方式从未缓冲日志记录更改为已缓冲日志记录，但不创建存储空间备份，请运行以下命令：

```
gtape -B stores_demo
```

使用 gtape 使数据库符合 ANSI 标准

要使用 **gtape** 使已经使用事务日志记录（未缓冲或已缓冲）的名为 stores_demo 的数据库成为符合 ANSI 标准的数据库，请运行以下命令：

```
gtape -A stores_demo
```

要使用 **gtape** 使尚未使用事务日志记录的名为 stores_demo 的数据库成为符合 ANSI 标准的数据库，请运行以下命令：

```
gtape -s -A stores_demo
```

除了使数据库符合 ANSI 标准以外，该命令还同时创建存储空间备份。当提示您提供级别时，请指定 0 级备份。

提示：在将日志记录方式更改为符合 ANSI 标准之后，就不能再轻易对其进行更改。要更改符合 ANSI 标准的数据库的日志记录方式，请卸载数据，使用新日志记录方式重新创建数据库，并重新装入数据。有关详细信息，请参阅更改符合 ANSI 标准的数据库的日志记录方式。

4.2.4 使用 ISA 修改数据库日志记录方式

Server Administrator (ISA) 使用 gdblogmode 实用程序修改数据库日志记录方式。如果您打开记录，请执行 0 级备份。有关更多信息，请参阅 ISA 联机帮助和使用 gdblogmode 修改数据库日志记录方式。

4.2.5 使用 ON-Monitor 修改数据库日志记录方式 (UNIX™)

您可以使用 ON-Monitor 使日志缓冲方式在未缓冲和已缓冲之间切换。如果要将日志记录添加到数据库或使数据库符合 ANSI 标准，那么不能使用 ON-Monitor，而必须使用 gtape。

要更改数据库的日志缓冲方式，请选择日志缓冲方式 > 数据库选项。

4.2.6 修改表日志记录方式

在缺省情况下，数据库服务器创建使用日志记录的标准表。要创建非日志记录表，请使用带 WITH LOG 子句的 CREATE TABLE 语句。有关 CREATE TABLE 和 ALTER TABLE 语句的信息，请参阅《GBase 8s SQL 指南：语法》。有关更多信息，请参阅 GBase 8s 的表类型。

更改表以关闭日志记录

要将表从日志记录切换为非日志记录，请使用 TYTE 选项为 RAW 的 SQL 语句 ALTER TABLE。例如，以下语句将表 **tablog** 更改为 RAW 表：

```
ALTER TABLE tablog TYPE (RAW)
```

更改表以打开日志记录

要从非日志记录表切换为日志记录表，请使用 TYPE 选项为 STANDARD 的 SQL 语句 ALTER TABLE。例如，以下语句将表 **tabnolog** 更改为 STANDARD 表：

```
ALTER TABLE tabnolog TYPE (STANDARD)
```

重要： 当您将表更改为 STANDRAD 时，请打开该表的日志记录。在更改表后，如果必须能够复原该表，请执行 0 级备份。

禁用对临时表的日志记录

可以禁用对临时表的日志记录以提高性能，并阻止 GBase 8s 在数据复制环境（含有 HDR 辅助服务器、RS 辅助服务器和 SD 辅助服务器）中使用主服务器时传输临时表。

要禁用对临时表的日志记录，可将 TEMPTAB_NOLOG 配置参数设置为 1。

对于高可用性集群中的 HDR、RSS 和 SDS 辅助服务器，必须通过将 TEMPTAB_NOLOG 配置参数设置为 1 来始终禁用对临时表的逻辑日志记录。

可以使用 gadmin -wf 命令来更改 TEMPTAB_NOLOG 的值。

4.2.7 监视事务

本主题中包含有关监视事务方法的信息的参考。

命令	描述	引用
gstat -x	监视事务。	监视全局事务
gstat -g sql	监视 SQL 语句，并按会话标识和数据库列出语句。	
gstat -g stm	监视预编译 SQL 语句的内存使用情况。	

4.2.8 监视数据库的日志记录方式

使用 SMI 表监视日志记录方式

查询 **sysmaster** 数据库中的 **sysdatabases** 表以确定日志记录方式。 该表包含对应于数据库服务器管理的每个数据库的行。**标志**字段指示数据库的日志记录方式。**is_logging**、**is_buff_log** 和 **is_ansi** 字段指示日志记录是否为活动的，以及使用已缓冲日志记录还是符合 ANSI 标准的日志记录。有关该表中的列的描述，请参阅《GBase 8s 管理员参考》中有关 **sysmaster** 数据库一章中的 **sysdatabases** 一节。

使用 ON-Monitor 监视日志记录方式 (UNIX™)

要使用 ON-Monitor 查明数据库的日志记录方式，请选 择状态 > 数据库选项。

当数据库日志记录打开时，ON-Monitor 显示缓冲方式。ON-Monitor 仅能显示最多 100 个数据库。如果您的数据库服务器上有 100 多个数据库，请按使用 SMI 表监视日志记录方式中所述，使用 SMI 表来显示完整列表。

使用 ISA 监视日志记录方式

要使用 Server Administrator (ISA) 显示数据库的日志记录状态和缓冲方式，请选择 SQL > 模式。

4.3 逻辑日志

日志记录中的信息和以下主题说明了数据库服务器如何使用逻辑日志。有关如何执行逻辑日志任务的信息，请参阅管理逻辑日志文件和管理数据库日志记录方式。

4.3.1 什么是逻辑日志？

为了保留自上次存储空间备份以来的事务和数据库服务器更改的历史记录，数据库服务器生成日志记录。数据库服务器将日志记录存储在逻辑日志中，这是由三个或更多逻辑日志文件组成的循环文件。将该日志称为逻辑的是因为日志记录代表数据库服务器的逻辑操作（而不是物理操作）。存储空间备份加上逻辑日志备份的组合在任何时候均包含数据库服务器数据的完整副本。

作为数据库服务器管理员，您必须配置并管理逻辑日志。例如，如果您不定期备份日志文件，那么逻辑日志会填满，而数据库服务器暂挂处理。

这些职责包含以下任务：

- 为逻辑日志选择适当的位置
  请参阅逻辑日志文件的位置。
- 监视逻辑日志文件状态
  请参阅逻辑日志文件的标识。
- 为逻辑日志分配适当的磁盘空间量
  请参阅逻辑日志文件的大小。
- 在任何需要之时分配附加日志文件
  请参阅分配日志文件。
- 将逻辑日志文件备份到介质上
  请参阅备份逻辑日志文件和释放逻辑日志文件。
- 管理 BLOB 空间和智能大对象空间的日志记录
  请参阅记录 BLOB 空间和简单大对象和记录智能大对象空间和智能大对象。

4.3.2 逻辑日志文件的位置

在数据库服务器初始化磁盘空间时，它将逻辑日志文件和物理日志放在根数据库空间中。您对该操作没有控制权。要提高性能（尤其是减少对根数据库空间的写入次数以及将争用最小化），可将逻辑日志文件从根数据库空间移出至磁盘上未被活动表或物理日志共享的数据库空间中。请参阅将逻辑日志文件移至另一个数据库空间。

要进一步提高性能，可将逻辑日志文件分成两组，并将这两组存储在两个不同的磁盘（两个磁盘均不包含数据）上。例如，如果您有 6 个逻辑日志文件，那么您可能会将文件 1、3 和 5 放在磁盘 1 上，将文件 2、4 和 6 放在磁盘 2 上。这种安排可提高性能，因为从不需要同一磁盘驱动器同时处理写入当前逻辑日志文件和备份到磁带。

逻辑日志文件包含关键信息，因此必须制作镜像以获取最大数据保护。如果将逻辑日志文件移至其他数据库空间，请计划对该数据库空间启动镜像过程。

4.3.3 逻辑日志文件的标识

每个逻辑日志文件（不管是否备份到介质）都有唯一的标识号。以初始化数据库服务器磁盘空间后所填充的第一个逻辑日志文件为 1 而开始该序列。在当前逻辑日志文件已满时，数据库服务器切换到下一个逻辑日志文件并为新日志文件将唯一标识号增加 1。新添加的或标记为删除的日志文件的唯一标识号为 0。

为每个逻辑日志文件分配的实际磁盘空间具有称为*日志文件号*的标识号。例如，如果您配置 6 个逻辑日志文件，那么这些文件具有从 1 到 6 的日志号。日志号可以不按顺序。在备份并释放逻辑日志文件时，数据库服务器将磁盘空间重新用于逻辑日志文件。

下表说明了日志号和唯一标识号之间的关系。 日志 7 在日志 5 之后插入，并在第二次循环交替中第一次使用。

表 1. 逻辑日志文件的编号顺序

日志文件号	第一次循环交替唯一标识号	第二次循环交替唯一标识号	第三次循环交替唯一标识号
1	1	7	14
2	2	8	15
3	3	9	16
4	4	10	17
5	5	11	18
7	0	12	19
6	6	13	20

4.3.4 逻辑日志文件的状态标志

所有逻辑日志文件在第一位置均具有以下状态标志之一：已添加 **(A)**、已删除 **(D)**、可用 **(F)** 或已使用 **(U)**。下表显示了可能的日志状态标志组合。

表 1. 逻辑日志状态标志

状态标志	逻辑日志文件的状态
A------	日志文件已添加且可用，但尚未使用。
D------	如果您删除具有状态 U-B 的日志文件，那么它会标记为已删除。该日志文件被删除，其空间得以释放，可在您为所有存储空间进行 0 级备份时重新使用。
F------	日志文件是空闲的且可供使用。 逻辑日志文件在备份后得以释放，逻辑日志文件中的所有事务均关闭，存储在该文件中的最旧更新会清空到磁盘。
U	日志文件已使用但尚未备份。
U-B----	日志文件已备份但仍需用于恢复。（当不再需要该日志文件用于恢复时，将释放该文件。）
U-B---L	日志已备份但仍需用于恢复。包含上一个检查点记录。
U---C	数据库服务器当前正在填充日志文件。
U---C-L	当前日志文件包含上一个检查点记录。

使用 gstat -l 命令按号码列出日志文件，并监视已使用日志空间的状态标志和百分比。有关更多详细信息，请参阅 gstat -l 命令。

4.3.5 逻辑日志文件的大小

逻辑日志文件的最小大小为 200 KB。

逻辑日志文件的最大大小为 524288 页（等于 0x7ffff + 1），其中基页大小为 2 KB 或 4 KB，具体取决于操作系统。要确定您操作系统上数据库服务器的基页大小，请运行 gstat -d，然后检查根数据库空间的 pgsize 值。

确定要使用的日志文件的大小和数量。如果分配了多于所需的磁盘空间，那么会浪费空间。但如果未分配足够的磁盘空间，那么性能可能会受到不利影响。当许多用户在同时写入日志时，使用较大日志文件。

注： 较小的日志文件意味着如果包含该日志文件的磁盘脱机，那么您可在稍后进行恢复。如果设置了连续日志备份，那么日志文件在填满时会自动备份。较小日志导致逻辑恢复时间稍长。

逻辑日志文件的数量

当您估计逻辑日志文件的数目时，请考虑以下要点：

- 必须始终有至少 3 个逻辑日志文件，最多为 32,767 个日志文件。日志文件数目取决于日志文件的大小。
- 日志文件的数目影响逻辑日志备份的频率。
- 日志文件的数目影响 BLOB 空间 BLOB 页可回收的速率。请参阅备份日志文件以释放 BLOB 页。

性能注意事项

对于给定的系统活动级别，分配的逻辑日志磁盘空间越少，那么逻辑日志空间就越快填满，且用户活动就越有可能因备份和检查点而阻塞。调整逻辑日志大小以找出您系统的最佳值。

- 逻辑日志备份

  当逻辑日志文件填满时，必须对其进行备份。备份进程会阻碍涉及到与逻辑日志文件位于同一磁盘上的数据的事务处理。将物理日志、逻辑日志和用户数据放在不同的磁盘上。（请参阅《GBase 8s 备份与复原指南》。）

- 逻辑日志的大小

  较小逻辑日志比较大逻辑日志填充更快。 您可以按手动添加逻辑日志文件中的说明添加较大逻辑日志文件。

- 个别逻辑日志记录的大小

  逻辑日志记录的大小根据处理操作和数据库服务器环境而变化。通常，数据行越长，逻辑日志记录就越大。 逻辑日志包含已插入、更新或删除的行的映像。更新可使用多达插入和删除所用空间两倍的空间，因为更新可能既包含前映像又包含后映像。（插入仅存储后映像，而删除仅存储前映像。）

- 逻辑日志记录的数目

  向逻辑日志写入的逻辑日志记录越多，逻辑日志填充就越快。带事务日志记录的数据库填充逻辑日志比对不带事务日志记录的数据库进行的事务更快。

- 日志缓存的类型

  使用未缓冲事务日志记录的数据库填充逻辑日志比使用已缓冲事务日志记录的数据库更快。

- 对表进行的 Enterprise Replication

  因为 Enterprise Replication 会生成复制表的前映像和后映像，因此可能使逻辑日志填满。

- 回滚频率

  更多回滚会更快地填充逻辑日志。虽然回滚记录很小，但回滚本身也需要逻辑日志文件空间。

- 智能大对象的数目

  已启用用户数据日志记录且具有大量用户数据更新的智能大对象会以极快的速率填充逻辑日志。如果您不希望记录这些元数据，可使用临时智能大对象。

4.3.6 动态日志分配

动态日志分配可防止在长事务回滚期间日志文件填满并挂起系统。仅当下一个日志文件包含打开的事务时，该功能才处于活动状态。（如果事务在到达长事务高水位标志时未落实或回滚，那么为长事务。）

数据库服务器在下一个日志文件包含打开的事务时，在当前日志文件后自动（动态）分配日志文件。可以将动态记录分配用于以下操作：

- 当系统活动时添加日志文件
- 在当前日志文件后插入日志文件
- 立即访问新日志文件（即使根数据库空间未备份）

测试动态日志分配的最好方法是产生跨所有日志文件的事务，然后使用 gstat -l 检查是否有新添加的日志文件。有关更多信息，请参阅分配日志文件。

要点： 仍然必须备份日志文件以防止其填满。如果日志文件填满，那么系统挂起，直至您执行备份。

4.3.7 释放逻辑日志文件

每次数据库服务器提交或回滚事务时，会尝试释放事务开始所在的逻辑日志文件。在数据库服务器为重新使用而释放逻辑日志文件之前，必须满足以下标准：

- 备份日志文件。
- 逻辑日志文件中没有记录与打开的事务相关联。
- 逻辑日志文件不包含尚未刷新到磁盘的最旧更新。

下一个逻辑日志文件未释放时要执行的操作

如果数据库服务器尝试切换到下一个逻辑日志文件但发现顺次的下一个日志文件仍在使用中，那么数据库服务器立即暂挂所有处理。即使其他逻辑日志文件是可用的，数据库服务器也无法跳过使用中的文件而写入非顺次的可用文件。处理将停止，以保护逻辑日志文件中的数据。

逻辑日志文件可能由于以下任一原因而在使用中：

- 文件包含尚未清空到磁盘的最近检查点或最旧更新。

  发出 gadmin -c 命令以执行检查点并释放逻辑日志文件。有关更多信息，请参阅强制执行检查点。

- 文件包含打开的事务。

  打开的事务就是在设置用于回滚长事务的高水位标志中所说明的长事务。

- 文件未备份。

  如果逻辑日志文件未备份，那么当您使用 ON-Bar 或 gtape 来备份逻辑日志文件时，处理会恢复进行。

下一个逻辑日志文件中包含上一个检查点时要执行的操作

数据库服务器在下一个日志文件包含上一个检查点或最旧更新时不暂挂处理。如果尚未清空到磁盘的上一个检查点记录或最旧更新位于紧随上一个可用日志后的日志中，那么数据库服务器在切换到上一个可用日志时，将始终强制执行检查点。例如，如果 4 个逻辑日志文件具有以下列表中显示的状态，那么数据库服务器在切换到逻辑日志文件 3 时强制执行检查点。

日志文件号	逻辑日志文件状态
1	U-B----

2	U---C--
3	F
4	U-B---L

4.3.8 记录 BLOB 空间和简单大对象

简单大对象数据（TEXT 和 BYTE 数据类型）可能太多，无法包含在逻辑日志记录中。 如果始终记录简单大对象，它们可能会非常大，以致减缓了逻辑日志。

数据库服务器假设您将数据库设计为较小简单大对象存储在数据库空间中而较大简单大对象存储在 BLOB 空间中：

- 对于存储在数据库空间中的简单大对象，数据库服务器将简单大对象数据包含在日志记录中。
- 对于存储在 BLOB 空间中的简单大对象，数据库服务器不会将简单大对象数据包含在日志记录中。仅当您备份逻辑日志时，逻辑日志才记录 BLOB 空间数据。

为了那些对 BLOB 空间中简单大对象执行频繁更新的应用程序能获取更好的整体性能，可减少逻辑日志的大小。较小日志可改进对必须重新使用的简单大对象的访问。

切换日志文件以激活 BLOB 空间

您必须在以下情况中切换到下一个逻辑日志文件：

- 在创建 BLOB 空间后，如果您想要立即将简单大对象插入 BLOB 空间
- 向现有 BLOB 空间添加新块后，如果您要将简单大对象插入到将使用该新块的 BLOB 空间

数据库服务器需要在不同的逻辑日志文件中创建用于创建 BLOB 空间的语句、在 BLOB 空间中创建块的语句，以及将简单大对象插入该 BLOB 空间的语句。此需求与数据库日志记录状态无关。

有关切换到下一个日志文件的指示信息，请参阅切换到下一个逻辑日志文件。

备份日志文件以释放 BLOB 页

在删除存储在 BLOB 空间页中的数据时，不必为重新使用而释放那些页。仅当以下两个操作均已发生时，BLOB 空间页是可用的：

- 通过对列的更新或通过删除行而删除了 TEXT 或 BYTE 数据
- 备份了存储行（包含 TEXT 或 BYTE 数据）中插入的逻辑日志

插入或删除 TEXT 和 BYTE 数据之前备份 BLOB 空间

请确保备份所有包含有关 BLOB 空间中所存储简单大对象的事务的 BLOB 空间和逻辑日志。在日志备份期间，数据库服务器在逻辑日志中使用数据指针将更改过的 text 和 byte 数据从 BLOB 空间复制到逻辑日志中。

4.3.9 记录智能大对象空间和智能大对象

智能大对象空间中描述的智能大对象空间包含两个组成部分：元数据和用户数据。在缺省情况下，不记录智能大对象空间。

智能大对象空间的元数据组成部分描述存储在特定智能大对象空间中的智能大对象的关键特征。元数据包含指向智能大对象的指针。如果元数据要遭受损坏或变得不可访问，那么智能大对象空间也将损坏，且智能大对象空间中的智能大对象将不可恢复。

即使关闭数据库的日志记录，标准智能大对象空间中的元数据将始终得以记录。记录智能大对象空间元数据可确保元数据始终恢复为一致的事务状态。但是，不记录临时智能大对象空间中的元数据。

智能大对象空间日志记录

在记录智能大对象空间时，数据库服务器速度减慢，且逻辑日志迅速填满。如果您对智能大对象空间使用日志记录，那么必须确保逻辑日志大到足以容纳日志记录数据。有关更多信息，请参阅估计记录智能大对象时的日志大小。

为数据库打开日志记录后，数据库服务器直至您执行 0 级备份时才开始日志记录。但是，为智能大对象打开日志记录后，数据库服务器立即开始执行其日志记录更改。要减少日志条目数，请装入已关闭日志记录的智能大对象，然后再将日志记录重新打开以捕获对智能大对象的更新。

重要： 在为智能大对象打开日志记录时，必须立即执行 0 级备份，从而可恢复和复原智能大对象。

智能大对象的日志记录

如果用户在频繁更新数据，或如果恢复任何已更新数据的能力很重要，可为智能大对象使用日志记录。数据库服务器将操作（插入、更新、删除、读取或写入）记录写入逻辑日志缓冲区。CLOB 或 BLOB 数据的已修改部分包含在日志记录中。

要提高性能，可关闭智能大对象的日志记录。如果用户主要在分析数据而非在频繁更新数据，或如果数据对于恢复并不重要，也可关闭日志记录。

已更新的智能大对象的日志记录

在更新智能大对象时，数据库服务器不会记录整个对象。假设用户在为智能大对象启用日志记录的情况下以偏移量 Y 写入 X 字节的数据。数据库服务器将记录以下信息：

- 如果在大对象的末尾设置 Y，那么数据库服务器记录 X 字节（更新的字节范围）。
- 如果 Y 位于大对象的开始或中间，那么数据库服务器记录以下选项中的最小值：
    - 旧映像和新映像之间的差额
    - 前映像和后映像
    - 如果前映像和后映像相同，那么不记录任何值

关闭或打开智能大对象空间的日志记录

可以使用多种不同的方法来打开或关闭智能大对象空间的日志记录。

如果要在智能大对象空间中使用日志记录，请在创建智能大对象空间时指定 gspaces 命令的 **-Df "LOGGING=ON"** 选项。 如果在智能大对象空间中关闭了日志记录，您可在特定列中为智能大对象打开日志记录。包含智能大对象的一列可在另一列的日志记录关闭期间打开日志记录。

要验证智能大对象空间中的智能大对象是否已记录，请使用 gcheck -pS *sbspace_name* | grep “Create Flags” 命令。

如果在带缺省日志记录选项的智能大对象空间中创建智能大对象，且在输出中看到 LO_NOLOG 标志，那么该智能大对象空间中的智能大对象不会记录。如果在输出中看到 LO_LOG 标志，那么该智能大对象空间中的所有智能大对象均记录。

您可以按以下任一方法来修改智能大对象空间的日志记录状态。

要指定的函数或语句	日志记录操作	参考资料
gspaces -ch -Df "LOGGING=ON" gspaces-ch -Df ″LOGGING=OFF″	为现有的智能大对象空间打开或关闭日志记录	更改智能大对象的存储特征 《GBase 8s 管理员参考》中 gspaces -ch: 更改 sbspace 缺省规范的内容
使用 set sbspace logging on 或 set sbspace logging off 自变量的 SQL 管理 API task() 或 admin() 函数	为现有的智能大对象空间打开或关闭日志记录	《GBase 8s 管理员参考》中设置智能大对象空间日志记录自变量：更改智能大对象空间的日志记录（SQL 管理 API）的内容
CREATE TABLE 或 alter table 语句的 PUT 子句中的 LOG 选项	为装入该列的所有智能大对象打开日志记录	日志记录 PUT 子句
ifx_lo_create GBase 8s ESQL/C 函数	在最初装入智能大对象之时关闭它的日志记录	《GBase 8s ESQL/C 程序员手册》
ifx_lo_alter GBase 8s ESQL/C 函数	在装入完成后打开日志记录	《GBase 8s ESQL/C 程序员手册》

智能大对象日志记录

当用 LOG 选项创建智能大对象时，逻辑日志创建*智能 BLOB 日志记录*。智能 BLOB 日志记录跟踪对用户数据或元数据的更改。在更新智能大对象时，日志记录中仅包含智能大对象页的已修改部分。仅当为智能大对象启用了日志记录时，才会在逻辑日志中创建用户数据日志记录。

警告： 请注意为频繁更新的智能大对象启用日志记录。该日志记录开销可能会显著减慢数据库服务器的速度。

有关智能大对象的日志记录的信息，请参阅《GBase 8s 管理员参考》中有关解释逻辑日志记录的章节。

记录智能大对象数据时阻止长事务

您可在单个智能大对象的数据收集过程持续很长时间的情况下使用智能大对象。例如，考虑记录了许多个小时低质量音频信息的应用程序。虽然收集的数据量可能是最少的，但记录会话可能很长，导致出现长事务的条件。

提示： 要防止长事务发生，请定期落实对智能大对象的写入。

4.3.10 日志记录过程

这些主题详细描述了数据库空间、BLOB 空间和智能大对象空间的日志记录过程。这些信息对于执行常规数据库服务器管理任务不是必需的。

数据库空间日志记录

数据库服务器对涉及数据库空间中所存储数据的操作使用以下日志记录过程：

1. 将数据页从磁盘读到共享内存页缓冲区
2. 将未更改的页复制到物理日志缓冲区（如果需要）
3. 将新数据写入页缓冲区并创建事务的逻辑日志记录（如果需要）
4. 将物理日志缓冲区清空到磁盘上的物理日志
5. 将逻辑日志缓冲区清空到磁盘上的逻辑日志文件
6. 清空页缓冲区并将其写回到磁盘上

BLOB 空间日志记录

数据库服务器记录 BLOB 空间数据，但数据不经过共享内存或磁盘上的逻辑日志文件。数据库服务器将存储在 BLOB 空间中的数据直接从磁盘复制到磁带上。对 BLOB 空间开销页（自由图页和位图页）的修改记录是仅有的到达逻辑日志的 BLOB 空间数据。

4.4 管理逻辑日志文件

即使您的数据库都不使用事务日志记录，您也必须管理逻辑日志文件。请参阅逻辑日志以获取有关逻辑日志的背景信息。

在 UNIX™ 上，您必须以 **gbasedbt** 或 **root**用户身份登录才能进行本章中所述的任何更改。

在设置逻辑日志时要执行以下任务：

- 在您初始化或重新启动数据库服务器之前，请使用 LOGFILES 参数来指定要创建的逻辑日志文件的数量。
- 在数据库服务器联机后，请估计系统需要的逻辑日志文件的大小和数量。

  请参阅估计日志文件的大小和数量。
- 如果您不想要使用缺省值，请更改 LOGSIZE 和 LOGBUFF 配置参数。

  请参阅更改日志记录配置参数。
- 添加估计数目的逻辑日志文件。

请参阅分配日志文件。

您要例行执行以下任务：

- 备份逻辑日志文件
- 切换到下一个逻辑日志文件
- 释放逻辑日志文件
- 监视日志记录活动和日志备份状态

如有需要，您要例行执行以下任务：

- 添加逻辑日志文件
- 删除逻辑日志文件
- 更改逻辑日志文件的大小
- 移动逻辑日志文件
- 更改逻辑日志配置参数
- 为逻辑日志监视事件警报
- 为事务设置高水位标志

有关使用 SQL 管理 API 命令（而不是一些 gcheck、gadmin、glogadmin 和 gspaces 命令）的信息，请参阅使用 SQL 管理 API 执行远程管理和《GBase 8s 管理员参考》。

4.4.1 估计日志文件的大小和数量

使用 LOGSIZE 配置参数可设置逻辑日志文件的大小。

对于您的数据库服务器系统最优的日志空间量取决于以下因素：

- 您的应用程序需求和应用程序经历的更新活动量。更高的更新活动需要更大的日志空间。
- 恢复时间目标 (RTO) 标准，用于在重新启动服务器并使服务器进入联机或静默方式后，服务器从问题恢复所用时间量（以秒计）。

  在灾难性事件的情况中，要考虑您可以容许多少数据丢失。更多频繁的日志备份（该备份可以减少数据和事务丢失所带来的风险），需要更大的日志空间。
- 是使用 Enterprise Replication 还是使用数据复制配置（例如，HDR 辅助服务器、SD 辅助服务器或 RS 辅助服务器）。

  这些复制服务都可以影响日志文件的数量和大小。如果您的系统使用其中任何一个复制服务，请参阅高可用性集群配置或《GBase 8s Enterprise Replication 指南》中的指南。

确定日志大小的指南：

- 通常，管理较少的大日志文件比管理大量的小日志文件要容易得多。
- 日志空间过多不会影响性能。但是，日志文件和日志空间不足却会影响性能，因为数据库服务器将触发频繁的检查点。
- BLOB 空间中没有登录的智能大对象，但是这些对象包含在创建对象的日志备份中。这意味着对象将不会释放直到服务器备份创建了对象的日志。 因此，如果 BLOB

空间中的智能大对象频繁更新，您可能需要更加频繁的日志备份以获得 BLOB 空间中的更多可用空间。

- 对于生成少量日志数据的应用程序，请一开始使用 10 个日志文件，每个文件 10 MB。
- 对于生成大量日志数据的应用程序，请一开始使用 10 个日志文件，每个文件 100 MB。

有以下两种维护 RPO 策略的方式，这决定了在灾难性事件中所容许的数据丢失，例如，数据服务器的丢失：

- 一种维护 RPO 策略的方式是使用自动日志备份，即一旦日志文件填满就触发日志备份。这样就将数据丢失限制到在备份期间包含在日志文件中的事务以及在日志备份期间发生的任何其他事务。
- 另一种维护 RPO 策略的方式是使用调度程序。您可以创建一个任务，该任务自上次日志备份以来在设定的时间间隔内自动备份任何新的日志数据。这样就将数据丢失限制到在设定时间间隔内没有进行备份的事务。有关使用调度程序的信息，请参阅调度程序。

如果 RPO 策略是必需的，那么可以使用调度程序插入一个按适当频率执行的任务来维护该策略。这样就在日常周期内的特定时间自动备份日志文件。如果日志空间在日志备份或回收之前就已填满，那么可以备份这些日志并添加新的日志文件以允许事务处理继续进行，或者可以使用调度程序添加新的任务来检测此情况，并自动执行其中任一操作。

您可以随时添加日志文件，并且当事务一致性需要时数据库服务器会自动添加日志文件（例如，可能占用大量日志空间的长事务）。

增加逻辑日志空间量的最简单的方法是添加另一逻辑日志文件。请参阅手动添加逻辑日志文件。

以下表达式提供了示例总日志空间配置（以 KB 计）：

LOGSIZE = (((*connections* * *maxrows*) * *rowsize*) / 1024) / LOGFILES

表达式元素	解释
LOGSIZE	指定每个逻辑日志文件的大小（以 KB 计）。
connections	为您在 sqlhosts 文件或注册表以及在 NETTYPE 参数中指定的所有网络类型指定最大连接数。如果您通过在您的配置文件中设置多个 NETTYPE 配置参数来配置多个连接，请为每个 NETTYPE 添加用户字段，并替换前公式中 *connections* 的总数。
maxrows	指定在单个事务中要更新的最大行数。
rowsize	指定表行的平均大小（以字节计）。要计算 *rowsize*，请添加行中各列的长度（来自 **syscolumns** 系统目录表）。
1024	将 LOGSIZE 转换为指定的单位 (KB)。
LOGFILES	指定逻辑日志文件的数量。

估计记录智能大对象时的日志大小

如果计划记录智能大对象用户数据，那么必须确保日志大小比正写入的数据量大得多。 如果您将智能大对象存储在标准智能大对象空间中，那么始终记录元数据（即使不记录智能大对象）。如果您将智能大对象存储在临时智能大对象空间中，那么根本不存在日志记录。

估计逻辑日志文件的数量

LOGFILES 参数提供系统初始化或重新启动时逻辑日志文件的数量。如果所有逻辑日志文件的大小相同，您可以如下所示计算分配给逻辑日志文件的总空间：

```
逻辑日志空间总数 = LOGFILES * LOGSIZE
```

如果数据库服务器包含不同大小的日志文件，那么不能使用 (LOGFILES * LOGSIZE) 表达式来来计算逻辑日志的大小。必须改为将磁盘上的每个日志文件的大小相加。检查 **gstat -l** 输出中的 size 字段。 有关更多信息，请参阅 gstat -l 命令。

有关 LOGSIZE、LOGFILES 和 NETTYPE 的信息，请参阅《GBase 8s 管理员参考》 中有关配置参数的主题。

4.4.2 备份逻辑日志文件

逻辑日志包含已执行的事务的历史记录。逻辑日志文件复制到介质的过程称为*备份*逻辑日志文件。备份逻辑日志文件实现以下两个目标：

- 它将逻辑日志记录存储在介质上，以便在需要数据复原时可以前滚这些记录。
- 它使逻辑日志文件空间可用于新的逻辑日志记录。

  如果您忽略了备份日志文件，那么可能会耗尽日志空间。

您可以启动手动逻辑日志备份或设置连续逻辑日志备份。在复原存储空间后，您必须复原逻辑日志以使数据处于一致状态。有关日志备份的更多信息，请参阅《GBase 8s 备份与复原指南》。

备份 BLOB 空间

先备份逻辑日志还是先备份 BLOB 空间，这一点无关紧要。

要备份 BLOB 空间，请执行以下操作：

1. 在当前的逻辑日志包含有关 BLOB 空间中简单大对象的事务时关闭该日志。
2. 在更新简单大对象数据后尽快执行对逻辑日志和 BLOB 空间的备份。

警告：如果您不备份这些 BLOB 空间和逻辑日志，就可能无法复原 BLOB 空间数据。如果您等到 BLOB 空间不可用时才执行日志备份，数据库服务器将无法访问 BLOB 空间以将更改过的数据复制到逻辑日志中。

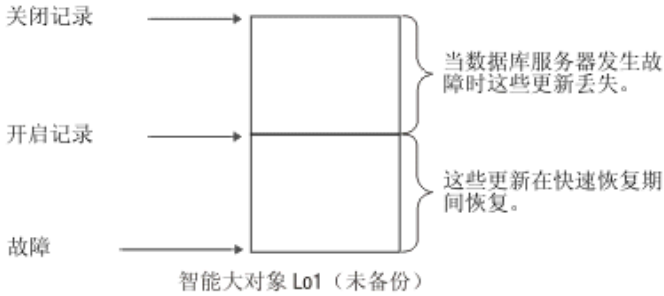
备份智能大对象空间

当为智能大对象打开日志记录时，必须执行该智能大对象空间的 0 级备份。

下图显示在未备份的智能大对象空间中打开日志记录时会发生的情况。尽管已记录的更改是可恢复的，但在故障期间会丢失未记录的对智能大对象 **LO1** 的更改。您将无法完全复原 **LO1**。

在快速恢复期间，数据库服务器前滚 **LO1** 的所有已落实事务。如果未记录 **LO1**，那么数据库服务器将无法回滚未落实的事务。而那时 **LO1** 内容将是不正确的。有关更多信息，请参阅快速恢复。

图: 打开智能大对象空间中的日志记录



4.4.3 切换到下一个逻辑日志文件

由于以下原因，您可能希望在当前日志文件变满之前切换到下一个逻辑日志文件：

- 备份当前日志
- 激活新的 BLOB 空间和 BLOB 空间块

数据库服务器可以处于联机方式以进行该更改。 运行以下命令以切换到下一个可用的日志文件：gadmin -l

更改将立即生效。（请确保您在命令行中输入小写的 L，而非数字 1。）

4.4.4 释放逻辑日志文件

如果日志文件是新添加的（状态 **A**），那么可立即使用该文件，也可立即删除。

您可能希望为以下原因而释放逻辑日志文件：

- 使数据库服务器不停止处理
- 释放已删除的 BLOB 页所用的空间

释放日志文件的过程根据日志文件的状态而变化。每个过程均在以下主题中进行了描述。要查明逻辑日志文件的状态，请参阅逻辑日志文件的状态标志和监视日志记录活动。

提示： 有关使用 ON-Bar 或 gtape 以备份存储空间和逻辑日志的信息，请参阅《GBase 8s 备份与复原指南》。

删除状态为 D 的日志文件

当删除已使用的日志文件时，该文件将标记为已删除（状态为 D）且无法再次使用，同时 glogadmin 将打印以下消息：

Log file *log_file_number* has been pre-dropped. It will be deleted from the log list and its space can be reused once you take level 0 archives of all BLOBspaces, Smart BLOBspaces and non-temporary DBspaces.

必需具有 0 级归档，以确保日志文件本身以及不同数据库空间中的所有关联信息都已归档。在 0 级归档的结束后会删除日志文件；但是，由于除去日志文件本身属于磁盘上根保留页结构中的更改，因此下一次执行的归档也必须是 0 级归档。必须先执行 0 级归档，然后才能执行 1 级或 2 级归档。

释放状态为 U 的日志文件

如果日志文件包含记录但尚未备份（状态为 U），请使用通常使用的备份工具来备份该文件。

如果备份日志文件无法将状态更改为可用 (F)，那么其状态会更改为 U-B 或 U-B-L。请参阅释放状态为 U-B 或 F 的日志文件或释放状态为 U-B-L 的日志文件。

释放状态为 U-B 或 F 的日志文件

如果日志文件已备份但仍在使用中（状态为 U-B），那么说明日志文件中的某些事务仍在进行，或日志文件包含快速恢复所需的最旧更新。由于过去已使用过状态为 F 的日志文件，因此该文件将与状态为 U-B 的日志文件一样遵循相同的规则。

要释放正在使用的已备份日志文件，请执行以下操作：

1. 如果您不希望等待至事务完成，可将数据库服务器切换到静默方式。

   请参阅立即从联机更改到静默方式。任何活动的事务将回滚。

2. 使用 gadmin -c 命令以强制产生一个检查点。执行此操作的原因是状态为 U-B 的日志文件可能包含最旧的更新。

已备份但不在使用中（状态为 U-B）的日志文件无需释放。在以下示例中，日志 34 无需释放，但日志 35 和 36 需要释放。日志 35 包含上一个检查点，而日志 36 已备份但仍在使用中。

```
34 U-B--    Log is used, backed up, and not in use
35 U-B-L    Log is used, backed up, contains last checkpoint
36 U-B--    Log is used, backed up, and not in use
37 U-C--    This is the current log file, not backed up
```

提示： 仅当逻辑日志没有通过活动的事务跨越并且不包含最旧的更新，您才能释放带有 U-B（而非 L）状态的逻辑日志。

释放状态为 U-C 或 U-C-L 的日志文件

请遵循以下步骤来释放当前日志文件。

要释放当前日志文件（状态为 C），请执行以下操作：

1. 运行以下命令将当前日志文件切换到下一个可用的日志文件：gadmin -l
2. 用 ON-Bar 或 gtape 备份原始日志文件。
3. 在备份了所有填满的日志文件后，会提示您切换到下一个可用的日志文件并备份新的当前日志文件。

   因为刚刚切换到此日志文件，所以无需执行备份。

在释放当前日志文件后，如果日志文件的状态为 U-B 或 U-B-L，请参阅释放状态为 U-B 或 F 的日志文件或释放状态为 U-B-L 的日志文件。

释放状态为 U-B-L 的日志文件

如果日志文件已备份，并且其中的所有事务已关闭，但该文件还未释放（状态为 U-B-L），那么此逻辑日志文件包含最近的检查点记录。可以释放状态为 U-B-L 的日志文件。

要释放状态为 U-B-L 的日志文件，数据库服务器必须创建新的检查点。 可以运行以下命令来强制执行检查点：gadmin -c

仅限 UNIX： 要用 ON-Monitor 强制执行检查点，请选择**强制执行检查点**选项。

4.4.5 监视日志记录活动

监视逻辑日志文件以确定总的可用空间（在所有文件中）、当前文件中的可用空间以及文件状态（例如，日志是否已备份）。有关监视逻辑日志缓冲区的信息，请参阅监视物理和逻辑日志记录活动。

监视逻辑日志记录以了解填充度

您可使用以下命令行实用程序来监视逻辑日志文件。

gstat -l 命令

gstat -l 命令显示有关物理日志和逻辑日志的信息。

包含有关每个逻辑日志文件的信息的输出部分包含以下信息：

- 逻辑日志文件描述符的地址
- 日志文件号
- 指示每个日志状态（空闲、已备份、当前等等）的状态标志
- 日志文件的唯一标识
- 文件的开始页
- 文件大小（以页计）、已用页数和已用页的百分比

如果您删除列表中间的几个日志或如果数据库服务器动态地添加日志文件，那么 numbers 字段的日志文件数量可以无序。

有关 gstat -l 输出的示例的更多信息，请参阅《GBase 8s 管理员参考》。

gcheck -pr 命令

数据库服务器将逻辑日志文件信息存储在专用于检查点信息的保留页中。因为数据库服务器仅在检查点期间更新该信息，所以该信息不如 gstat -l 选项所显示的信息新。有关使用这些选项来显示保留页信息的更多详细信息，请参阅《GBase 8s 管理员参考》。

您可以用 gcheck -pr 命令查看检查点保留页。以下示例显示一个逻辑日志文件的样本输出。

```
...
Log file number                 1
Unique identifier               7
Log contains last checkpoint Page 0, byte 272
Log file flags                  0x3    Log file in use
                                       Current log file

Physical location               0x1004ef
Log size                          750 (p)
Number pages used                   1
Date/Time file filled           01/29/2001 14:48:32

...
```

监视临时逻辑日志

因为在热复原期间永久日志不可用，所以此时数据库服务器使用*临时逻辑日志*前滚事务。当前滚完成时，数据库服务器将释放临时日志文件。如果您在热复原期间发出 gstat -l，那么输出将包含有关临时日志文件（其格式与常规日志文件相同）的第四部分。临时日志文件仅使用 B、C、F 和 U 状态标志。

SMI 表

查询 **syslogs** 表以获取有关逻辑日志文件的信息。该表包含对应于每个逻辑日志文件的行。各列如下所示。

number
逻辑日志文件的标识号

uniqid
日志文件的唯一标识

size
文件大小（以页计）

used
已用页数

is_used
指示日志文件是否正在使用的标志

is_current
指示日志文件是否为当前文件的标志

is_backed_up
指示日志文件是否已备份的标志

is_new
指示日志文件自上次存储空间备份以来是否已添加的标志

is_archived
指示日志文件是否已写入归档磁带的标志
is_temp
指示日志文件是否标记为临时日志文件的标志

使用 ON-Monitor 监视日志状态 (UNIX™)

可以使用 ON-Monitor 来查看有关逻辑日志的信息。

状态 > 日志选项显示的许多有关逻辑日志文件的信息与gstat -l 选项所显示信息相同。另外，还有一列包含各逻辑日志文件所在的数据库空间。

监视日志备份状态

为了监视日志的状态以及为了查看哪些日志已备份，请使用 gstat -l 命令。状态标志 B 表示日志已备份。

4.4.6 分配日志文件

当您初始化或重新启动数据库服务器时，该服务器将创建您在 LOGFILES 配置参数中所指定数量的逻辑日志文件。这些日志文件是您在 LOGSIZE 参数中指定的大小。

动态添加逻辑日志文件

DYNAMIC_LOGS 配置参数确定数据库服务器何时将动态添加逻辑日志文件。当您使用 DYNAMIC_LOGS 的缺省值 2 时，如果下一个活动日志文件包含最旧打开事务的开头，那么数据库服务器将动态添加新的日志文件并将警报设置为关闭。

数据库服务器将在以下时刻检查逻辑日志空间：

- 在切换到新日志文件后
- 在逻辑恢复的事务清除阶段开始时

如果 DYNAMIC_LOGS 参数设置为 1 并且下一个活动日志文件包含来自打开事务的记录，那么数据库服务器将提示您手动添加日志文件并将警报设置为关闭。在您添加日志文件后，数据库服务器将重新开始处理事务。

如果 DYNAMIC_LOGS 参数设置为 0 并且在长事务回滚过程中逻辑日志文件耗尽了空间，那么该数据库服务器可能挂起。（长事务将防止第一个逻辑日志文件变为空闲并防止其可用于重新使用。）要修复该问题并完成长事务，请将 DYNAMIC_LOGS 设置为 2 并且重新启动数据库服务器。

有关更多信息，请参阅监视动态添加的日志的事件和设置用于回滚长事务的高水位标志

动态添加的日志文件的大小和数量

动态日志的用途是添加足够的日志空间，以允许回滚事务。在动态添加日志文件时，数据库服务器使用以下因素来计算日志文件的大小：

- 平均日志大小
- 可用的连续空间量

如果逻辑日志空间过小，那么数据库服务器将按需添加尽可能多的日志文件，以回滚事务。日志文件的数目由以下条件限制：

- 支持的最大数目的日志文件
- 日志文件的磁盘空间量
- Root 块中可用的连续空间量

如果数据库服务器因为耗尽磁盘空间而停止添加新日志文件，它会写入错误消息并发出警报。向现有数据库空间添加数据库空间或块。然后数据库服务器会自动重新开始处理事务。

Root 块中的保留页存储有关每个日志文件的信息。在添加更多日志文件时，包含该信息的扩展数据块会扩展。Root 块需要两个 1.4 兆字节的扩展数据块，各用于跟踪 32,767 个日志文件（支持的最大数目）。

如果在还原期间要从非 Root 块分配块保留页扩展数据块，那么服务器会尝试将其放回 Root 块中。如果在 Root 块中没有足够空间可用，还原将失败。包含所需空间的消息会显示在联机日志中。所需空间必须在再次尝试还原之前从 Root 块中释放。

动态添加的日志文件的位置

数据库服务器按以下搜索顺序在数据库空间中分配日志文件。一个数据库空间如果包含逻辑日志文件或物理日志，就可成为关键的数据库空间。

途径	分配日志文件的位置
1	包含最新日志文件的数据库空间 （如果该数据库空间已满，数据库服务器会搜索其他数据库空间。）
2	包含日志文件的镜像数据库空间（但根数据库空间除外）
3	所有已经包含日志文件的数据库空间（根数据库空间除外）
4	包含物理日志的数据库空间
5	根数据库空间
6	任何镜像数据库空间
7	任何数据库空间

如果不想使用此搜索顺序来分配新的日志文件，那么必须将 DYNAMIC_LOGS 参数设置为 1 并在要用于新日志的位置运行 glogadmin -a -i。有关详细信息，请参阅监视动态添加的日志的事件。

手动添加逻辑日志文件

可以使用 glogadmin 命令或 ON-Monitor 来添加逻辑日志文件。

您可以为以下原因而手动添加逻辑日志文件：

- 增加分配给逻辑日志的磁盘空间
- 更改逻辑日志文件的大小
- 允许打开的事务回滚
- 作为将逻辑日志文件移至不同数据库空间的操作的一部分

限制： 您不能执行以下操作：

- 向 BLOB 空间或智能大对象空间添加日志文件。
- 向具有非缺省页大小的数据库空间添加逻辑或物理日志。

一次添加一个，最多向任一数据库空间添加 32,767 个逻辑日志文件。一旦您向数据库空间添加了日志文件，该空间就成为关键数据库空间。您可以在存储空间备份期间添加逻辑日志文件。

您可以在以下任一位置添加逻辑日志文件：

- 文件列表的末尾（使用 glogadmin -a 命令）
- 当前逻辑日志文件之后（使用 glogadmin -a -i 命令）

要使用 glogadmin 添加逻辑日志文件，请执行以下操作：

1. 以用户 gbasedbt 或 root 的身份（在 UNIX™ 上）。
2. 请确保数据库服务器处于联机、管理、静默或快速恢复方式的清除阶段。

   数据库服务器在清除阶段将以下消息写入日志：

```
Logical recovery has reached the transaction cleanup phase.
```

3. 决定要将日志文件添加到日志文件列表的末尾还是添加到当前日志文件之后。

   无论 DYNAMIC_LOGS 参数值为多少，您都可以在当前日志文件后插入日志文件。添加新大小的日志文件不会更改 LOGSIZE 的值。

   - 以下命令使用 LOGSIZE 配置参数指定的日志文件大小将逻辑日志文件添加到 logspace 数据库空间中逻辑日志文件列表的末尾：

     ```
     glogadmin -a -d logspace
     ```

   - 以下命令将 1000 KB 的逻辑日志文件插入 logspace 数据库空间中的当前日志文件之后：

     ```
     glogadmin -a -d logspace -s 1000 -i
     ```

   - 要添加具有新大小（此例中为 250 KB）的逻辑日志文件，请运行以下命令：

     ```
     glogadmin -a -d logspace -s 250
     ```

4. 使用 gstat -l 检查日志文件状态。

   新日志文件的状态为 **A**，且立即可用。

5. 下次必须备份数据时，请对根数据库空间和包含新日志文件的数据库空间执行 0 级备份。

   虽然在添加日志文件后不再需要立即备份，但因为数据结构已更改，所以下一次备份必须为 0 级备份。有关备份数据的信息，请参阅《GBase 8s 备份与复原指南》。

有关使用 glogadmin 来添加逻辑日志文件的更多信息，请参阅 《GBase 8s 管理员参考》。

使用 ON-Monitor 添加逻辑日志文件 (UNIX)

1. 请遵循手动添加逻辑日志文件中有关添加日志文件的指示信息，只是使用 ON-Monitor 来代替 glogadmin。
2. 请选择**参数** > **添加日志**来添加逻辑日志文件。
3. 在标为**数据库空间名称**的字段中，输入新逻辑日志文件将位于的数据库空间的名称。

   逻辑日志大小字段中将自动包含日志文件的大小。新的日志文件始终是由 LOGSIZE 指定的值。

4.4.7 删除逻辑日志文件

可以使用 glogadmin 命令或 ON-Monitor 来删除逻辑日志文件。

要删除逻辑日志文件并增加数据库空间中可用的磁盘空间量，可以使用 glogadmin。数据库服务器始终需要最少三个逻辑日志文件。如果逻辑日志仅由三个日志文件组成，就无法删除该日志。

删除日志文件的规则已更改：

- 如果您删除从未被写入的日志文件（状态 A），数据库服务器将立即删除它并释放空间。
- 如果您删除已使用的日志文件（状态 U-B），数据库服务器将其标记为已删除 (D)。在您对包含日志文件的数据库空间和根数据库空间进行了 0 级备份之后，数据库服务器删除日志文件并释放空间。
- 您无法删除当前正在使用或包含上一个检查点记录（状态 C 或 L）的日志文件。

要使用 glogadmin 删除逻辑日志文件，请执行以下操作：

1. 请确保数据库服务器处于联机、管理或静默方式中。
2. 运行以下命令来删除日志文件号为 21 的逻辑日志文件：glogadmin -d -l 21

   一次删除一个日志文件。您必须知道要删除的每个逻辑日志的日志文件号。
3. 如果日志文件具有新添加 (A) 的状态，那么将其立即删除。

   如果日志文件具有已使用 (U) 的状态，那么将其标记为删除 (D)。
4. 要删除已使用的日志文件，请对所有数据库空间进行 0 级备份。

该备份防止数据库服务器在复原期间使用已删除的日志文件，并确保保留页包含有关当前日志文件数量的信息。

有关使用 glogadmin 删除逻辑日志文件的信息，请参阅《 GBase 8s 管理员参考》。

有关使用 glogdump 显示逻辑日志文件和唯一标识号的信息，请参阅显示逻辑日志记录。

要使用 ON-Monitor 删除逻辑日志文件 (UNIX™)，请执行以下操作：

1. 请确保数据库服务器处于联机、管理或静默方式中。
2. 要删除逻辑日志文件，请选择参数 > 删除日志。
3. 如果日志文件的状态为“新添加”(A)，会立即删除该文件。

   如果日志文件具有已使用 (U) 的状态，那么将其标记为删除 (D)。
4. 要删除已使用的日志文件，请对所有数据库空间进行 0 级备份。

   提示： 如果从未备份根数据库空间，您可以立即删除已使用的日志文件。

4.4.8 更改逻辑日志文件的大小

如果要更改日志文件的大小，那么添加适当大小的新日志文件并随后删除旧文件会更为容易。可以用以下方法来更改逻辑日志文件的大小：

- 使用 glogadmin 连同 -s 选项可添加不同大小的新日志文件。

  请参阅手动添加逻辑日志文件。
- 如果希望数据库服务器创建较大的日志文件，请增加 onconfig 文件中的 LOGSIZE 值。

  请参阅更改日志记录配置参数。

4.4.9 将逻辑日志文件移至另一个数据库空间

按逻辑日志文件的位置中的说明，您可能由于性能原因或为使数据库空间中有更多空间而希望移动逻辑日志文件。要查明逻辑日志文件的位置，请参阅监视日志记录活动。 虽然移动逻辑日志文件并不难，但它可能很耗时。

移动逻辑日志文件将由两个较简单的操作组合完成：

- 从逻辑日志文件当前所在数据库空间删除这些文件
- 将逻辑日志文件添加到其新数据库空间

以下过程提供如何将 6 个逻辑日志文件从根数据库空间移至另一数据库空间 dbspace_1 的示例。

限制： 您不能在非缺省页大小的数据库空间中移动逻辑和物理日志文件。

要将逻辑日志文件从根数据库空间移出（示例），请执行以下操作：

1. 请确保数据库服务器处于联机、管理、静默或快速恢复方式中。
2. 向 dbspace_1 添加 6 个新逻辑日志文件。

   请参阅手动添加逻辑日志文件。
3. 对所有存储空间进行 0 级备份，以释放除当前日志文件以外的所有日志文件。

   （如果使用 gbackuprestore -l -b -c，那么可备份包括当前日志文件在内的所有日志文件。）请参阅释放逻辑日志文件。
4. 使用 gadmin -l 切换到新的当前日志文件。

   请参阅切换到下一个逻辑日志文件。
5. 删除根数据库空间中的全部 6 个逻辑日志文件。

   您不能删除当前逻辑日志文件。

   请参阅删除逻辑日志文件。
6. 创建根数据库空间和 dbspace_1 的 0 级备份。

   有关更多信息，请参阅《GBase 8s 备份与复原指南》。

4.4.10 更改日志记录配置参数

可以使用文本编辑器来更改数据库服务器用于日志记录的配置参数

下表显示了逻辑日志的配置参数。有关更多信息，请参阅 GBase 8s 管理员参考 中有关配置参数的主题。

配置参数	最小值	缺省值	最大值
DYNAMIC_LOGS	0 或 1	2	2
LOGBUFF	2 * 页大小	64 KB	LOGSIZE 值
LOGFILES	3 个文件	5 个文件	32,767 个文件
LOGSIZE	10000 KB（在 UNIX™ 上）	10000 KB	请参阅《GBase 8s 管理员参考》
LTXEHWM	LTXHWM 值	90%	100%
LTXHWM	1%	80%	100%

要点： 对于 LOGFILES 的更改直到您重新初始化或重新启动了磁盘空间才会生效。

要更改 onconfig 文件中的逻辑日志配置参数，请执行以下操作：

1. 使得数据库服务器脱机或进入静默或管理方式中。
2. 使用文本编辑器来更新配置参数。

   DYNAMIC_LOGS、LTXHWM 和 LTXEHWM 参数不在 onconfig.std 文件中。要更改这些参数的值，请将它们添加到 onconfig 文件。
3. 关闭并重新启动数据库服务器。
4. 仅当您要更改 LOGFILES，且希望所有日志文件位于连续空间中时，请执行该步骤。（通常您会使用 glogadmin 实用程序来添加和删除 LOGFILES。）
   a. 卸载所有数据库服务器数据。

      这么做是因为您不能依赖于存储空间备份来卸载和复原数据，因为复原会将参数返回到以前的值。
   b. 重新初始化数据库服务器磁盘空间。

      请参阅初始化磁盘空间。
   c. 重新创建所有数据库和表。
   d. 重新装入所有数据库服务器数据。
5. 备份根数据库空间以启用更改过的逻辑日志。

4.4.11 显示逻辑日志记录

使用 glogdump 实用程序显示并解释逻辑日志记录有关使用 glogdump 的信息，请参阅《GBase 8s 管理员参考》。

4.4.12 监视动态添加的日志的事件

监视以下由动态添加的日志文件触发的事件警报（见下表）。 当每个警报被触发，就向消息日志写入一条消息。有关更多信息，请参阅《GBase 8s 管理员参考》中有关事件警报和配置参数的章节。

您可以包含 glogadmin 命令以在您的事件类标识 27（日志文件必需的）的警报脚本中添加日志文件。您的脚本还可运行 gstat -d 命令来检查有否足够空间，并对具有足够空间的位置执行 glogadmin a -i。必须使用 -i 选项在当前日志文件之后添加新的日志。

表 1. 动态添加的日志文件的事件警报

类标识	严重性	类消息	消息
26	3	动态添加的日志文件 log_number	当数据库服务器动态添加日志文件时显示该消息。 动态地将日志文件 log_number 添加到数据库空间 dbspace_number。
27	4	需要日志文件	当 DYNAMIC_LOGS 设置为 1 且数据库服务器等待您添加日志文件时，显示该消息。 警报：最旧的逻辑日志 log_number 包含来自打开的事务 transaction_address 的记录。逻辑日志记录将保持阻塞状态，直到添加了日志文件为止。如下所示，使用 glogadmin -a 命令和 -i（插入）选项添加日志文件：glogadmin -a -d dbspace -ssize-i 然后尽快完成该事务。
28	4	没有可用于日志文件的空间	警报：因为最旧的逻辑日志 log_number 包含来自 transaction_address 打开事务的记录，那么服务器将尝试动态添加日志文件。但没有可用的空间。添加数据库空间或块，然后尽快完成该事务。

下表显示数据库服务器对于 DYNAMIC_LOGS 配置参数的每个设置执行的操作。

表 2. DYNAMIC_LOGS 设置

DYNAMIC_LOGS	含义	事件警报	等待添加日志	动态日志添加
2（缺省值）	允许自动分配新日志文件，从而防止打开的事务挂起系统。	是（26、28）	否	是
1	允许手动添加新日志文件。	是 (27)	是	否
0	不分配日志文件，但发出以下有	否	否	否

DYNAMIC_LOGS	含义	事件警报	等待添加日志	动态日志添加
	关打开的事务的消息： **警告：** 最旧的逻辑日志文件 log_number 包含来自打开事务 transaction_address 的记录，但是动态日志功能已关闭。			

4.4.13 设置用于回滚长事务的高水位标志

数据库服务器使用 LTXHWM 和 LTXEHWM 配置参数来设置长事务的高水位标志。如果 DYNAMIC_LOGS 设置为 1 或 2，那么缺省 LTXHWM 值为 80%，而 LTXEHWM 为 90%。如果 DYNAMIC_LOGS 设置为 0，那么缺省的 LTXHWM 值是 50% 而缺省的 LTXHEWM 值是 60%。

如果您减少高水位标志值，就增加了长事务的可能性。要进行补偿，须分配附加的日志空间。有关 LTXHWM 和 LTXEHWM 的信息，请参阅《GBase 8s 管理员参考》中有关配置参数的章节。

长事务高水位标志 (LTXHWM)

长事务高水位标志是在回滚事务前允许事务跨及的总日志空间的百分比。如果数据库服务器在最旧使用的日志文件中发现打开的事务，它会动态添加日志文件。因为日志空间一直在增加，所以高水位标志会向外扩展。当日志空间到达高水位标志，数据库服务器会回滚事务。事务回滚及其他过程也会生成逻辑日志记录。数据库服务器继续添加日志文件直至回滚完成，以便防止逻辑日志耗尽空间。如果存在一个以上的长事务，那么可回滚一个以上的事务。

例如，数据库服务器具有 10 个逻辑日志且 LTXHWM 将设置为 98。事务从日志文件 1 开始，而更新活动填满日志 1 到 9。数据库服务器在日志文件 10 之后动态添加日志文件 11。只要事务不完成，该过程就一直持续至数据库服务器添加了 40 个日志文件。当数据库服务器添加第 50 个日志时，事务已赶上高水位标志，而数据库服务器会将其回滚。

互斥存取长事务高水位标志 (LTXEHWM)

在当前正回滚的长事务获得对逻辑日志的互斥存取权时，发生了互斥存取长事务高水位标志。数据库服务器显著减少了日志记录的生成。仅允许当前正在回滚事务的线程和当前正在写 COMMIT 记录的线程访问逻辑日志。限制访问逻辑日志为用户线程正在写入的回滚记录保留了尽可能多的空间（该线程正在回滚事务）。

重要： 如果您同时将 LTXHWM 和 LTXEHWM 设置为 100，那么长事务将永不停止。 因此，必须将 LTXHWM 设置为低于 100，才能进行正常的数据库服务器操作。为运行未知长度的

已调度事务，将 LTXHWM 设置为 100。如果当长事务在回滚并且当您有大量磁盘空间时您从不希望阻塞其他用户，那么将 LTXEHWM 设置为 100。

调整日志文件大小以防止长事务

当许多用户在同时写入日志时，使用较大日志文件。如果您使用小日志，而长事务又有可能发生，那么降低高水位标志。请将 LTXHWM 值设置为 50 并将 LTXEHWM 的值设置为 60。

如果日志文件太小，数据库服务器可能会在回滚长事务时耗尽日志空间。在这种情况下，数据库服务器无法足够快速地阻拦以便及时在上一个日志文件填满之前添加新日志文件。如果上一个日志文件填满，系统将挂起并显示错误消息。要解决该问题，可关闭并重新启动数据库服务器。有关详细信息，请参阅从长事务挂起恢复。

从长事务挂起恢复

如果您的系统具有足够的磁盘空间并且您想要执行未知长度的事务，请考虑将 LTXHWM 设置为 100 以强制数据库服务器继续添加日志文件直到您完成事务。

事务可能因为数据库服务器已耗尽磁盘空间而挂起。数据库服务器将停止添加新的日志文件、写入错误消息并且发出警报。

要继续事务，请执行以下操作：

1. 向数据库空间添加数据库空间或块。
2. 重新开始处理该事务。

如果您无法将更多的磁盘空间添加到数据库服务器，请终止该事务。

要添加事务

- 发出 gadmin -z 命令。
- 关闭并重新启动数据库服务器。

当数据库服务器以快速恢复方式启动时，事务将回滚。然后执行以下步骤：

从长事务挂起恢复

1. 添加更多磁盘空间或另一磁盘，直至事务成功回滚。
2. 在长事务开始之前执行时间点复原或尽早执行时间点复原以便使数据库服务器可以回滚事务。
3. 将额外的日志文件、数据库空间或块从数据库服务器实例删除。
4. 执行完全的 0 级备份以释放逻辑日志空间。

4.5 物理日志记录、检查点和快速恢复

这些主题涵盖数据库服务器用来实现数据一致性的三个过程：

- 物理日志记录
- 检查点
- 快速恢复

物理日志磁盘页的集合，数据库服务器在此存储称为前映像的未修改页的副本。物理日志记录是存储数据库服务器将要更改的页的前映像的过程。 检查点是当数据库服务器将磁盘上的页与共享内存缓冲区中的页进行同步时的时间点。快速恢复是一种自动过程，该过程在数据库服务器在无控的条件下脱机后将数据库服务器复原到一致状态。

这些过程确保将多个逻辑相关的写入记录为一个单元，并将共享内存中的数据与磁盘上的数据定期调整为一致。

有关管理和监视物理日志和检查点的任务，请参阅管理物理日志。

4.5.1 临界区

临界区是必须作为单个单元执行的代码段（或机器指令）。临界区通过允许线程在其交换出去之前运行一系列指令，可确保线程的完整性。

4.5.2 物理日志记录

物理日志记录是在更改页实际记录在磁盘上之前存储数据库服务器将要更改的页的过程。在数据库服务器修改共享内存缓冲池中的某些页之前，它将页的前映像存储在共享内存中的物理日志缓冲区。

数据库服务器为这些页而将前映像页保留在共享内存的物理日志中，直至一个或多个页清除程序将页清空到磁盘。未修改的页在数据库服务器发生故障或备份过程需要它们的情况下可用于提供数据库服务器数据的准确快照。快速恢复和数据库服务器备份会使用这些快照。

数据库服务器重新启动每个检查点上的物理日志，但特殊情况下的除外。有关检查点的更多信息，请参阅检查点。

快速恢复物理记录页的使用

在故障后，数据库服务器使用页的前映像将磁盘上这些页复原至它们在上一个检查点中的状态。 然后数据库服务器使用逻辑日志记录使所有数据返回至最近完成的事务之时物理和逻辑上的一致状态。快速恢复更详细地说明了该过程。

物理记录的页的备份使用

当您执行备份时，数据库服务器执行检查点并检查物理日志，以确定备份上所属的页的正确版本。在 0 级备份中，数据库服务器备份所有磁盘页。 有关更多详细信息，请参阅《GBase 8s 备份与复原指南》。

物理记录的数据库服务器活动

如果对检查点之间的页作了多次调整，那么通常仅在物理日志中记录第一个前映像。

物理日志是一种循环日志，其中仅对每个检查点使用一次物理日志中的页。如果设置了 RTO_SERVER_RESTART 配置参数，那么将出现其他物理日志以提高快速恢复性能。

物理恢复消息

当快速恢复开始时，数据库服务器记录以下带有块和偏移量名称的消息：

```
Physical recovery started at page chunk:offset.
```

当快速恢复完成时，数据库服务器记录以下带有已检查和已复原页的数目的消息：

```
Physical recovery complete: number pages examined, number pages restored.
```

物理日志记录和简单大对象

物理日志中的数据库服务器页可以是任何数据库服务器页，包括表空间 (tblspaces) 中的简单大对象。甚至开销页（如块可用列表页、BLOB 空间可用图页和 BLOB 空间位图页）也会在页上数据修改并清空到磁盘之前复制到物理日志。

BLOB 空间 BLOB 页不在物理日志中记录。有关 BLOB 空间日志记录的更多信息，请参阅记录 BLOB 空间和简单大对象。

物理日志记录和智能大对象

智能大对象的用户数据部分不进行物理记录。但是，元数据要物理记录。有关智能大对象的信息，请参阅智能大对象空间。

4.5.3 物理日志的大小和位置

这些主题描述了如何配置物理日志的大小和位置。

指定物理日志的位置

在数据库服务器初始化磁盘空间时，它将逻辑日志文件和物理日志放在根数据库空间中。您对该放置没有初始控制权。要提高性能（尤其是减少对根数据库空间的写入次数以及将磁盘争用最小化），可将物理日志从根数据库空间移出至另一数据库空间（最好是不包含活动表或逻辑日志文件的磁盘）。

建议： 找到容错存储设备上的关键数据库空间。如果物理日志所在的存储器不是容错存储器，请对包含物理日志的数据库空间使用 GBase 8s 镜像。这将在存储设备发生故障时保护数据库。

用于估计物理日志的大小的策略

物理日志的大小取决于两个因素：事务生成物理日志活动的速率和是否设置了 RTO_SERVER_RESTART 配置参数

事务生成物理日志活动的速率可影响检查点性能。在检查点处理期间，如果因事务持续生成物理日志数据而导致物理日志开始变得太满，那么数据库服务器将阻塞事务以使检查点完成，并避免物理日志溢出。

要避免事务阻塞，数据库服务器必须具有足够的物理日志空间来包含检查点处理期间出现的所有事务活动。在物理日志填充度达到 75% 时触发检查点。在物理日志填充度达到 75% 时，必须在完成检查点处理之后才能使用剩余的 25% 的物理日志。一旦系统检测到有物理日志溢出的可能性，就会发生事务阻塞，因为每个活动事务都可能生成一个物理日志活动。

例如，假设您具有 1 千兆字节的物理日志和 1000 个活动事务。如果每个事务同时处于临界区，那么这 1000 个活动事务具有生成大约 80 兆字节物理日志活动的可能性。当填入 750 兆字节的物理日志时，数据库服务器将触发检查点。如果在使用 920 兆字节物理日志时检查点仍未完成，那么将出现事务阻塞直到检查点完成。如果发生事务阻塞，那么服务器将自动触发更频繁的检查点以避免事务阻塞。您可以禁用自动检查点的生成。

如果存在大量脏分区，那么即使物理日志填充度不到 75%，服务器也可能触发检查点，因为将修改的分区数据清空到磁盘需要物理日志空间。 当服务器检查物理日志填充度是否达到 75% 时，服务器还会检查以下情况是否为真：

```
（使用的物理日志页数 + 脏分区数量）>= （（物理日志大小 * 9）/10）
```

有关检查点处理和自动检查点的更多信息，请参阅检查点。

估计物理日志大小时要考虑的第二个因素取决于是否使用 RTO_SERVER_RESTART 配置参数指定了快速恢复的目标时间量。如果无需考虑快速恢复时间，那么不需要启用 RTO_SERVER_RESTART 配置参数。如果为 RTO_SERVER_RESTART 配置参数指定值，那么事务活动将生成附加物理日志活动。

通常，此附加物理日志活动对事务性能影响较小或没有影响。额外的日志记录用于在快速恢复期间辅助缓冲池，以便以最佳方式执行日志重放。如果物理日志比所有缓冲池的总大小要大很多，那么快速恢复期间将对页清空并出现缺页故障。页清空和缺页故障大幅减小了快速恢复性能，而且数据库服务器不能维护 RTO_SERVER_RESTART 策略。

对于缓冲池空间小于 4 千兆字节的系统，物理日志的大小可定在所有缓冲池总大小的 110%。对于较大的缓冲池，以 4 千兆字节的物理日志空间开始，然后监视检查点活动。如果检查点发生过于频繁，似乎会影响性能，请增加物理日志大小。

为数据库服务器配置了较小的物理日志并且该服务器具有大量用户时，可能会出现一种称为物理日志溢出的罕见情况。遵循上述有关大小的准则可帮助避免物理日志溢出。每当消息日志检测到未达到最佳标准的配置时，数据库服务器将对其生成性能警告。

如果检测到未达最佳标准的配置，那么您可以使用 gstat -g ckp 命令来显示配置建议。

事务日志记录关闭时物理日志溢出

如以下示例所示，如果您在事务日志记录关闭的数据库中使用简单大对象或智能大对象，物理日志可能会溢出。

在数据库服务器处理简单大对象时，数据库服务器存储在磁盘上的简单大对象的每个部分均可分别记录，允许线程退出各部分之间的代码临界区。但如果日志记录关闭，那么数据库服务器必须对一个临界区中的简单大对象执行所有操作。如果该简单大对象很大，而物理日志很小，那么该情况可能导致物理日志填满。如果发生这种情况，数据库服务器会将以下消息发送至消息日志：

```
Physical log file overflow
```

然后数据库服务器启动关机操作。有关建议的更正操作，请参阅消息日志。

4.5.4 检查点

数据库服务器会定期将缓冲池内的事务和数据清空到磁盘。直到将事务和数据清空到磁盘之前，数据和事务都处于流出的状态。除了在事务完成后立即强制将每个事务清空到磁盘，数据库服务器还将事务写入到逻辑日志中。数据库服务器在事务发生时记录事务。 如果出现系统故障，那么服务器执行以下操作：

- 重放日志以重做和复原事务。
- 将数据库返回至与发生故障时数据库系统的状态一致的状态。

为了便于数据库系统的复原或逻辑恢复，数据库服务器生成一致性点，称为检查点。检查点是建立数据库系统的已知和一致状态时日志中的时间点。通常，检查点涉及到记录特定数量的信息，因此，如果发生故障，数据库服务器可在已建立的点上重新启动。

检查点的目的在于定期将逻辑日志中的重新启动点向前移动。如果检查点不存在而且发生故障，那么数据库服务器需要处理自系统重新启动以来逻辑日志中记录的所有事务。

检查点可在以下某个情境中出现：

- 当指定事件发生时。例如，每当将数据库空间添加到服务器或执行数据库备份时，检查点将出现。

  通常，这些类型的事件会触发阻塞事务处理的检查点。因此，这些检查点称为阻塞检查点。
- 当资源限制发生时。例如，逻辑日志空间的每个范围需要检查点来保证日志具有开始快速恢复的检查点。数据库服务器将在物理日志达到总大小的 75% 时触发检查点，以避免物理日志溢出。

  资源限制触发的检查点通常不会阻塞事务。因此，这些检查点称为非阻塞检查点。

  但是，如果在检查点处理期间数据库服务器将要耗尽资源，那么在检查点处理的中段将出现事务阻塞，以保证耗尽资源之前检查点能够完成。如果事务被阻塞，那么服务器将更频繁的尝试触发检查点，以避免检查点处理期间的事务阻塞。有关更多信息，请参阅用于估计物理日志的大小的策略。

自动检查点引起数据库服务器触发更频繁的检查点，以避免事务阻塞。自动检查点尝试监视系统活动和资源使用情况（物理和逻辑日志使用情况以及缓冲池脏的程度）以能够及时地触发检查点，这样检查点的处理就可在物理日志或逻辑日志耗尽之前完成。

数据库服务器为逻辑日志空间的每个范围生成至少一个自动检查点。这保证了可开始快速恢复的检查点的存在。

使用 AUTO_CKPTS 配置参数可在数据库服务器启动时启用或禁用自动检查点。（可通过使用 gadmin -wm 或 gadmin -wf 来动态地启用或禁用自动检查点。）

手动检查点是您可以启动的基于事件的检查点。

数据库服务器提供了两种方法来确定发生意外中断时快速恢复所用时间。

- 使用 CKPTINTVL 配置参数可指定服务器触发检查点的频率。
- 使用 RTO_SERVER_RESTART 配置参数可指定快速恢复需要的时间。

  当您使用 RTO_SERVER_RESTART 配置参数时：

  - 数据库服务器忽略 CKPTINTVL 配置参数。
  - 数据库服务器监视物理和逻辑日志使用情况，以估计快速恢复的持续时间。如果服务器估计快速恢复将超出 RTO_SERVER_RESTART 配置参数中指定的时间，那么服务器将自动触发检查点。

RTO_SERVER_RESTART 配置参数可以是目标时间量，不能是保证的时间量。

可增加重新启动时间的多个因素也可影响快速恢复时间。这些因素包括回滚遇到意外中断时处于活动状态的长事务。

有关 RTO_SERVER_RESTART 和 AUTO_CKPTS 配置参数的更多信息，请参阅《GBase 8s 管理员参考》中有关配置参数的主题。

用于清空检查点之间缓冲池的 LRU 值

用于清空检查点之间缓冲池的 LRU 值对于检查点性能不是特别重要。在 BUFFERPOOL 配置参数中设置的 lru_max_dirty 和 lru_min_dirty 值通常仅对于维护页替换的足够清洁页是必需的。通过将 lru_min_dirty 设置为 70 和将 lru_max_dirty 设置为 80 开始。

如果检查点期间事务被阻塞，那么数据库服务器随后将尝试增加检查点频率以消除被阻塞的事务。当服务器搜索空闲页以执行页替换并且发生前台写入时，服务器随后将自动增加 LRU 清空频率以防止该事件再次发生。当数据库服务器完成页替换并找到后续访问的页时，服务器将自动增加 LRU 清空。对 LRU 清空所作的任何自动调整都不会持续到 onconfig 文件。

备份期间的检查点

如果执行备份，那么数据库服务器将运行检查点并将所有经更改的页清空到磁盘中。如果您执行复原，那么数据库服务器重新应用所有逻辑日志记录。

有关 ON-Bar 或 gtape 的信息，请参阅《GBase 8s 备份与复原指南》。

4.5.5 快速恢复

快速恢复是自动容错功能，数据库服务器每次从脱机方式转向静默、管理或联机方式时将执行该功能。无需为快速恢复执行任何管理操作；它是自动执行的功能。

快速恢复过程检查数据库服务器上次脱机时是否在不受控条件下进行的。如果是，那么快速恢复将数据库服务器返回至物理和逻辑一致性状态。

如果快速恢复过程发现数据库服务器在受控方式下脱机，那么快速恢复过程终止，并且数据库服务器转向联机方式。

请参阅检查点之后的快速恢复。

需要快速恢复

快速恢复在任何导致数据库服务器内存内容丢失的故障之后将数据库服务器复原至物理和逻辑一致性。例如，操作系统发生故障，但没有任何警告。系统故障不损坏数据库，但却影响发生故障时正在进行的事务。

启动快速恢复时的情境

每次管理员将数据库服务器从脱机方式转为静默、管理或联机方式时，数据库服务器都会检查是否需要快速恢复。

作为共享内存初始化的一部分，数据库服务器检查物理日志的内容。当数据库服务器在受控条件下关闭时，物理日志为空。 从联机方式转向静默方式的过程包含一个清空物理日志的检查点。因此，如果数据库服务器在物理日志中找到页，很明显数据库服务器会在不受控条件下脱机，并且快速恢复开始。

快速恢复和已缓冲日志记录

如果数据库使用已缓冲日志记录（如已缓冲的事务日志记录中所述），那么与已落实事务相关联的一些逻辑日志记录可能不会在发生故障时写入逻辑日志。 如果发生这种情况，快速恢复不会复原那些事务。快速恢复仅能复原那些在磁盘上的逻辑日志中存储有相关联的 COMMIT 记录的事务。（由于该原因，已缓冲日志记录代表性能和数据脆弱性之间的平衡。）

快速恢复期间可能的物理日志溢出

在快速恢复期间，物理日志可能会溢出。如果此情况发生，那么数据库服务器将尝试将物理日志空间扩展到名为 plog_extend.servernum 的磁盘文件。 此文件的缺省位置件 $GBASEDBTDIR/tmp。

使用 ONCONFIG 参数 PLOG_OVERFLOW_PATH 可定义用于创建此文件的位置。

当在快速恢复期间执行第一个检查点时，数据库服务器将除去 plog_extend.servernum 文件。

快速恢复和无日志记录

对于不使用日志记录的数据库或表，快速恢复将数据库复原到它在最近检查点之时的状态。自上一个检查点以来对数据库所作的所有更改都将丢失。

检查点之后的快速恢复

作为共享内存初始化的一部分，快速恢复将数据库服务器返回至一致状态。将存储所有已落实的事务，并将回滚所有未落实的事务。

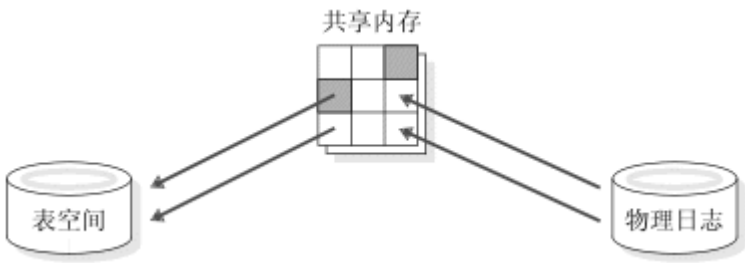
快速恢复按以下步骤发生：

1. 数据库服务器使用物理日志中的数据将所有磁盘页返回至它们在最近检查点时的状态。该点称为物理一致性。
2. 数据库服务器在逻辑日志文件中查找最新的检查点记录。
3. 数据库服务器前滚所有在最近检查点记录之后写入的逻辑日志记录。
4. 数据库服务器回滚所有未落实的事务。某些 XA 事务在 XA 资源管理器可用之前可能未解析。

服务器恢复到上一个检查点的状态

要将所有磁盘页返回到它们在最近检查点之时的状态，数据库服务器将物理日志中存储的前映像写入共享内存，然后写回到磁盘。物理日志中的每个前映像包含在检查点之后更新的页的地址。当数据库服务器将物理日志中的每个前映像页写到共享内存并又写回到磁盘时，自最近检查点时刻以来对数据库服务器数据的更改将撤销。现在数据库服务器在物理上是一致的。 下图说明了此步骤。

图： 将物理日志中的所有剩余前映像写回到磁盘



服务器在逻辑日志中查找检查点记录

在返回到上一个检查点状态之后，数据库服务器在逻辑日志中定位最近检查点记录的地址。最近检查点记录可保证位于磁盘上的逻辑日志中。

服务器前滚逻辑日志记录

在定位逻辑日志中的检查点记录之后，数据库服务器前滚在最近检查点记录之后写入的逻辑日志记录。此操作再现自检查上一个检查点以来直到发生不受控的关机之时对数据库的所有更改。下图说明了此步骤。

图：前滚自最近检查点以来写入的逻辑日志记录

在此图之前的段描述了此图的内容。

服务器回滚未落实的事务

在前滚逻辑日志记录之后，数据库服务器回滚系统失败时未落实的事务的所有逻辑日志记录。所有数据库逻辑一致，因为所有已落实事务已前滚并且所有未落实事务已回滚。某些 XA 事务在 XA 资源管理器可用之前可能未解析。

已完成两阶段落实的第一阶段的事务是例外情况。有关更多信息，请参阅两阶段落实协议如何处理故障。

因为可能有一个或多个事务跨及几个检查点而未落实，所以该回滚过程可能会越过最近检查点记录而往回读完逻辑日志。包含打开的事务的记录的所有逻辑日志文件可用于数据库服务器，因为直至日志文件包含的所有事务关闭才会释放该日志文件。

下图说明了回滚过程。此处，未落实的更改从逻辑日志回滚到特定磁盘上的数据库空间。当快速恢复完成时，数据库服务器将转回静默、管理或联机方式。

图： 回滚未完成的所有事务

4.6 管理物理日志

这些主题描述了以下过程：

- 更改物理日志的位置或大小
- 监视物理日志、物理日志缓冲区和逻辑日志缓冲区

- 监视并强制执行检查点

请参阅物理日志记录、检查点和快速恢复以获取背景信息。

4.6.1 更改物理日志的位置和大小

可以使用 glogadmin 实用程序来更改物理日志的位置和大小。

您可以移动物理日志文件来尝试提高性能。 在数据库服务器初始化磁盘空间时，它会将分配给逻辑日志和物理日志的磁盘页放在根数据库空间中。 您可以通过将物理日志和/或逻辑日志文件移至其他数据库空间来提高性能。

限制： 无法将逻辑或物理日志添加到没有缺省页大小的数据库空间中。

先决条件：

- 执行更改时，以用户 gbasedbt 或 root 身份（在 UNIX™ 上）登录。
- 通过运行 gcheck -pe 命令，确定是否有足够的连续空间可用。

为物理日志分配的空间必须是连续的。 当更改物理日志的大小或位置时，如果目标数据库空间包含的连续空间不足，那么服务器将不更改物理日志。此外，如果在初始化数据库服务器时物理日志没有足够的资源，初始化会失败。

要更改物理日志的大小和位置，请在数据库服务器处于管理、停顿或联机方式时运行以下命令：

```
glogadmin -p -s size -d dbspace -y
```

size

物理日志的新大小（以 KB 计）

数据库空间

指定物理日志要位于的数据库空间

以下示例更改物理日志的大小和位置。新的物理日志大小为 400 KB，并且该日志位于 dbspace6 数据库空间中：

```
glogadmin -p -s 400 -d dbspace6 -y
```

4.6.2 监视物理和逻辑日志记录活动

监视物理日志以确定在检查点出现前使用的物理日志文件的百分比。可使用这些信息来找出物理日志文件的最佳大小。该大小必须足够大，使数据库服务器无需过于频繁地强制执行检查点，也必须足够小，以节省磁盘空间并保证快速恢复。

监视物理日志和逻辑日志缓冲区以确定它们对于当前处理级别是否为最佳大小。要监视的重要统计信息就是每个磁盘的写入页数统计信息。

要监视物理日志文件、物理日志缓冲区和逻辑日志缓冲区，请使用以下命令。

实用程序	命令	更多信息
命令行 命令行或 ISA	gstat -l	第一行显示以下有关每个物理日志缓冲区的信息： • 已用的缓冲区页数 (bufused) • 每个物理日志缓冲区以页计的大小 (bufsize) • 向缓冲区写入的页数 (numpages) • 从缓冲区到磁盘的写入次数 (numwrits) • 向缓冲区写入的页数与向磁盘写入的次数之比率 (pages/IO) 第二行显示以下有关物理日志的信息： • 物理日志文件中首页的页码 (phybegin) • 物理日志文件以页计的大小 (physize) • 日志中要发生下次写入的当前位置，指定为页码 (physpos) • 日志中的已用页数 (phyused) • 已用的总物理日志页数的百分比 (%used) 第三行显示以下有关每个逻辑日志缓冲区的信息： • 已用的缓冲区页数 (bufused) • 每个逻辑日志缓冲区以页计的大小 (bufsize) • 向缓冲区写入的记录数 (numrecs) • 向缓冲区写入的页数 (numpages) • 从缓冲区到磁盘的写入次数 (numwrits) • 缓冲区中记录数与页数的比率 (recs/pages) • 向缓冲区写入的页数与向磁盘写入的次数之比率 (pages/IO)
命令行	glogadmin -p	移动物理日志或调整物理日志大小

实用程序	命令	更多信息
命令行或 ISA		
命令行 命令行或 ISA	gadmin -l	前进到下一逻辑日志文件。
ISA	**日志 > 逻辑**	单击**推进日志文件**。

有关 gstat -l 输出的示例的更多信息，请参阅《GBase 8s 管理员参考》。

有关使用 SQL 管理 API 命令（而不是一些 glogadmin 和 gadmin 命令）的信息，请参阅使用 SQL 管理 API 执行远程管理和《GBase 8s SQL 指南：语法》。

4.6.3 监视检查点信息

监视检查点活动以查看各种信息，包括线程需要等待检查点完成的次数。这些信息对于确定检查点间隔是否适当非常有用。

要监视检查点，请使用以下命令。

实用程序	命令	更多信息
gstat 实用程序	gstat -m	查看消息日志中的最近 20 行。 如果最近 20 行中不包含检查点消息，那么直接用文本编辑器读取消息日志。数据库服务器在检查点结束时将个别检查点消息写入日志。 如果发生了检查点，但数据库服务器没有页可写入磁盘，那么数据库服务器不会将任何消息写入消息日志。
gstat 实用程序	gstat -p	获取这些检查点统计信息： • numckpts：自数据库服务器联机以来发生的检查点数。 • ckptwaits：用户线程等待检查点完成的

实用程序	命令	更多信息
		次数。数据库服务器将防止用户线程在检查点期间进入临界区。
ON-Monitor (UNIX™)	**状态 > 概要文件**	**检查点数**和**检查等待次数**字段显示的信息与 **gstat -p** 中的 **numckpts** 和 ckpwaits 字段显示的信息相同。

打开或关闭检查点调整

要打开自动检查点调整，请发出 gadmin -wf AUTO_CKPTS=1 命令。要关闭自动检查点调整，请发出 gadmin -wf AUTO_CKPTS=0 命令。

强制执行检查点

如果必要，可以使用 gadmin 或 SQL 管理 API 命令来强制执行检查点。

在以下任何情况下强制执行检查点：

- 要释放包含最近检查点记录且已备份但尚未释放（ gstat -l 的 U-B-L 或 U-B 状态）的逻辑日志文件
- 在您发出 gadmin -sy 将数据库服务器置为静默方式之前
- 在构建较大索引后，如果数据库服务器在下一个检查点之前终止。索引构建将在下一次重新启动数据库服务器时重新启动。
- 如果检查点有很长时间未出现，而您要尝试进行可能中断数据库服务器的系统操作
- 如果前台写入将耗用比预期更多的资源（执行强制检查点临时将资源使用量降至零）
- 运行 dbexport 或卸载表之前，请确保在导出或卸载数据之前所有数据在物理上保持一致
- 在使用 PUT 或 INSERT 语句执行大量表的装入后（因为表的装入使用缓冲区高速缓存，强制执行检查点可清除高速缓存。）

要强制执行检查点，请运行 gadmin -c。

或者，如果使用的是 ON-Monitor (UNIX™)，请从主菜单中选择**强制执行检查点**选项。直至出现检查点，**上一个检查点完成**字段中的时间才会更改。上**一个检查点检查**字段显示上一个检查点检查的时间。如果自检查上一个检查点以来没有进行任何修改，那么数据库服务器不会执行检查点。

有关使用 SQL 管理 API 命令（而不是一些 gadmin 命令）的信息，请参阅使用 SQL 管理 API 执行远程管理和《GBase 8s SQL 指南：语法》。

服务器提供的检查点统计信息

数据库服务器提供有关前 20 个检查点的历史记录信息。可以通过 SMI sysckptinfo 表来访问这些信息。

SMI 表

查询 sysprofile 表以获取有关物理日志和逻辑日志缓冲区的统计信息。sysprofile 表还提供与 gstat -p 选项提供的信息相同的检查点统计信息。以下行包含以下统计信息。

plgpagewrites

写入物理日志缓冲区的页数

plgwrites

从物理日志缓冲区到物理日志文件的写入次数

llgrecs

写入逻辑日志缓冲区的记录数

llgpagewrites

写入逻辑日志缓冲区的页数

llgwrites

从逻辑日志缓冲区到逻辑日志文件的写入次数

numckpts

自数据库服务器变为联机后所出现的检查点数

ckptwaits

线程在检查点期间等待检查点完成进入临界区的次数

value

numckpts 和 **ckptwaits** 的值

4.6.4 打开或关闭自动 LRU 调整

使用 AUTO_LRU_TUNING 配置参数可在数据库服务器启动时启用或禁用自动 LRU 调整。

如果设置了 RTO_SERVER_RESTART 配置参数，数据库服务器将自动触发检查点，以便能在指定的时间内使服务器转为联机。如果服务器不满足 RTO_SERVER_RESTART 策略的要求，那么数据库服务器会打印消息日志中的警告信息。

要关闭特定会话的自动 LRU 调整，请发出 gadmin -wm AUTO_LRU_TUNING=0 命令。

要在会话期间关闭自动 LRU 调整后再将其打开，请发出 gadmin -wm AUTO_LRU_TUNING=1 命令

自动 LRU 调整更改会影响所有缓冲池并调整 BUFFERPOOL 配置参数中的 lru_min_dirty 和 lru_max_dirty 值。

5 容错

5.1 镜像

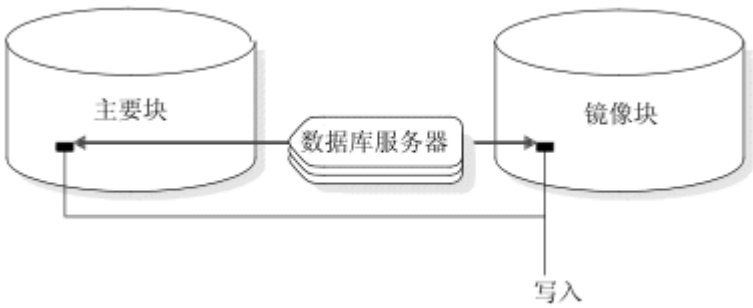
这些主题描述数据库服务器镜像功能。有关如何执行镜像任务的指示信息，请参阅使用镜像。

5.1.1 镜像

镜像是一种策略，用于将一个已定义的数据库空间、BLOB 空间或 BLOB 空间的主块与同等大小的镜像块联结成对。

每次写入主块将自动伴随向镜像块的相同写入。下图说明了此概念。如果在主块上发生故障，镜像可使您从镜像块读取或向镜像块写入，直至您可恢复主块，而不会中断用户对数据的访问。

图： 将数据同时写入主块和镜像块



在通过网络管理的磁盘上不支持镜像。 同一数据库服务器实例必须管理镜像集的所有块。

镜像的好处

如果发生介质故障，那么镜像将为数据库服务器管理员提供恢复数据但不必使数据库服务器脱机的方法。该功能可带来更大的可靠性和更少的系统停机时间。而且，如果对该数据

制作镜像的块位于不同介质上，那么应用程序可以继续从数据库读取或写入数据库（该数据库的主块位于受影响的介质上）。

所有关键数据库都必须位于镜像数据库空间中。 必须对包含数据库服务器保留页的根数据库空间制作镜像。

镜像的成本

磁盘空间成本和性能成本与镜像过程相关联。磁盘空间成本是由于存储镜像数据需要附加空间。性能成本源于对主块和镜像块都执行写入。 为磁盘写入使用多个虚拟处理器将降低该性能成本。根据块中的数据位置使用分开读取（数据库服务器通过分开读取从主块或镜像块读取数据）实际上对于只读数据可使性能提高。有关数据库服务器如何执行对镜像块的读写的更多信息，请参阅处理期间执行的操作。

不制作镜像的后果

如果您未对数据库空间制作镜像，那么在介质故障后必须从存储空间备份进行复原的频率会增加。

当镜像块受到介质故障影响时，数据库服务器仅从仍联机的块中读取，直至您将脱机块恢复为联机。当镜像对的第二个块脱机时，数据库服务器将无法访问存储在该块上的数据。如果块包含逻辑日志文件、物理日志或根数据库空间，那么数据库服务器将立即脱机。 如果块不包含逻辑日志文件、物理日志或根数据库空间，那么数据库服务器可以继续操作，但线程无法从脱机块中读取或写入脱机块。如果未镜像的块停止运行，那么您必须通过从备份恢复数据库空间来复原该块。

要制作镜像的数据

理想情况下必须对所有数据制作镜像。但如果磁盘空间成为问题，您可能无法对所有数据制作镜像。在这种情况下，请选择某些关键块来制作镜像。

关键块始终包括作为根数据库空间一部分的块、存储逻辑日志文件的块和存储物理日志的块。如果这些关键块中有任何一个发生故障，那么数据库服务器立即脱机。

如果某些块容纳那些对您的业务很关键的数据，那么对这些块授予高优先级以用于镜像。

此外，针对为那些存储常用数据的块制作镜像授予优先级。该操作确保当一个广泛使用的块脱机时许多用户的活动不会停止。

镜像备用方法

如本手册中的说明，镜像是一种数据库服务器功能。您的操作系统或硬件可能会提供备用的镜像解决方案。

如果您在考虑使用操作系统提供的镜像功能而不是数据库服务器镜像，那么在您决定使用哪种功能之前将两个功能的实现进行比较。镜像过程中最慢的步骤是将数据实际写入磁盘。数据库服务器以并行方式执行对镜像块的写入的策略可帮助减少该步骤所需的时间。（请参阅磁盘写入镜像块。）另外，数据库服务器镜像使用分开读取以提高读性能。（请参阅

磁盘从镜像块读取。）未使用并行镜像写入和分开读取的操作系统镜像功能可能提供的性能较差。

您也可以同时运行数据库服务器镜像和操作系统镜像。它们可独立于彼此运行。在某些情况下，您可能决定同时使用数据库服务器镜像和您的操作系统提供的镜像功能。例如，您可能在同一磁盘驱动器上既有数据库服务器数据又有其他数据。可以使用操作系统镜像对其他数据进行数据，使用数据库服务器镜像对数据库服务器数据制作镜像。

逻辑卷管理器

逻辑卷管理器是备用镜像解决方案。有些操作系统供应商提供此类型的实用程序，以便将多个磁盘视为一个文件系统。保存数据至两个以上的磁盘将给您更多保护以防止介质故障，但附加的写入会增加性能成本。

硬件镜像

另一个解决方案是使用硬件镜像，例如低价磁盘的冗余阵列 (RAID)。此类型硬件镜像的优势在于与数据库服务器所需相比，它需要更少的磁盘空间来存储相同的数据量以防止介质故障。

有些硬件镜像系统支持热交换。您可以在保持数据库服务器联机时交换坏磁盘。 建议在执行热交换之前减少 I/O 活动。

重要： 如果在使用硬件镜像时数据库服务器发生问题，请参阅操作系统或磁盘文档或技术支持以获取协助。

外部备份与复原

如果使用硬件磁盘镜像，与使用常规的 ON-Bar 命令相比，使用外部备份与复原可以使系统更快地联机。有关外部备份与复原的更多信息，请参阅《GBase 8s 备份与复原指南》。

5.1.2 镜像过程

本节更详细地描述了镜像过程。有关如何执行诸如创建镜像块、启动镜像过程、更改镜像块状态等镜像操作的指示信息，请参阅使用镜像。

创建镜像块

当您指定镜像块，数据库服务器就将所有数据从主块复制到镜像块。 该复制过程被称为恢复。一旦恢复完成，镜像过程就立即开始。

如果您对包含逻辑日志文件的数据库空间中的块开始制作镜像，那么将延迟恢复过程（该过程标记镜像开始）。 在您创建根数据库空间的 0 级备份之后，才会开始对包含逻辑日志文件的数据库空间制作镜像。延迟可确保如果包含这些逻辑日志文件的主块在数据库空间复原期间变为不可用，数据库服务器可使用镜像的逻辑日志文件。

0 级备份将更新的数据库服务器配置信息（包括有关新镜像块的信息）从根数据库空间保留页复制到备份。如果您执行数据复原，那么如果主块不可用，在备份开始时已更新的配

置信息会让数据库服务器查找逻辑日志文件的镜像副本。如果该新的存储空间备份信息不存在，那么数据库服务器无法利用已制作镜像的日志文件。

由于类似原因，您无法在创建数据库空间备份时对包含逻辑日志文件的数据库空间制作镜像。备份开始以后，不能复制数据库空间备份磁带的第一个块中必须包含的新信息。

有关创建镜像块的更多信息，请参阅使用镜像。

镜像状态标志

数据库空间、BLOB 空间和智能大对象空间具有指示它们已镜像还是未镜像的状态标志。

您必须在镜像开始前对根数据库空间执行 0 级备份。

块具有指示以下信息的状态标志：

- 块是主块还是镜像块
- 当前块是联机、脱机、新镜像的块（需要对根数据库空间进行 0 级备份）还是处于正在恢复的过程。

有关这些块状态标志的描述，请参阅《GBase 8s 管理员参考》中有关 gstat -d 选项的描述。有关如何显示这些状态标志的信息，请参阅监视磁盘使用量。

恢复

当数据库服务器恢复镜像块时，它执行与镜像开始时所使用过程相同的恢复过程。镜像恢复过程包括将数据从现有联机块复制到新的已修复的块，直至这两者完全相同。

当您启动恢复时，数据库服务器将脱机块置于恢复方式并将信息从联机块复制到恢复块。当恢复完成，该块会自动获取联机状态。无论您正在恢复镜像对的主块还是在恢复镜像块，均执行相同的步骤。

提示： 在恢复过程中您仍可使用联机块。如果将数据写入已经复制到恢复块的页，数据库服务器会在继续进行恢复过程之前在恢复块上更新相应页。

有关如何恢复脱机块的信息，请参阅恢复镜像块页面上有关恢复镜像块的信息。

处理期间执行的操作

这些主题说明了镜像块磁盘 I/O 的一些详细信息以及数据库服务器如何为这些块处理介质故障。

磁盘写入镜像块

在数据库服务器处理期间，数据库服务器通过为每次修改执行两个并行写入（一个写入主块，一个写入镜像块）来执行镜像。

磁盘从镜像块读取

因为数据的两个版本位于不同磁盘上，所以数据库服务器使用镜像来提高读性能。根据块的哪一半包含数据页的地址，将决定是从主块还是从镜像块读取数据页。此功能称为分开

读取。分开读取通过减少磁盘搜索时间来提高性能。因为磁盘头必须经过的最大距离减少了一半，所以减少了磁盘搜索时间。下图说明了分开读取。

图： 分开读取可减少磁盘头必须经过的最大距离。

检测介质故障

数据库服务器首先在打开块时检查返回码，然后在任何读取或写入后均检查返回码。无论何时数据库服务器检测到主（或镜像）块设备发生故障，它会将块状态标志设置为脱机 (D)。有关块状态标志的信息，请参阅镜像状态标志。

如果数据库服务器检测到主（或镜像）块设备发生故障，对保持联机的那个块仍继续读取和写入。即使管理员有意使其中一个块脱机，情况也是如此。

管理员恢复脱机块并将其返回至联机状态以后，会再次在主块和镜像块之间分开读取，并对两个块均进行写入。

块恢复

数据库服务器使用异步 I/O 以使恢复块所需的时间最小化。从联机块进行读取可与对脱机块的写入重叠进行，而非这两个过程逐次地发生。即，执行读取的线程无需等待执行写入的线程完成，就可读取更多数据。

停止镜像过程的结果

结束镜像过程时，数据库服务器会立即释放镜像块，使空间可用于重新分配。 结束镜像过程的操作仅需几秒钟。

在结束镜像过程后对根数据库空间创建 0 级备份，以确保将包含更新镜像块信息的保留页复制到备份。该操作使复原过程不会再假设镜像数据仍可用。

镜像块的结构

如下所示，镜像块与主块包含相同的控制结构：

- BLOB 空间块的镜像包含 BLOB 空间开销页。
- 数据库空间块的镜像包含数据库空间开销页。
- 智能大对象空间的镜像包含元数据页。

有关这些结构的信息，请参阅《GBase 8s 管理员参考》中磁盘结构和存储器章节中有关镜像块结构的部分。

假如采用监视块中说明的方法之一，磁盘空间使用的显示将始终指示镜像块已满（即使主块有可用空间）。充满的镜像块表明除充当主块的镜像以外，块中没有空间可用于其他用途。只要主块和镜像块联机，该状态就保持充满。

如果主块脱机并且镜像块成为主块，那么磁盘空间分配报告会准确描述新主块的填充度。

5.2 使用镜像

这些主题描述使用数据库服务器镜像功能所需的各种镜像任务。它概述了对数据制作镜像所需的步骤。

5.2.1 准备对数据制作镜像

本节描述如何在正在运行但未启用镜像功能的数据库服务器上为数据启动镜像过程。

要准备对数据制作镜像，请执行以下操作：

1. 使数据库服务器脱机并启用镜像。

   请参阅启用 MIRROR 配置参数。

2. 使数据库服务器恢复联机。

3. 为镜像块分配磁盘空间。

   您随时可以分配该磁盘空间，只要在下一步指定镜像块时磁盘空间可用。镜像块必须与相应主块位于不同的磁盘上。请参阅为镜像数据分配磁盘空间。

4. 选择要制作镜像的数据库空间、BLOB 空间或智能大对象空间，并为该存储空间中的每个主块指定镜像块路径名和偏移量。

   在执行这一步后镜像过程开始。为所有希望镜像的存储空间重复该步骤。请参阅使用镜像。

5.2.2 启用 MIRROR 配置参数

可以设置 MIRROR 配置参数来启用（或禁用）镜像。

启用镜像将启动镜像任务所需的数据库服务器功能。但是，当您启用镜像时，您并未启动镜像过程。直至您为数据库空间、BLOB 空间或智能大对象空间创建镜像块，镜像才会实际开始。请参阅使用镜像。

如果您计划为根数据库空间创建镜像，将其作为初始化的一部分，那么在初始化数据库服务器时启用镜像；否则，保持镜像禁用。如果您稍后决定对存储空间制作镜像，您可以更改 MIRROR 配置参数的值。

要为数据库服务器启用镜像，必须在 onconfig 中将 MIRROR 参数设置为 1。 MIRROR 的缺省值为 0，指示镜像已禁用。

如果您未使用镜像，那么不要将 MIRROR 参数设置为 1 。

要更改 MIRROR 的值，您可以在数据库服务器处于联机方式时使用文本编辑器编辑 onconfig 文件。更改 onconfig 文件后，将数据库服务器转为脱机并随后转为静默，以使更改生效。

5.2.3 为镜像数据分配磁盘空间

在您可以创建镜像块之前，必须为该用途分配磁盘空间。您可以为镜像块分配原始磁盘空间或熟文件空间。有关分配磁盘空间的说明，请参阅分配磁盘空间。

始终用不同的控制器（理想情况下）在不同于相应主块的磁盘上为镜像块分配磁盘空间。可以使用此设置在主块所在磁盘脱机时访问镜像块，反之亦然。

链接块 (UNIX)

使用 UNIX™ 链接 (ln) 命令将镜像块的实际文件或原始设备链接至镜像路径名。如果发生磁盘故障，您可以将新的文件或原始设备链接至该路径名，而不必实际更换故障磁盘后再将块恢复联机。

在磁盘故障后将块重新链接至设备

在 UNIX™ 上，如果实际镜像文件或原始设备所在的磁盘脱机，您可以将块重新链接至不同磁盘上的文件或原始设备。如果执行此操作，那么可在将故障磁盘恢复联机之前恢复镜像块。您可用于重新链接的常用 UNIX 命令显示在以下示例中。

原始设置由主 Root 块和镜像 Root 块组成，它们被链接至实际的原始磁盘设备，如下所示：

表 2. 将辅助数据库服务器更改为标准服务器后的自动重新启动步骤假设原始设备 /dev/rsd2b 所在磁盘已脱机。您可以使用 rm 命令除去相应的符号链接，如下所示：

```
rm /dev/mirror_root
```

此时可以将镜像块路径名重新链接至正在运行的磁盘上的原始磁盘设备，并继续恢复块，如下所示：

```
ln -s /dev/rab0a /dev/mirror_root
```

5.2.4 使用镜像

当您为数据库空间、BLOB 空间或智能大对象空间中的每个主块创建镜像块后，镜像开始。

当您创建镜像块，数据库服务器就将数据从主块复制到镜像块。 当该过程完成，数据库服务器开始镜像数据。 如果主块包含逻辑日志文件，数据库服务器不会在您创建镜像块后立

即复制数据，而是一直等待，直至您执行 0 级备份。有关此行为的说明，请参阅创建镜像块。

重要： 您必须始终启动对整个数据库空间、BLOB 空间或智能大对象空间的镜像过程。数据库服务器不允许您在数据库空间、BLOB 空间或智能大对象空间中选择特定块来镜像。您必须为空间中的每个块创建镜像块。

在执行以下操作后，启动对存储空间的镜像过程：

- 在系统初始化期间创建镜像根数据库空间
- 将数据库空间的状态从未镜像更改为已镜像
- 创建镜像的数据库空间、BLOB 空间或智能大对象空间

这些操作中的每一个操作均需要您为存储空间中的现有块创建镜像块。

初始化期间为根数据库空间制作镜像

如果您在初始化数据库服务器时启用镜像，您还可为 Root 块指定镜像路径名和偏移量。数据库服务器在初始化服务器时创建镜像块。但因为 Root 块包含逻辑日志文件，所以直至您执行 0 级备份镜像才会实际开始。

要指定 Root 镜像路径名和偏移量，请在启动数据库服务器之前在 onconfig 文件中设置 MIRRORPATH 和 MIRROROFFSET 的值。

如果未提供镜像路径名和偏移量，但希望为根数据库空间启动镜像过程，那么必须在初始化数据库服务器后更改根数据库空间的镜像状态。

更改镜像状态

您可以进行以下两种对镜像块状态的更改：

- 将镜像块从联机更改为脱机
- 将镜像块从脱机更改为恢复

仅当块是镜像对的一部分时才可以使块脱机或复原块。只要该对中的另一块处于联机状态，您就可以使主块或镜像块脱机。

有关如何确定块状态的信息，请参阅监视磁盘使用量。

5.2.5 管理镜像

您可以使用 gspaces 实用程序管理镜像。

在 UNIX™ 上，您还可使用 ON-Monitor 管理镜像。

有关 gspaces 语法的完整描述，请参阅《GBase 8s 管理员参考》中的 gspaces 实用程序的内容。

启动未镜像存储空间的镜像过程

您可以随时准备对数据库空间、BLOB 空间或智能大对象空间制作镜像。但直至您执行 0 级备份该镜像才会开始。

使用 gspaces 启动未镜像数据库空间的镜像过程

您可以使用 gspaces 实用程序为数据库空间、BLOB 空间或智能大对象空间启动镜像过程。例如，以下 gspaces 命令为数据库空间 **db_project**（它包含 **data1** 和 **data2** 两个块）启动镜像过程：

```
gspaces -m db_project\
-p /dev/data1 -o 0 -m /dev/mirror_data1 0\
-p /dev/data2 -o 5000 -m /dev/mirror_data2 5000
```

以下示例显示如何启动对名为 sp1 的数据库空间的镜像过程。您可以在命令中或在文件中指定主路径、主偏移量、镜像路径和镜像偏移量。

```
gspaces -m sp1 -f mirfile
```

mirfile 文件包含以下行：

```
/ix/9.3/sp1 0 /ix/9.2/sp1mir 0
```

在这一行中，/ix/9.3/sp1 是主路径，0 是主偏移量，/ix/9.3/sp1mir 是镜像路径，0 是镜像偏移量。

使用 ISA 启动镜像过程

要使用 Server Administrator (ISA) 启动镜像过程，请执行以下操作：

1. 选择**存储器** > **块**。
2. 选择数据库空间名称并单击启动镜像过程。

使用 ON-Monitor 启动未镜像数据库空间的镜像过程 (UNIX™)

使用 ON-Monitor **数据库空间** > **镜像**选项，为数据库空间启动镜像过程。

要选择您希望制作镜像的数据库空间，可在列表中将游标下移至正确的数据库空间并按下 **CTRL-B**。然后**镜像**选项为数据库空间中的每个块显示一个屏幕。您可以在此屏幕中输入**镜像**路径名和偏移量。

在为每个块输入信息后，按下 **ESC** 退出该选项。数据库服务器恢复新镜像块，这表示它将数据从主块复制到镜像块。如果块包含逻辑日志文件，那么将延迟恢复，直至您创建 0 级备份后。

启动新存储空间的镜像过程

您还可在创建新的数据库空间、BLOB 空间或智能大对象空间时启动镜像过程。

使用 gspaces 启动新空间的镜像过程

您可以使用 gspaces 实用程序创建镜像数据库空间。例如，以下命令创建带有初始块 /dev/chunk1 和镜像块 /dev/mirror_chk1 的数据库空间 db_acct：

```
gspaces -c -d db_acct -p /dev/chunk1 -o 0 -s 2500 -m /dev/mirror_chk1 0
```

另一个启动镜像过程的方法是选择通过实用程序建立索引 > gspaces -m。

使用 ISA 启动新空间的镜像过程

要使用 Server Administrator (ISA) 启动新存储空间的镜像过程，请执行以下操作：

1. 选择**存储器** > **空间**。
2. 单击**添加数据库空间**、**添加 BLOB 空间**或**添加智能大对象空间**。
3. 为镜像块输入路径和偏移量。

使用 ON-Monitor 启动新数据库空间的镜像过程 (UNIX™)

要使用镜像创建数据库空间，请选择数据库空间 > 创建选项。

此选项将显示一个屏幕，您可以在其中为新数据库空间指定主块的路径名、偏移量和大小以及镜像块的路径名和偏移量。

添加镜像块

如果您向已镜像的数据库空间、BLOB 空间或智能大对象空间添加块，您必须也添加相应的镜像块。

使用 gspaces 添加镜像块

您可以使用 gspaces 实用程序向数据库空间、BLOB 空间或智能大对象空间添加主块及其镜像块。以下示例将块 chunk2 添加至 db_acct 数据库空间。因为数据库空间已镜像，所以也添加镜像块 mirror_chk2。

```
gspaces -a db_acct -p /dev/chunk2 -o 5000 -s 2500 -m /dev/mirror_chk2 5000
```

使用 ISA 添加镜像块

要使用 Server Administrator (ISA) 添加镜像块，请执行以下操作：

1. 选择**存储器** > **块**。
2. 选择数据库空间名称并单击**添加块**。
3. 为镜像块输入路径和偏移量。

使用 ON-Monitor 添加镜像块 (UNIX™)

要添加镜像块，请选择**数据库空间** > **添加块**选项。

数据库空间 > **添加块**选项会显示一些字段，用于输入主块路径名、偏移量和大小以及镜像块路径名和偏移量。

使镜像块脱机

当镜像块脱机时，数据库服务器无法向其中写入或从其中读取。您可以使镜像块脱机，从而将块重新链接至不同设备。 （请参阅在磁盘故障后将块重新链接至设备。）

使块脱机与结束镜像过程不同。您为整个数据库空间结束镜像过程，这使数据库服务器为该数据库空间删除所有镜像块。

使用 gspaces 使镜像块脱机

您可以使用 gspaces 实用程序使块脱机。以下示例使一个块（它是数据库空间 **db_acct** 的一部分）脱机：

```
gspaces -s db_acct -p /dev/mirror_chk1 -o 0 -D
```

使用 ON-Monitor 使镜像块脱机 (UNIX™)

要使用 ON-Monitor 使镜像块脱机，请选择**数据库空间** > **状态**选项。

在游标位于包含您希望使之脱机的块的数据库空间上时，按下 **F3** 或 **CTRL-B**。

数据库服务器显示一个屏幕，列出数据库空间中的所有块。将游标移至您希望使之脱机的块，并按下 **F3** 或 **CTRL-B** 以更改状态（使其脱机）。

恢复镜像块

要开始对联机块中的数据制作镜像，您必须恢复脱机块。

使用 gspaces 恢复镜像块

可以使用 gspaces -s 实用程序恢复脱机块。例如，要恢复路径名为 /dev/mirror_chk1 且偏移量为 0 KB 的块，请发出以下命令：

```
gspaces -s db_acct -p /dev/mirror_chk1 -o 0 -O
```

使用 ISA 恢复镜像块

要使用 Server Administrator (ISA) 恢复镜像块，请选择**通过实用程序建立索引** > **gspaces** -s。

使用 ON-Monitor 恢复镜像块 (UNIX™)

要使用 ON-Monitor 来恢复脱机镜像块，请选择数据库空间 > 状态选项。

结束镜像过程

当您对数据库空间、BLOB 空间或智能大对象空间结束镜像过程时，数据库服务器立即释放该空间的镜像块。这些块立即可用于重新分配至其他存储空间。

只有用户 gbasedbt 和 root（在 UNIX™ 上）才能结束镜像过程。

如果数据库空间中的任一主块脱机，那么您无法结束镜像过程。结束镜像过程时，系统可以处于联机方式。

使用 gspaces 结束镜像过程

您可以用 gspaces 实用程序结束镜像过程。 例如，要结束根数据库空间的镜像过程，可输入以下命令：

```
gspaces -r rootdbs
```

另一个结束镜像过程的方法是选择**通过实用程序建立索引 > gspaces -r**。

使用 ON-Monitor 结束镜像过程 (UNIX™)

要使用 ON-Monitor 结束数据库空间或 BLOB 空间的镜像过程，请选择**数据库空间** > **镜像**选项。

选择已镜像的数据库空间或 BLOB 空间，并按下 **CTRL-B** 或 **F3**。

使用 ISA 结束镜像过程

要使用 Server Administrator (ISA) 结束镜像过程，请执行以下操作：

1. 选择**存储器** > **块**。
2. 选择数据库空间名称并单击**停止镜像过程**。

5.3 一致性检查

GBase 8s 数据库服务器用于检测硬件或操作系统错误所导致的数据库服务器故障或问题。它通过在其许多关键功能中执行断言来检测问题。断言是一种一致性检查，验证页、结构或其他实体与将另外假定的页、结构或其他实体相匹配。

当这些检查中有一个检查发现内容不正确时，数据库服务器将报告断言失败并在数据库服务器消息日志中写入文本以描述失败的检查。数据库服务器还在另外的文件中收集进一步的诊断信息，该文件可能对 GBase 8s 技术支持人员有用。

这些主题提供了一致性检查措施以及处理不一致的方法的概述。

5.3.1 执行定期的一致性检查

要从一致性检查获取最大好处并确保数据库空间备份的完整性，必须定期执行以下操作：

- 验证所有数据和数据库服务器开销信息是一致的。
- 检查消息日志是否在您验证一致性时有断言失败。
- 在您验证一致性后创建 0 级数据库空间备份。

以下主题描述了其中每个操作。

验证一致性

由于此检查需要一定时间并且此检查可能导致争用，因此请将此检查调度为在活动最少的时候执行。必须在创建 0 级备份之前执行此检查。

运行下表中显示的命令以作为一致性检查的一部分。

表 1. 检查数据一致性

验证类型	命令
系统目录表	gcheck -cc
数据	gcheck -cD dbname
扩展数据块数	gcheck -ce
索引	gcheck -cI dbname
保留页	gcheck -cr
逻辑日志和保留页	gcheck -cR
元数据和智能大对象	gcheck -cs

您可以在数据库服务器处于联机方式时运行这些命令中的每个命令。有关每个命令如何在检查对象时锁定它们以及哪些用户可以执行验证的信息，请参阅 《GBase 8s 管理员参考》 中的 gcheck。

在大多数情况下，如果这些验证过程中的一个或多个过程检测到错误，那么解决方案是从数据库空间备份复原数据库。但是，错误的来源也可能是硬件或操作系统。

验证系统目录表

要验证系统目录表，使用 gcheck -cc 命令。

每个数据库包含它本身的系统目录，该目录包含有关数据库表、列、索引、视图、约束、存储过程和特权的信息。

如果当验证完成时显示警告，那么该警告的唯一用途是提醒您找不到特定类型的记录。这些警告并非指示您的数据、系统目录或甚至数据库设计有任何问题。该警告仅指示不存在任何表的同义词；即，系统目录在表 syssyntable 中不包含记录。 例如，如果您为没有为任何表定义同义词的数据库验证系统目录表，那么可能显示以下警告：

```
WARNING: No syssyntable records found.
```

但是，如果您在验证系统目录表时接收到错误消息，那么情况就完全不同了。请立即与 GBase 8s 技术支持联系。

验证数据页

要确认数据页，请使用 gcheck -cD 命令。

如果数据页验证检测到错误，请尝试从指定表中卸载数据，删除该表，重新创建该表，并重新装入该数据。

验证扩展数据块

要确认每个数据库中的扩展数据块，请使用 gcheck -ce 命令。

扩展数据块一定不能重叠。如果该命令检测到错误，请从存储空间备份执行数据复原。

验证索引

如果索引已损坏，那么数据库服务器在查询中将无法使用索引。

可以通过使用 gcheck -cI 命令来验证数据库中每个表上的索引。

此外，调度程序任务 **bad_index_alert** 将查找已被服务器标记为已损坏的索引。此任务在每晚运行。对于此任务找到的每个已损坏索引，都会在 **sysadmin:ph_alert** 表中建立一个条目。

如果索引已损坏，请将其删除，然后重新创建。

验证逻辑日志

要确认逻辑日志和保留页，请使用 gcheck -cR 命令。

验证保留页

要确认保留页，请使用 gcheck -cr 命令。

保留页是位于根数据库空间初始块开始处的页。这些页包含主数据库服务器开销信息。如果该命令检测到错误，请从存储空间备份执行数据复原。

该命令可能会提供警告。在大多数情况下，这些警告让您注意的是您已经知道的情况。

验证元数据

对每个数据库运行 gcheck -cs 以验证数据库中所有智能大对象的元数据。如有必要，从数据库空间备份复原数据。

监视数据不一致性

如果一致性检查代码在数据库服务器操作期间检测到不一致，将向数据库服务器消息日志报告断言失败。（请参阅《GBase 8s 管理员参考》中的消息日志主题。）

消息日志和转储文件中的读断言失败

以下示例显示断言失败在消息日志中采用的格式。

```
Assert Failed: Short description of what failed
    Who: Description of user/session/thread running at the time
    Result: State of the affected database server entity
    Action: What action the database server administrator should take
See Also: file(s) containing additional diagnostics
```

See Also: 行中包含以下一个或多个文件名：

- af.xxx
- shmem.xxx
- gcore.xxx
- gcore.xxx
- /path name/core

在所有情况中，xxx 对与单个线程的断言失败相关联的所有文件而言是一个公共的十六进制数。文件 af.xxx、shmem.xxx 和 gcore.xxx 位于 ONCONFIG 参数 DUMPDIR 指定的目录下。

文件 af.xxx 包含发送到消息日志的断言失败消息的副本以及当前相关结构和数据缓冲区的内容。

仅当 ONCONFIG 参数 DUMPSHMEM 设置为 1 或 2 时，文件 shmem.xxx 才包含断言失败时数据库服务器共享内存的完整副本。

仅限 UNIX：在 UNIX™ 上，仅当 ONCONFIG 参数 DUMPGCORE 设置为 1 且您的操作系统支持 gcore 实用程序时，gcore.xxx 才包含数据库服务器虚拟进程（线程此时在其上运行）的核心转储。仅当 ONCONFIG 参数 DUMPCORE 设置为 1 时，core 才包含数据库服务器虚拟进程（线程此时在其上运行）的核心转储。core 文件的路径名是上次调用数据库服务器的目录。

验证表和表空间数据

要验证表和表空间数据，请对数据库或表使用 gcheck -cD 命令。

大多数的一般断言失败消息后面均跟随其他信息，其他信息通常包含检测到错误的表空间。如果此检查确认存在不一致情况，请从表中卸载数据，删除该表，重新创建该表，并重新装入该数据。否则，无需执行其他任何操作。

在许多情况中，数据库服务器当断言失败时会立即停止。但是，当失败看起来是特定于某个表或较小实体时，数据库服务器会继续运行。

当断言是由于数据库服务器代表用户访问的数据页上的不一致而失败时，还会将错误发送至应用程序进程。SQL 错误取决于正在进行的操作。但是，ISAM错误几乎始终是 -105 或 -172，如下所示：

```
-105 ISAM error: bad isam file format
-172 ISAM error: Unexpected internal error
```

有关消息的目标和内容的更多详细信息，请参阅《GBase 8s 管理员参考》中有关消息日志消息的主题。

保留一致的 0 级备份

在您执行验证一致性中描述的检查而不出错之后，请创建 0 级备份。保留该存储空间备份和所有后续的逻辑日志备份磁带，直至您完成下一次一致性检查。请在每次 0 级备份之前

执行一致性检查。如果不这样做，那么至少保留所有从存储空间备份恢复所需的磁带，该备份是在验证数据库服务器一致之后立即创建的。

5.3.2 处理损坏

本节描述数据库服务器系统损坏的一些症状以及数据库服务器或您（作为管理员）为解决问题可采取的操作。数据库中的损坏可能作为硬件或操作系统问题的后果而出现，或可能由于某些未知的数据库服务器问题而导致。损坏可能影响数据或数据库服务器开销信息。

查找损坏症状

可以通过多种不同的方法查找有关损坏的信息。

数据库服务器通过以下方式提醒用户和管理员可能有损坏：

- 报告给应用程序的错误消息，说明找不到页、表或数据库。如果由于基础数据或开销信息中的不一致而导致操作失败，那么以下错误之一将始终返回至应用程序：

```
-105 ISAM error: bad isam file format
-172 ISAM error: Unexpected internal error
```

- 断言失败报告将写入数据库服务器消息日志。它们始终指示包含可帮助您确定问题来源的附加诊断信息的文件。 请参阅验证一致性。
- gcheck 实用程序返回错误。
- ph_alert 表显示了有关已损坏索引的信息。

修复索引损坏

在第一次指示损坏时，运行 gcheck -cI 命令以确定损坏是否存在于索引中。

如果您在数据库服务器处于联机方式时检查索引，gcheck 检测损坏但不会提示您进行修复。如果存在损坏，您可以在处于联机方式（数据库服务器锁定表和索引）时使用 SQL 语句删除并重新创建索引。如果您在静默方式中运行 gcheck -cI 并检测到损坏，将提示您确认实用程序是否尝试修复该损坏。

修复块上的 I/O 错误

如果在数据库服务器操作期间发生 I/O 错误，那么发生了错误的块的状态将更改为脱机。

如果块脱机，那么 gstat -d 将显示主块的块状态为 PD-，镜像块的块状态为 MD-。有关 gstat -d 输出的示例，请参阅《GBase 8s 管理员参考》。

另外，消息日志列出了带有出错位置和建议的解决方案的消息。所列解决方案是可能的解决办法，但不一定能纠正该问题。

如果脱机块已镜像，那么数据库服务器继续使用镜像块操作。使用操作系统实用程序来确定脱机块的问题并纠正该问题。 然后您必须定向数据库服务器以复原镜像块数据。

有关恢复镜像块的信息，请参阅恢复镜像块。

如果脱机的块未镜像并且包含逻辑日志文件、物理日志文件或根数据库空间，那么数据库服务器将立即启动停止操作。否则，数据库服务器可以继续操作，但无法写入或读取脱机块或该块所在数据库空间中的任何其他块。必须采取步骤以确定 I/O 错误发生的原因、纠正问题并从备份复原数据库空间。

如果当某块标记为脱机 (D) 时将数据库服务器变为脱机方式，那么您可以重新启动数据库服务器，前提是标记为脱机的块不包含关键数据（逻辑日志文件、物理日志文件或根数据库空间）。

5.3.3 收集诊断信息

若干 ONCONFIG 参数会影响数据库服务器收集诊断信息的方式。因为断言失败通常指示不可预见的问题，所以无论何时发生问题都要通知 GBase 8s技术支持。 收集的诊断信息旨在为 GBase 8s 技术人员所使用。af.xxx 文件的内容和使用以及共享核心不再做进一步的说明。

要确定触发断言失败的问题的原因，那么您不能删除诊断信息（直至 GBase 8s 技术支持指示您可以这样做），这一点至关重要。af.xxx 文件通常包含解决问题所需的信息。

若干 ONCONFIG 参数会指示数据库服务器不管在何时检测到断言失败或数据库服务器进入结束序列，都保留诊断信息：

- DUMPDIR
- DUMPSHMEM
- DUMPCNT
- DUMPCORE
- DUMPGCORE

有关配置参数的更多信息，请参阅《GBase 8s 管理员参考》。

您可决定是否设置这些参数。诊断输出可能会使用大量磁盘空间。（准确内容取决于环境变量设置和您的操作系统。）输出元素可能包含共享内存的副本和核心转储。

提示：核心转储在断言失败时内存中的进程映像。 在有些系统上，核心转储包括共享内存的副本。核心转储仅在这种情况中才是有用的。

如果有磁盘空间约束，数据库服务器管理员可能更愿意编写脚本来检测诊断输出是否存在指定目录中并将该输出发送至磁带。该方法保留诊断信息并使已使用的磁盘空间量最小化。

5.3.4 禁用 I/O 错误

GBase 8s 将禁用 I/O 错误分为两大类：破坏性的和非破坏性的。当包含数据库的磁盘在某些方面受到损坏时，禁用 I/O 错误是破坏性的。该类型的事件会威胁数据的完整性，并且

数据库服务器会将块和数据库空间标记为脱机。数据库服务器禁止对损坏磁盘进行访问，直至您修复或替换该磁盘并执行物理复原和逻辑复原。

当错误不威胁数据的完整性时，禁用 I/O 错误是非破坏性的。当有人意外地断开电缆连接时，会发生非破坏性的错误，您有可能擦除了设置为指向块的符号链接，或可能磁盘控制器被损坏。

在数据库服务器将某 I/O 错误视为禁用错误之前，该错误必须符合两个标准。首先，错误必须是在数据库服务器尝试对至少具有以下某个特征的块执行操作时发生：

- 块没有镜像。
- 与所涉及的块成对的主块或镜像块脱机。

其次，错误必须是在数据库服务器尝试执行以下操作之一不成功时发生：

- 在块上搜索、读取或写入
- 打开块
- 验证有关首次使用的页的块信息是否有效

  数据库服务器在其打开块之后将该验证作为稳定情况检查而执行。

可以防止数据库服务器在您调查禁用 I/O 错误时将数据库空间标记为脱机。如果您发现问题是小问题（如电缆松动），您可以将数据库服务器变为脱机，然后再次变为联机，不用从备份复原受影响的数据库空间。如果您发现问题更为严重（如磁盘损坏），您可以使用 gadmin -O 将受影响的数据库空间标记为脱机并继续处理。

5.3.5 监视数据库服务器是否有禁用 I/O 操作

数据库服务器用两种方法向您通知禁用 I/O 错误：

- 消息日志
- 事件警报

用于监视禁用 I/O 错误的消息日志

发生禁用 I/O 错误时，数据库服务器会向消息日志发送一条消息。

消息为：

```
Assert Failed: Chunk {chunk-number} is being taken OFFLINE.
Who: Description of user/session/thread running at the time
Result: State of the affected database server entity
Action: What action the database server administrator should take
See Also: DUMPDIR/af.uniqid containing more diagnostics
```

如下表所述，结果和操作取决于 ONDBSPACEDOWN 的当前设置。

ONDBSPACEDOWN 设置	结果	操作

0	禁用数据库空间 {space_name}。	复原数据库空间 {space_name}。
	禁用 BLOB 空间 {space_name}。	复原 BLOB 空间 {space_name}。
1	数据库服务器必须停止。	关闭并重新启动数据库服务器。
2	数据库服务器在下一检查点时阻塞。	使用 gadmin -k 关机，或使用 gadmin -O 覆盖。

ONDBSPACEDOWN 的值对临时数据库空间没有影响。对于临时数据库空间，不管怎样设置 ONDBSPACEDOWN，数据库服务器将继续处理。如果临时数据库空间需要修订，可将其删除并重新创建。

有关解释数据库服务器发送至消息日志的消息的更多信息，请参阅《GBase 8s 管理员参考》中有关消息日志消息的主题。

用于监视禁用 I/O 错误的事件警报

当数据库空间引发禁用 I/O 错误时，数据库服务器会将指定值作为参数传递给事件警报可执行文件。

事件警报值：

严重性

4（紧急）

类

5

类消息

Dbspace is disabled: 'dbspace-name'

特定消息

Chunk {chunk-number} is being taken OFFLINE.

事件标识

5001

如果您想要数据库服务器使用事件警报向您通知禁用 I/O 错误，可编写脚本，数据库服务器在检测到禁用 I/O 错误时将执行该脚本。有关如何设置您编写的此可执行文件的信息，请参阅 《GBase 8s 管理员参考》中有关事件警报的附录和有关配置参数的主题。

无坏扇区映射

GBase 8s 依赖您的主计算机的操作系统来进行坏扇区映射。 数据库服务器在从系统调用接收到失败返回码时知道了坏扇区或坏磁道。当发生这种情况时，数据库服务器将数次重新尝试访问，以确保该情况不是虚假情况。 如果确认了该情况，数据库服务器将尝试了读或写的块标记为脱机。

数据库服务器无法采取任何操作来识别坏的柱面、磁道或扇区位置，因为仅有的可用信息是尝试 I/O 操作的块中的字节位移。

如果数据库服务器在未制作镜像的块上检测到 I/O 错误，将会将该块标记为脱机。如果脱机块包含逻辑日志文件、物理日志或根数据库空间，那么数据库服务器将立即启动停止操作。否则，数据库服务器可以继续操作，但在复原其数据库空间之前应用程序无法访问脱机块。

6 高可用性和可伸缩性

成功的生产环境要求数据库系统始终可用，计划的中断要降到最少（如果有），并且数据库系统还可随着业务需求的变化而快速轻松地伸缩。

计划或意外中断期间，公司必须提供对数据资源的连续访问。计划的中断包括定期维护软件或硬件。意外中断是指意外的系统故障，如断电、断网、硬件故障、操作系统或其他软件错误。如果发生灾难，如地震或海啸，可能出现大范围的系统故障。

公司希望避免系统中的服务器过载，以确保数据可用性和防止服务拒绝攻击等。

公司还希望在业务增长时、季节性业务高峰期和月末或年终处理期间，快速轻松地扩展系统。

具有以下一项或多项功能的系统可轻松应对中断，并可提高数据可用性：

冗余

系统维护辅助服务器（充当主服务器的备份且在发生故障时接管主服务器）的能力。

故障转移

系统将所有工作负载从失败的服务器转移到另一个服务器的能力。

工作负载均衡

系统自动将客户机请求转向工作负载能力最高的服务器的能力。

可伸缩性

系统利用更多资源（如处理器、内存或磁盘空间）的能力。

通过维护将影响最小化

快速轻松维护所有服务器，以便尽可能降低用户应用程序受到的影响的能力。

6.1 高可用性和可伸缩性策略

可定制 GBase 8s 数据库软件来创建适当的高可用性和可伸缩性解决方案，以满足业务目标和环境需求。

要确定为实现高可用性和可伸缩性而定制数据库系统的最佳方法，必须确定有助于实现业务目标的策略。可使用适当的 GBase 8s 技术和组件来为这些策略提供支持。

6.1.1 支持高可用性和可伸缩性的组件

GBase 8s 数据库软件中包含若干可定制的组件，用于创建提供不中断连续服务的系统，以便将停机时间和维护降到最低。

- 集群
- Enterprise Replication
- 连接管理器
- 网格

集群

集群由通过网络连接的单个主服务器和一个或多个辅助服务器组成。如果集群中的主服务器出现故障，辅助服务器可接管主服务器的角色。

主服务器是拥有数据主副本且与集群中所有辅助服务器相连的数据库服务器。辅助服务器是与主服务器相连且包含或具有与主服务器相同的数据访问权的数据库服务器。集群中的主服务器和辅助服务器都配置了相同的硬件和软件。

应用程序可在任何服务器上启动。应用程序无需“感知”任何特定服务器。主服务器和辅助服务器之间的通信是安全的。此外，客户机应用程序与所有服务器之间的通信也是安全的。

辅助服务器可位于主服务器附近（同一房间，同一层或同一建筑物中），也可在地理上远离主服务器（如位于另一建筑物，另一个城市或另一个国家/地区）。 有些辅助服务器与主服务器具有相同的数据访问权，而另一些辅助服务器通过同步与主服务器包含相同数据。主服务器上落实了事务之后，主服务器会将全部事务发送到辅助服务器。主服务器不发送未落实的事务，所以数据库数据是可靠的。有关辅助服务器的描述，请参阅表 1。这些辅助服务器与主服务器同步，以构建连续可用的可伸缩数据存储。

表 1. 辅助服务器的类型

辅助服务器类型	描述
共享磁盘 (SD) 辅助服务器	与主服务器共享磁盘空间的服务器。这种类型的辅助服务器不在自己的磁盘空间中维护物理数据库副本；而是与主

辅助服务器类型	描述
	服务器共享磁盘。
高可用性数据复制（HDR）辅助服务器	该服务器通过同步或异步数据复制维护整个主服务器的备份副本。 如果主服务器出现故障，应用程序可快速访问这种类型的辅助服务器。
远程独立（RS）辅助服务器	该服务器维护数据的完整副本，并通过安全网络连接从主服务器异步传输更新。这种类型的辅助服务器在地理位置上可与主服务器之间有很远的距离，并且可在灾难恢复场景中充当远程备份服务器。

下面是集群的优势：

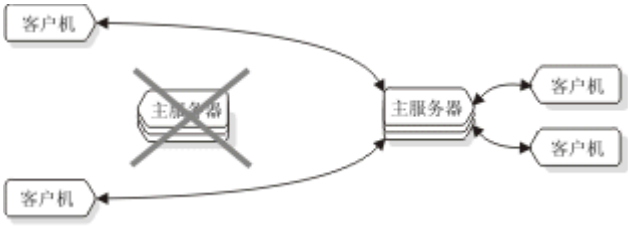
- 接收复制数据的站点上的客户机性能有了提高，因为那些客户机可以本地访问数据，而不是通过网络连接至远程数据库服务器。
- 所有站点上的客户机可感受复制数据可用性有所提高。如果复制数据的本地副本不可用，客户机还可访问数据的远程副本。

如下图所示，主数据库服务器管理的数据发生更改后，辅助数据库服务器上会随之动态更新。

图： 集群配置中的主数据库服务器和辅助数据库服务器

如果某个数据库服务器发生故障（如下图中所示），那么可以将使用该数据库服务器的客户机重定向至该对中的其他数据库服务器，这将成为主服务器。

图： 发生故障后数据复制配置中的数据库服务器和客户机



Enterprise Replication

基于日志记录的一种异步工具，用于在 GBase 8s 数据库服务器之间复制数据。 Enterprise Replication 实施异步数据复制，从而容许网络和目标数据库服务器中断。如果数据库服务器或网络发生故障，本地数据库服务器将继续为本地用户提供服务。在远程服务器可用之前，本地数据库服务器将已复制的事务存储在持久存储中。源服务器上的 Enterprise Replication 通过以下方法捕获要复制的事务：读取逻辑日志，存储事务，然后以可靠方式将每个事务作为复制数据传输到目标服务器。

Enterprise Replication 确保所有数据到达适当的服务器，而无论网络或系统是否发生故障。如果发生故障，Enterprise Replication 将存储数据，直到网络或系统恢复运行。 Enterprise Replication 将数据复制和发送量降到最低，从而有效复制数据。

连接管理器

连接管理器是用于自动管理和定向客户机连接请求的程序。它根据系统管理员配置的，称为服务级别协议 (SLA) 的重定向规则来重定向请求。

可使用连接管理器故障转移配置功能来配置自动故障转移。如果连接管理器检测到主服务器出现故障，并且在接下来的超时周期期间主服务器未执行任何操作来重新连接，那么最适合的辅助服务器将转换为主服务器。可通过使用配置文件来配置连接管理器的故障转移参数。

连接管理器执行可配置的负载均衡，这样客户机将基于服务器的工作负载进行重定向。连接管理器连接到集群中的每台服务器，收集有关服务器类型、未使用的工作负载容量及当前服务器状态的统计信息。 通过这些信息，连接管理器将客户机连接重定向到活动量最少的服务器。

可在不同计算机上配置多个连接管理器实例，以避免连接管理器成为单个故障点。可以在 GBase 8s 服务器实例上安装和运行连接管理器，或者为了将连接管理器从数据库服务器的故障中隔离开，也可以将其安装到单独的非 GBase 8s 机器上。此外，还可将连接管理器配置为在集群以外的机器的硬件平台上运行。

要进一步增加可用性，可配置多个连接管理器实例。要避免连接管理器成为单个故障点，配置多个连接管理器实例尤为重要。发生连接管理器故障之后，客户机仍然保持与服务器的连接；但是，除非配置了另一个连接管理器实例来充当备份，否则新客户机连接不能连接到服务器。如果连接管理器出现故障，配置另一个连接管理器实例来充当备份之前，所有客户机连接都将丢失。

除了在高可用性集群内执行客户机应用程序重定向之外，还可使用连接管理器将连接请求路由到复制集中的一个或多个服务器。配置连接管理器以指定服务器的名称和有序列表（这些服务器全部位于同一个 Enterprise Replication 域中），然后指定服务级别协议 （SLA） 名称和重定向策略。重定向策略指定客户机将连接的目标数据库服务器的类型或名称。客户机应用程序使用 SLA 名称连接到正确的数据库服务器。

网格

网格是一种互连的复制服务器的命名集合，用于将命令从授权服务器传播到集合中的其余服务器。与电网在区域内分配电力非常相似，您定义的网格将 SQL 命令分发给网格中的复制服务器。如果您有多个复制服务器，并且经常需要在每个复制服务器上执行相同任务，那么网格可能非常有用。可通过网格轻松运行以下类型的任务：

- 管理服务器；例如，添加块、除去逻辑日志或更改配置参数设置
- 更新数据库模式；例如，变更表或添加表
- 运行或创建存储过程或用户定义的例程
- 更新数据；例如，基于条件清除旧数据或更新值
- 维护复制：创建表时启用复制，并且在变更复制的表时变更复制定义。

数据复制的优点

数据复制的优势并非不需成本。数据复制显然需要更多存储，并且更新复制数据可能比更新单个对象要花费更多处理时间。

可以通过明确指定何处数据必须更新，从而以客户机应用程序的逻辑实现数据复制。但这种实现数据复制的方法成本高昂、容易出错且难以维护。相反，数据复制的概念通常与复制透明度相结合。 将复制透明度构建到数据库服务器（而非构建到客户机应用程序）以自动处理定位和维护数据副本的详细信息。

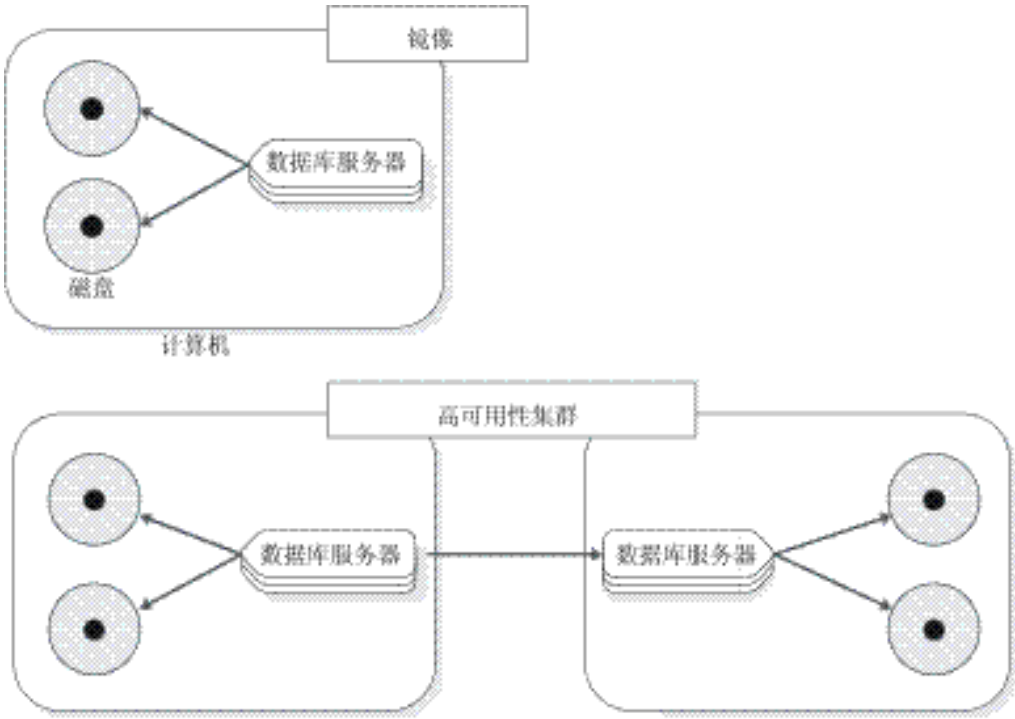
集群与镜像

集群和镜像是提高容错性的透明方法。

镜像中所述的镜像是单个数据库服务器用以维护不同磁盘上的特定数据库空间副本的机制。该机制保护镜像数据库空间中的数据免受磁盘故障的影响，因为数据库服务器自动更新两个磁盘上的数据，并且在一个数据库空间发生故障时自动使用另一磁盘。

或者，集群在一个完全独立的数据库服务器上复制另一个数据库服务器管理的所有数据，而不仅仅是复制指定的数据库空间。因为集群涉及两个不同的数据库服务器，所以它不仅可保护这两个数据库服务器管理的数据免受磁盘故障影响，而且可保护这些数据免受所有类型的数据库服务器故障影响（包括计算机故障或整个站点的灾难性故障）。

图: 镜像和集群的比较



集群与两阶段落实

两阶段落实协议（在多阶段落实协议中详细描述）确保跨多个数据库服务器统一落实或回滚事务。

理论上，可以利用两阶段落实来复制数据，方法是配置使用相同数据的两台数据库服务器并随后在其中一台数据库服务器上定义触发器，用于复制另一数据库服务器的更新。但是，这种实现在不同故障场合中有很多同步问题。 另外，分布式事务的性能比集群的性能要差。

集群和 ON-Monitor

可以将数据集群与 Enterprise Replication 组合使用，以创建强大的复制系统。集群通过为关键复制节点提供备份数据库服务器，从而确保Enterprise Replication 系统保持完全连接。

将集群与 Enterprise Replication 组合使用时，只有主服务器会连接至 Enterprise Replication 系统。除非主服务器发生故障，否则辅助服务器不会参与 Enterprise Replication。

有关更多信息，请参阅《GBase 8s Enterprise Replication 指南》和Enterprise Replication 作为可恢复组的一部分。

集群中复制的数据类型

高可用性集群复制可复制数据库空间和智能大对象空间中的数据，但不会复制 BLOB 空间中的数据。

所有内置和扩展数据类型都将复制到辅助服务器。必须记录用户定义的类型 (UDT)，并且这些类型必须位于单一数据库服务器中。 如果行外数据存储在智能大对象空间中或在同一数据库服务器上的不同表中，那么复制带有行外数据的数据类型。 对于要复制的存储在智能大对象空间中的数据，必须记录智能大对象空间。

不复制操作系统文件中存储的数据，或不复制与用户定义的例程关联的持久外部文件或内存对象中存储的数据。

用户定义的类型和用户定义的例程有特殊的安装和注册需求。有关指示信息，请参阅数据初始复制的工作原理。

主要和辅助数据库服务器

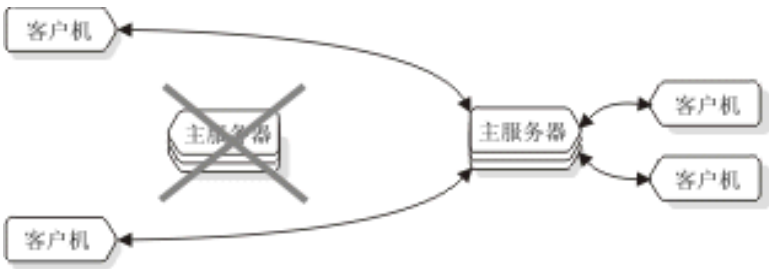
当您配置一组数据库服务器以使用数据复制时，一个数据库服务器被称为主数据库服务器，其他数据库服务器被称为辅助数据库服务器。（在此上下文中，不使用数据复制的数据库服务器称为标准数据库服务器。）辅助服务器可包含 SD 辅助服务器、RS 辅助服务器和 HDR 辅助服务器的任何组合。

如下图所示，主数据库服务器管理的数据发生更改后，辅助数据库服务器上会随之动态更新。

图: 数据复制配置中的主数据库服务器和辅助数据库服务器

如果某个数据库服务器发生故障（如下图中所示），那么可以将使用该数据库服务器的客户机重定向至该对中的其他数据库服务器，这将成为主服务器。

图: 发生故障后数据复制配置中的数据库服务器和客户机



6.1.2 透明缩放与工作负载平衡策略

GBase 8s 服务器可轻松伸缩，也可动态执行工作负载均衡，以确保对资源进行最优使用。

GBase 8s 可实现以下目标：

- 定期增加容量
- 增加报告容量实现的地理分散处理
- 均衡工作负载以优化资源的使用

定期增加容量

如果业务环境遇到了高峰期，可能需要定期增加容量。可以通过添加远程独立辅助服务器来增加容量。这种类型的辅助服务器维护数据的完整副本，并通过安全网络连接从主服务器异步传输更新。如果数据量极大，难以为其创建多个副本，那么可使用共享磁盘服务器，而不是远程独立辅助服务器。如果只需为报告（只读）工作负载增加容量，可使用高可用性数据复制 (HDR) 辅助服务器。

表 1. 共享磁盘辅助服务器提供的可伸缩性

优点	潜在缺点
• 可用性极高。这种辅助服务器与主服务器共享磁盘。	• 无故障转移。这种辅助服务器可能会配置为与主服务器在同一计算机硬件上运行。 • 无数据冗余。此辅助服务器不维护数据副本。（使用 SAN 设备进行磁盘存储。） • 主辅助服务器和辅助服务器需要相同的硬件、操作系统和数据库服务器产品版本。

由于以下原因而使用 SD 辅助服务器：

- 增加的报告容量

  多台辅助服务器使您能够在不影响主服务器的情况下卸载报告功能。
- 服务器故障备份

  如果主服务器出现故障，那么 SD 辅助服务器可快速并轻易地提升为主服务器。例如，如果您正在使用提供足够和可靠磁盘存储器的 SAN（存储区域网络）设备，但是却担心服务器故障，那么 SD 辅助服务器可提供可靠的备份。

表 2. 远程独立辅助服务器提供的可伸缩性

优点	潜在缺点
• 数据冗余。这种辅助服务器维护数据的副本。 • 故障转移。辅助服务器可在地理上远离主服务器，如位于另一建筑物，另一个城市或另一个国家/地区。 • 无需更改应用程序。服务器出现故障时，可自动切换客户机与主服务器或辅助服务器的连接。	• 主辅助服务器和辅助服务器需要相同的硬件、操作系统和数据库服务器产品版本。

增加报告容量实现的地理分散处理

如果公司在多处具有办事处，那么可能希望使用本地服务器来处理本地请求，而不是依赖于单个集中服务器。在这种情况下，可设置 Enterprise Replication 服务器网络。这样将容许目标数据库服务器中断。如果数据库服务器或网络出现故障，本地数据库服务器将继续为本地用户提供服务。在远程服务器可用之前，本地数据库服务器将已复制的事务存储在持久存储中。源服务器上的 Enterprise Replication 通过以下方法捕获要复制的事务：读取逻辑日志，存储事务，然后以可靠方式将每个事务作为复制数据传输到目标服务器。

表 3. Enterprise Replication 提供的可伸缩性

优点	潜在缺点
• 服务器可位于另一建筑物，另一个城市或另一个国家/地区。 • 服务器可安装在不同硬件上。 • 服务器可在不同操作系统上运行。 • 服务器可运行不同版本的数据库服务器产品。 • 可复制少量数据（基于日志的异步复制）。 • 使用多个连接管理器执行客户机自动重定向，从而通过添加共享磁盘辅助服务器来协助复制服务器。	• 可能存在冲突。 • 事务可能失败。在这种情况下，必须修复不一致的数据。

由于以下原因而在您的环境中使用 RS 辅助服务器：

- 增加的服务器可用性

  一个或多个 RS 辅助服务器通过维护多个可用于增加可用性的服务器来提供进一步的保证。
- 地理位置上的远程备份支持

  通常需要将辅助服务器安装在距离站点一段距离的位置，以备在最坏情况下灾难恢复场景所用。RS 辅助服务器是理想的远程备份解决方案。 如果辅助服务器位于 WAN（广域网）上，那么主 HDR 和辅助 HDR 之间的高级别调整可引起性能问题。使主服务器与辅助服务器保持相对密切的关系可以简化维护并将对性能的影响降到最低。
- 改善的报告性能

  多台辅助服务器使您能够在不影响主服务器的情况下卸载报告功能。此外，RS 辅助服务器配置使得分开报告需求和 HA 需求变得更容易，从而产生更适合于这两种环境的解决方案。
- 不稳定网络上的可用性

  如果检查点同步到达，那么较慢或不稳定的网络环境可导致主服务器和辅助服务器都延迟。RS 辅助服务器配置使用全双工联网，而且不需要这种调整。 如果在主服

务器和 RS 辅助服务器之间的网络性能未达到最佳状态，那么 RS 辅助服务器将是一种极具吸引力的解决方案。

均衡工作负载以优化资源的使用

创建或修改服务级别协议 (SLA) 时，可配置工作负载均衡。GBase 8s 从集群中的每个服务器收集信息，并将客户机应用程序自动连接到活动量最少的服务器。

可在集群内创建特定于某些应用程序类型的组，如用于联机事务处理 (OLTP) 或仓库的应用程序。 应用程序可选择连接到特定组，以为每种类型的查询实现最佳性能。

6.1.3 高可用性策略

可配置 GBase 8s，以在各种业务情况中最大化可用性。

目标	策略	优点	潜在缺点
在服务器出现故障时保护系统	使用与主服务器共享磁盘空间的辅助服务器。（共享磁盘辅助服务器）	• 可用性极高。此辅助服务器与主服务器具有相同的数据访问权。如果主服务器出现故障，辅助服务器可快速接管。 • 数据库始终处于同步状态，因为此辅助服务器与主服务器具有相同的数据访问权。 • 无需更改应用程序。服务器出现故障时，可自动切换客户机与主服务器或辅助服务器的连接。	• 此辅助服务器与主服务器位于同一台计算机上。 • 无数据冗余。此辅助服务器不维护数据副本。（使用 SAN 设备进行磁盘存储。） • 主辅助服务器和辅助服务器需要相同的硬件、操作系统和数据库服务器产品版本。 • 辅助服务器硬件必须可以处理与主服务器相同的负载。如果辅助服务器过小，可能会影响主服务器的性能。

在站点出现故障时进行保护	• 使用维护数据库服务器和数据副本的辅助服务器。（高可用性数据复制服务器） • （也可使用 RSS 和 ER）	• 可用性极高。应用程序如果不能连接到主服务器，可快速访问此服务器。 • 数据同步复制。 • 提高了可伸缩性 • 无需更改应用程序	• 位于主服务器本地 • 需要数据的精确副本（包括表和数据库模式）。 • 主辅助服务器和辅助服务器需要相同的硬件、操作系统和数据库服务器产品版本。
多级别站点故障保护	• 使用地理上远离主服务器且从主服务器异步更新的辅助服务器。（远程独立辅助服务器） • （也可使用 ER）	• 可用性极高。应用程序如果不能连接到主服务器，可快速访问此服务器。 • 数据异步复制。 • 提高了可伸缩性 • 无需更改应用程序	
站点故障保护实现的地理分散处理	ER 和 HDR	ER 和 ER 的备份	需要多个连接管理器

6.2 高可用性集群配置

这些主题描述了如何为 GBase 8s 规划、配置、启动和监视高可用性集群，以及如何在介质发生故障后复原数据。 如果计划使用高可用性集群，请首先阅读本节中的所有主题。如果计划使用 GBase 8s Enterprise Replication，请参阅《 GBase 8s Enterprise Replication 指南》。

高可用性和可伸缩性说明了什么是高可用性集群、其工作原理，以及如何为集群环境设计客户机应用程序。

6.2.1 高可用性集群规划

在您开始设置计算机和数据库服务器以使用高可用性集群之前，您可能想要做一些初步规划。以下列表包含要执行的规划任务：

- 选择并获取适当的硬件。
- 如果是使用多个数据库服务器来存储要复制的数据，那么迁移并重新分发数据，使其可以由单一数据库服务器管理。
- 确保希望复制的所有数据库使用事务日志记录。要打开事务日志记录，请参阅管理数据库日志记录方式。
- 开发客户机应用程序以同时利用复制对中的两个数据库服务器。有关设计注意事项的说明，请参阅数据复制客户机的重定向和连接和设计数据复制组客户机。
- 为第一次启动 HDR 创建调度。
- 为主数据库服务器设计存储空间和逻辑日志备份调度。
- 为如何处理其中一个数据库服务器的失败以及如何在失败后重新启动 HDR 生成一个计划。请阅读数据复制客户机的重定向和连接。

6.2.2 配置集群

要将系统配置为高可用性集群，必须执行以下操作：

- 满足硬件和操作系统要求。
- 满足数据库和数据要求。
- 满足数据库服务器配置要求。
- 配置连接。

以上每个主题均在本节中进行了说明。

您可以配置系统以便使用安全套接字层 (SSL) 协议进行 HDR 通信，SSL 协议是一个可以确保基于网络传送数据的隐私和完整性的通信协议。 在高可用性配置中，您可以使用 SSL 协议以实现主服务器和辅助服务器的连接，以及与远程独立 (RS) 服务器和共享磁盘 (SD) 辅助服务器的连接。

集群的硬件和操作系统需求

要使高可用性集群运行，硬件必须满足特定需求。

硬件必须满足以下需求：

- 主服务器和辅助服务器必须能够运行相同的 GBase 8s 可执行映像，即便主服务器和辅助服务器没有相同的硬件或操作系统也不例外。 例如，可以使用具有不同 Linux™ 32 位操作系统的服务器，因为这些操作系统可以运行相同的 GBase 8s 可执行映像。在这种情况下，不能添加 Linux 64 位操作系统的服务器，因为该操作系统需要不同的 GBase 8s 可执行映像。 检查机器说明文件：可以使用相同机器说明文件

中列为支持的硬件和操作系统的任意组合。

- 运行主数据库服务器和辅助数据库服务器的硬件必须支持网络能力。
- 分配给主数据库服务器和辅助数据库服务器的数据库空间的磁盘空间量必须相等。磁盘空间类型是不相关的；您可以在两个数据库服务器上使用任何原始或熟空间组合。
- 每台计算机上的块具有相同的路径名。允许将符号链接用于 UNIX™ 平台。

集群的数据库和数据需求

要使高可用性集群运行，数据库和数据必须满足特定需求。

数据库和数据必须满足以下需求：

- 必须记录所有数据。

  所有希望复制的数据库必须将事务日志记录打开。

  该要求很重要，因为辅助数据库服务器使用主数据库服务器的逻辑日志记录来更新它管理的数据。如果主数据库服务器管理的数据库不使用日志记录，那么对那些数据库的更新不会生成日志记录，这样辅助数据库服务器就没有更新复制数据的手段。日志记录可以是已缓冲日志记录，也可以是未缓冲日志记录。

  如果必须在启动 HDR 之前打开事务日志记录，请参阅使用 gtape 打开事务日志记录。
- 数据必须位于数据库空间或智能大对象空间中。

  如果主数据库服务器具有存储在 BLOB 空间中的简单大对象，那么在那些 BLOB 空间中对数据的修改将不会作为正常的 HDR 处理的一部分进行复制。但要复制数据库空间中的简单大对象数据。

  复制存储在智能大对象空间中的智能大对象。 必须记录智能大对象空间。复制用户定义的类型 (UDT)，除非它们有存储在操作系统文件中的行外数据。 如果行外数据存储在智能大对象空间中或在同一数据库服务器上的不同表中，那么复制带有行外数据的数据类型。
- 辅助服务器不得使用磁盘压缩。

  如果使用 GBase 8s 磁盘压缩功能，源表中的压缩数据在目标表中也处于压缩状态。不能在 HDR 辅助服务器、RS 辅助服务器或 SD 辅助服务器上执行压缩操作，因为 HDR 目标服务器必须具有与源服务器相同的数据和物理布局。

集群的数据库服务器配置需求

要使高可用性集群服务器对运行，必须完全配置每个数据库服务器。有关配置数据库服务器的信息，请参阅数据库服务器的安装和配置。然后您可以使用该配置的相关方面来配置

该数据库服务器对中另一数据库服务器。有关配置参数的更多信息，请参阅《GBase 8s 管理员参考》。

这些主题描述了以下有关集群数据库服务器对的配置注意事项：

数据库服务器版本

主数据库服务器和辅助数据库服务器上的数据库服务器版本必须相同。

存储空间和块的配置

在主数据库服务器和辅助数据库服务器上，数据库空间数、块数、它们的大小、路径名以及它们的偏移量必须相同。 此外，如果 HDR 辅助服务器用于创建活动报告，那么该配置必须至少包含一个临时数据库空间。请参阅用于排序的临时数据库空间和临时表的使用。

仅限 UNIX™：

必须按在 UNIX 上分配原始磁盘空间中所述，为块路径名使用符号链接。

重要： 如果不为块路径名使用符号链接，那么无法轻松更改块的路径名。有关更多信息，请参阅重命名块。

以下 ONCONFIG 参数在每个数据库服务器上都必须具有相同值：

- ROOTNAME
- ROOTOFFSET
- ROOTPATH
- ROOTSIZE

HDR 环境中的非缺省页大小

数据库空间的页大小和缓冲池规范自动从主数据库服务器向辅助数据库服务器传播。虽然主数据库服务器和辅助数据库服务器必须具有相同的缓冲池，但是缓冲池中的缓冲区数无需匹配。

镜像

无需将两个数据库服务器上的 MIRROR 参数设置为相同的值；您可以启用一个数据库服务器上的镜像，而禁用另一数据库服务器上的镜像。但如果 您为主数据库服务器的 Root 块指定镜像块，那么必须也为辅助数据库服务器上的 Root 块指定镜像块。因此，以下 ONCONFIG 参数必须在两个数据库服务器上设置为相同的值:

- MIRROROFFSET
- MIRRORPATH

物理日志配置

物理日志在两个数据库服务器上必须相同。以下 ONCONFIG 参数在每个数据库服务器上都必须具有相同值：

- PHYSBUFF
- PHYSFILE

数据库空间和逻辑日志磁带备份设备

您可以为主数据库服务器和辅助数据库服务器指定不同磁带设备。

如果您使用 ON-Bar，那么在两个数据库服务器上将 ON-Bar 配置参数设置为相同的值。有关 ON-Bar 参数的信息，请参阅《GBase 8s 备份与复原指南》。

如果使用 gtape，那么存储空间和逻辑日志备份设备的磁带大小和磁带块大小必须相同。以下 ONCONFIG 参数在每个数据库服务器上都必须具有相同值：

- LTAPEBLK
- LTAPESIZE
- TAPEBLK
- TAPESIZE

要用足磁带的物理容量，请将 LTAPESIZE 和 TAPESIZE 设置为 0。

逻辑日志配置

将所有日志记录复制到辅助服务器。 您必须为两个数据库服务器配置相同数目的逻辑日志文件和相同的逻辑日志大小。以下 ONCONFIG 参数在每个数据库服务器上都必须具有相同值：

- LOGBUFF
- LOGFILES
- LOGSIZE
- DYNAMIC_LOGS

数据库服务器记录逻辑日志文件的添加。在主服务器上动态添加的逻辑日志文件将在辅助服务器上自动复制。尽管辅助服务器上的 DYNAMIC_LOGS 值不起作用，请保持主服务器上 DYNAMIC_LOGS 与值的同步，以免它们切换角色。

HDR 配置参数

以下 HDR 配置参数在复制对中的两个数据库服务器上必须设置为相同的值：

- DRAUTO
- DRINTERVAL
- DRTIMEOUT

对于高可用性集群中的 HDR、RSS 和 SDS 辅助服务器，必须通过将 TEMPTAB_NOLOG 配置参数设置为 1 来始终禁用对临时表的逻辑日志记录。

集群事务协调

可以设置 CLUSTER_TXN_SCOPE 配置参数或运行 SET ENVIRONMENT CLUSTER_TXN_SCOPE 语句来要求客户机应用程序运行的操作在集群中完全传播或在辅助服务器上应用之前不接收任何落实。

如果客户机应用程序执行了多步骤操作（如装入或更新数据），然后尝试处理其他服务器会话中或高可用性集群中其他节点上的数据，那么该操作可能会因异步日志处理而失败。

装入或更新操作必须在集群数据库节点上应用，然后才能在该节点上进行处理。**集群事务**协调将强制操作等待在集群范围或在辅助服务器上的应用完成，然后才返回落实。集群事务协调可确保多步骤过程中的各步骤按顺序完成。

DELAY_APPLY 或 STOP_APPLY 配置参数值不为 0 的远程独立辅助 (RSS) 服务器无法参与集群事务协调。

为集群配置安全连接

要使高可用性集群能够运行，数据库服务器相互之间必须建立可信连接。通过使用 sqlhosts 文件中的连接安全性选项并创建 hosts.equiv 文件，可以保护集群服务器之间的连接。通过使用 sqlhosts 文件中的连接安全性选项并使用 REMOTE_SERVER_CFG 配置参数指定的一个或多个 hosts.equiv 文件，可以保护集群服务器之间的连接。

sqlhosts 文件中列出的安全端口只能用于集群通信。客户机应用程序不能连接到安全端口。

要配置可信环境用于复制，请执行以下操作：

1. 在集群中每台服务器上编辑 sqlhosts 文件：

   a. 为该计算机上运行的数据库服务器添加一个条目，并包含 s=6 选项。

   b. 为集群中的其他所有服务器添加条目。不要在这些条目中包含 s=6 选项。

2. 设置 sqlhosts 文件或注册表的 nettype 字段，并将 NETTYPE 配置参数设置为网络协议（例如，ontlitcp 或 onsoctcp），以便两个不同计算机上的数据库服务器可以相互通信。不能使用非网络协议，例如 **onipcshm**、**onipcstr** 或 **onipcnmp**。

3. 在 $GBASEDBTDIR/etc 目录中创建 hosts.equiv 文件，并包含其他集群服务器的主机名，每个主机名单独占用一行。

   文件的所有者必须是属于组 **gbasedbt** 的用户 **gbasedbt**，并且必须限制许可权，这样只有用户 **gbasedbt** 可以修改该文件（使用 octal 许可权时，允许值为 644、640、444 或 440 中的一个）。

4. 通过以下某种方法设置可信主机信息

   - 在 $GBASEDBTDIR/etc 目录中创建 hosts.equiv 文件，并包含其他集群服务器的主机名，每个主机名单独占用一行。文件的所有者必须是属于组 gbasedbt 的用户 gbasedbt，并且必须限制许可权，这样只有用户 gbasedbt 可以修改该文件（使用 octal 许可权时，允许值为 644、640、444 或 440 中的一个）。
   - 将 S6_USE_REMOTE_SERVER_CFG 配置参数设置为 1，然后将 REMOTE_SERVER_CFG 配置参数设置为包含其他集群服务器的主机名的文件。

5. 创建服务器别名，用于运行实用程序和客户机应用程序。

   例如，将 GBASEDBTSERVER 环境变量设置为该别名，以便运行实用程序（例如，gstat 和 gtape）和客户机应用程序（例如，DB-Access）。

6.2.3 首次启动 HDR

当您完成了 HDR 的配置后，您将准备启动 HDR。本主题描述了启动 HDR 的必需步骤。

您希望在两个数据库服务器（ServerA 和 ServerB）上启动 HDR。启动 HDR 的过程（将 ServerA 用作主数据库服务器，将 ServerB 用作辅助数据库服务器）将在以下步骤中进行描述。下表列出了执行每个步骤所需的命令以及发送到消息日志的消息。您可以使用 gtape 或 ON-Bar 执行备份与复原。您必须在整个过程中使用相同的实用程序。

重要：即使使用 ON-Bar 执行备份与复原，两个数据库服务器上仍然需要 gtape 实用程序来执行备份和应用逻辑日志。不要从参与 HDR 集群环境的数据库服务器除去 gtape 实用程序。

还可以使用外部备份与复原来设置 HDR。 请参阅《GBase 8s 备份与复原指南》，以获取有关如何执行外部备份与复原的信息。请参阅使用 gtape STDIO 功能来减少设置时间，以获取直接从 HDR 主服务器设置 HDR 辅助服务器的最快方法。

要启动 HDR，请执行以下操作：

1. 在两个数据库服务器上均安装用户定义的类型和用户定义的例程然后仅在 ServerA 上注册它们。
2. 创建 ServerA 的 0 级备份。
3. 使用 gadmin -d 命令将 ServerA 的类型设置为主类型并指示相关联的辅助数据库服务器的名称（在本例中为 ServerB）。

   当您发出 gadmin -d 命令时，数据库服务器将尝试与 HDR 对中的另一数据库服务器建立连接并启动 HDR 操作。仅当复制对中另一数据库服务器已经设置为正确类型时，建立连接的尝试才会成功。

   此时，ServerB 未联机且未设置为类型辅助，因此将不会建立 HDR 连接。
4. 从您在步骤 1 中创建的 0 级备份执行对 ServerB 的物理复原。不要执行逻辑复原。

   如果您使用的是：

   - ON-Bar，请使用 gbackuprestore -r -p 命令来执行物理复原。
   - ON-Bar 并正在执行外部复原，请使用 gbackuprestore -r -p -e 命令来执行物理复原。
   - gtape，使用 gtape -p 选项。 您无法使用 gtape -r 选项，因为它同时执行物理复原和逻辑复原。

     注：如果主服务器和辅助服务器位于两台不同的机器上，那么必须将主服务器的物理复原放在辅助服务器上。物理复原的位置将由 onconfig 参数 TAPE 定义。 必须先在辅助服务器上设置 IFX_ONTAPE_FILE_PREFIX 变量，然后才能运行 gtape -p。
   - gtape 并正在执行外部复原，请使用 gtape -p -e 命令来执行物理复原。
5. 使用 gadmin -d 命令将 ServerB 的类型设置为辅助类型并指示相关联的主数据库服务器。

ServerB 尝试与主数据库服务器 (ServerA) 建立 HDR 连接，并启动操作。必须成功建立连接。

在 HDR 开始前，辅助数据库服务器使用自步骤 2 以来写入主数据库服务器的逻辑日志记录来执行逻辑恢复。如果所有这些逻辑日志记录仍然位于主数据库服务器磁盘上，那么主数据库服务器将通过网络直接将这些记录发送到辅助数据库服务器，并且自动执行逻辑恢复。

如果您已备份并释放主数据库服务器上的逻辑日志文件，那么这些文件中的记录不再在磁盘上。 辅助数据库服务器会提示您从磁带恢复这些文件。在这种情况下，您必须执行步骤 6。

重要： 在同一会话期间，必须完成步骤 4 到 5。如果必须在步骤 4 后关闭并重新启动辅助数据库服务器，那么必须重新执行步骤 4。

6. 如果写入主数据库服务器的逻辑日志记录不再在主磁盘上，那么辅助数据库服务器会提示您从磁带备份恢复这些文件。

   如果辅助数据库服务器必须通过网络读取备份逻辑日志文件，那么将辅助数据库服务器上的磁带设备参数设置为正在运行主数据库服务器的计算机上的设备或设置为与主数据库服务器处于相同位置的设备。

   在您恢复磁带上的所有逻辑日志文件后，逻辑复原将完成使用主数据库服务器磁盘上的逻辑日志文件。

下表说明了前面的步骤，这样您可以清楚地确定在主服务器上执行的步骤和在辅助服务器上执行的步骤。该表还显示在执行每个步骤之后写入日志文件的信息。

表 1. 首次启动 HDR 的步骤

步骤	在主服务器 (ServerA) 上	在辅助服务器 (ServerB) 上
1.	安装 UDR、UDT 注册 UDR、UDT模块。	安装 UDR、UDT
2.	gtape 命令： 运行 gtape -s -L 0 ON-Bar 命令 ：运行 gbackuprestore -b -L 0 发送至消息日志的消息： Level 0 archive started on	

步骤	在主服务器（ServerA）上	在辅助服务器（ServerB）上
	rootdbs Archive on rootdbs completed	
3.	gadmin 命令 gadmin -d primary sec_name 发送至消息日志的消息： DR: new type = primary server name = sec_name DR: Trying to connect to secondary server DR: Cannot connect to secondary server	
4.		gtape 命令 运行 gtape -p 或 gtape -p -e 当提示您备份日志时回答否。 ON-Bar 命令 运行 gbackuprestore -r -p 或 gbackuprestore -r -p -e 发送至消息日志的消息： GBase 8s Database Server Initialized -- Shared Memory Initialized Recovery Mode Physical restore of rootdbs started

步骤	在主服务器（ServerA）上	在辅助服务器（ServerB）上
		Physical restore of rootdbs completed
5.		运行 gadmin -d secondary prim_name 发送至消息日志的消息： DR: new type = secondary, primary server name = prim_name 如果自步骤 1 以来写入主数据库服务器的所有逻辑日志记录仍位于主数据库服务器磁盘上，那么辅助数据库服务器将读取这些记录以执行逻辑恢复。（否则，必须执行步骤 6）。
	发送至消息日志的消息： DR: Primary server connected DR: Primary server operational	发送至消息日志的消息： DR: Trying to connect to primary server DR: Secondary server connected DR: Failure recovery from disk in progressn recovery worker threads will be started Logical Recovery Started Start Logical Recovery - Start Log n, End Log ? Starting Log Position - n 0xnnnnn DR: Secondary server operational
6.		gtape 命令 gtape -l

步骤	在主服务器（ServerA）上	在辅助服务器（ServerB）上
		ON-Bar 命令 gbackuprestore -r -l
	发送至消息日志的消息： DR: Primary server connected DR: Primary server operational	发送至消息日志的消息： DR: Secondary server connected DR: Failure recovery from disk in progressn recovery worker threads will be started Logical Recovery Started Start Logical Recovery - Start Log n, End Log ? Starting Log Position - n 0xnnnnn DR: Secondary server operational

使用 gtape STDIO 功能来减少设置时间

通过使用 gtape STDIO 功能，可显著提高设置 HDR 的速度。使用此功能，gtape 会在备份期间将数据写入 shell 的标准输出，然后在复原期间从标准输入中读取该数据。 使用远程命令解释器（如 rsh 或 ssh）将 STDIO 备份与同时发生的 STDIO 复原组合在一个管道中，以允许使用一个命令行来执行 HDR（或 RSS）辅助服务器的初始设置。由于不对磁带或磁盘执行读写操作，所以可以节省存储空间，而且无需等待备份完成即可开始复原。

请参阅《GBase 8s 备份与复原指南》以获取有关使用 STDIO 值的详细信息。

不管使用哪种备份实用程序（gtape 或 ON-Bar），都可以使用这种使用 gtape 设置 HDR 的方法。

重要：当以这种方法使用 STDIO 时，不会在任何位置保存可用于执行复原的永久备份。如果在源（备份）端使用 -F（假）选项，那么不会将备份记录在数据库服务器的保留页中。 此外，还会禁止任何交互式对话，并且不会显示任何提示或询问。您还必须确保管道的远程端为远程 GBase 8s 实例挑选合适的环境。 除了备份数据外，脚本不应该产生任何输出，因为备份数据由复原过程读取（例如，不启用跟踪）。

下表中的步骤必须由用户 gbasedbt 执行，脚本必须是可执行的，如果不是用完整路径调用，那么脚本必须位于主目录中。如果您需要在网络上传输保密数据，那么可以使用 ssh 来代替 rsh。

表 1. 使用 rsh 从主服务器设置 HDR 的备用方法

步骤	在主服务器上	在辅助服务器上
1.		安装 UDR、UDT。
2.	安装 UDR、UDT。	
3.	注册 UDR、UDT 和模块。	
4.	运行 gadmin -d primary sec_name	
5	运行 gtape -s -L 0 -t STDIO -F | rsh sec_nameontape_HDR_restore.ksh	
6.		运行 gadmin -d secondary pri_name

在上表中，辅助服务器上的脚本 gtape_HDR_restore.ksh 必须包含以下命令：

```
#!/bin/ksh
# first get the proper Gbasedbt environment set
. hdr_sec.env
# redirecting stdout and stderr required since otherwise command might never return
gtape -p -t STDIO > /dev/null 2>&1
```

下列步骤显示如何从辅助服务器设置 HDR。

表 2. 使用 rsh 从辅助服务器设置 HDR 的备用方法：

步骤	在主服务器上	在辅助服务器上
1.		安装 UDR、UDT。
2.	安装 UDR、UDT。	
3.	注册 UDR、UDT。	
4.	运行 gadmin -d primary sec_name	
5.		运行 rsh pri_name gtape_HDR_backup.ksh | gtape -p -t STDIO
6.		运行 gadmin -d secondary pri_name

在上表中，主服务器上的脚本 gtape_HDR_backup.ksh 必须包含以下命令：

```
#!/bin/ksh
# first get the proper GBase 8s environment set
. hdr_pri.env
gtape -s -L 0 -F -t STDIO
```

6.2.4 远程独立辅助服务器

这些主题概述了在高可用性环境中设置和配置远程独立 (RS) 辅助服务器。

比较 RS 辅助服务器和 HDR 辅助服务器

RS 辅助服务器在很多方面都与 HDR 辅助服务器相似。将日志发送到 RS 辅助服务器的方式与主服务器将日志发送到 HDR 辅助服务器的方式很相似。但是，RS 辅助服务器用于完全在异步通信框架内运行，因此对主服务器的影响达到最小。事务落实和检查点在主服务器和 RS 辅助服务器之间均未同步。不保证在主服务器上落实的任何事务也在同一时间在 RS 辅助服务器上得到落实。

在高可用性集群中，HDR 辅助服务器的日志必须优先于任何 RS 辅助服务器的日志。如果 HDR 辅助服务器脱机，主服务器将继续把日志发送到 RS 辅助服务器。但是，当 HDR 辅助服务器恢复联机时，GBase 8s 会停止向 RS 辅助服务器发送日志，并优先将日志发送到 HDR 辅助服务器，使其日志重放优先于 RS 辅助服务器。因为 HDR 辅助服务器是集群中的第一个故障转移选项，所以需要优先使用 HDR 辅助服务器日志。如果故障转移发生时 RS 辅助服务器日志优于于 HDR 辅助服务器日志，那么 RS 辅助服务器无法与新的主服务器同步。

尽管 RS 辅助服务器与 HDR 辅助服务器类似，但有某些操作是 HDR 辅助服务器支持的而 RS 辅助服务器不支持的：

- SYNC 方式
- DRAUTO 参数
- 同步检查点

对于高可用性集群中的 HDR、RSS 和 SDS 辅助服务器，必须通过将 TEMPTAB_NOLOG 配置参数设置为 1 来始终禁用对临时表的逻辑日志记录。

为 RS 或 SD 辅助服务器指定别名

利用 HA_ALIAS 配置参数将 RS 辅助服务器或 SD 辅助服务器名称指定给高可用性集群配置。

以下示例显示为辅助服务器指定别名的 RS 辅助服务器文件的 ONCONFIG 配置中的条目。

```
DBSERVERNAME            reports_srvr
```

```
HA_ALIAS                failover_srvr
```

当辅助服务器与主服务器连接时，会将别名发送给主服务器。 连接管理器仲裁器可以使用该别名故障转移至辅助服务器。 如果未指定 HA_ALIAS，就使用 DBSERVERNAME。 请参阅HA_ALIAS 配置参数中设置的连接信息以获取更多信息。

索引页日志记录

要使用 RS 辅助服务器，必须启用索引页日志记录。

索引页日志记录的工作原理

在创建索引时，索引页日志记录将各页写入到逻辑日志，以使高可用性环境中各服务器之间的索引创建同步。

索引页日志记录将完整索引写入到日志文件，然后将该日志文件异步地传输到辅助服务器。辅助服务器可以是 RS 辅助服务器，也可以是 HDR 辅助服务器。 然后，将日志文件事务读入辅助服务器上的数据库。在恢复期间，辅助服务器无需重新构建索引。对于 RS 辅助服务器，主服务器不会等待来自该辅助服务器的确认，这允许立即访问主服务器上的索引。

使用 ONCONFIG 参数 LOG_INDEX_BUILDS 控制索引页日志记录。将 LOG_INDEX_BUILDS 参数设置为 1（已启用），可在主服务器上构建索引，然后将其发送到辅助服务器。

启用或禁用索引页日志记录

使用 LOG_INDEX_BUILDS 配置参数在数据库服务器启动时启用或禁用索引页日志记录。通过运行 gadmin -wf LOG_INDEX_BUILDS=1（启用）或 0（禁用）可更改 onconfig 文件中 LOG_INDEX_BUILDS 的值。

当 RS 辅助服务器存在于高可用性环境中时，必须启用索引页日志记录。

查看索引页日志记录统计信息

可以使用 gstat 实用程序或系统监视接口 (SMI) 表来查看是启用还是禁用了索引页日志记录。统计信息也显示启用或禁用索引页日志记录的日期和时间。

要查看索引页日志记录统计信息，请使用 gstat -g ipl 命令，或查询 **sysipl** 表。

有关 gstat -g ipl 输出的示例，请参阅《GBase 8s 管理员参考》中有关 gstat 实用程序的信息。

服务器多路复用器组 (SMX) 连接

服务器多路复用器组 (SMX) 是支持高可用性环境中各服务器之间的加密多路复用网络连接的通信接口。SMX 提供数据库服务器实例之间的可靠、安全和高性能通信机制。

启用 SMX 加密

使用 ENCRYPT_SMX 配置参数设置高可用性配置的加密级别。如果将 ENCRYPT_SMX 参数设置为 1，那么仅当连接的数据库服务器也支持加密时才对 SMX 事务使用加密。如

果将 ENCRYPT_SMX 配置参数设置为 2，那么仅允许到加密数据库服务器的连接。将 ENCRYPT_SMX 设置为 0 将禁用服务器之间的加密。

获取 SMX 统计信息

可以使用 gstat 实用程序或系统监视接口 (SMI) 表来查看 SMX 连接统计信息或 SMX 会话统计信息。

要查看 SMX 连接统计信息，请使用 gstat -g smx 命令。

要查看 SMX 会话统计信息，请使用 gstat -g smx ses 命令。

有关 gstat -g smx 和 gstat -g smx ses 输出的示例，请参阅《GBase 8s 管理员参考》中有关 gstat 实用程序的信息。

首次启动 RS 辅助服务器

在完成 RS 辅助服务器的硬件配置后，就可以随时启动 RS 辅助服务器，并将其连接到主服务器。

假设您想要启动主服务器和 RS 辅助服务器（ServerA 和 ServerB）。在以下步骤中描述了启动服务器的过程（使用 ServerA 作为主数据库服务器并使用 ServerB 作为 RS 辅助数据库服务器）。上表列出了执行每一步骤所需的命令。

该过程要求对主服务器进行备份，然后将备份复原到辅助服务器上。您可以使用 gtape 或 ON-Bar 执行备份与复原。您必须在整个过程中使用相同的实用程序。

重要：即使使用 ON-Bar 执行备份与复原，两个数据库服务器上仍然需要 gtape 实用程序来执行备份和应用逻辑日志。不要从参与 HDR 集群环境的数据库服务器除去 gtape 实用程序。

您还可以使用用于外部备份与复原的标准 ON-Bar 或 gtape 命令来设置 RS 辅助服务器。

要启动带有 RS 辅助服务器的主服务器，请执行以下操作：

1. 在两个数据库服务器上均安装用户定义的类型和用户定义的例程，然后仅在 ServerA 上注册它们。

   有关如何安装用户定义的类型或用户定义的例程的信息，请参阅《GBase 8s 用户定义的例程与数据类型开发者指南》。

2. 激活主服务器上的索引页日志记录。
3. 记录主服务器上 RS 辅助服务器的身份。 首次建立主服务器和 RS 辅助服务器之间的连接时，可选密码在这两者之间提供认证。
4. 创建 ServerA 的 0 级备份。
5. 从您在步骤 4 中创建的 0 级备份执行对 ServerB 的物理复原。不要执行逻辑复原。

使用适当的命令：

使用 gbackuprestore -r -p 命令来执行物理复原。

使用 gbackuprestore -r -p -e 命令来执行物理外部复原。

使用 gtape -p 选项。（不要使用 gtape -r 选项，因为它同时执行物理复原和逻辑复原。）

使用 gtape -p -e 命令来执行物理外部复原。

6. 使用 gadmin -d RSS ServerA password 命令可将 ServerB 的类型设置为 RS 辅助服务器，并指示相关联的主数据库服务器。

   ServerB 尝试建立与主数据库服务器 (ServerA) 的连接，并启动操作。必须成功建立连接。

   辅助数据库服务器使用自步骤 4 以来写入主数据库服务器的逻辑日志记录来执行逻辑恢复。如果所有这些逻辑日志记录仍然位于主数据库服务器磁盘上，那么主数据库服务器将通过网络将这些记录直接发送到 RS 辅助服务器，然后自动执行逻辑恢复。

7. 如果您已备份并释放主数据库服务器上的逻辑日志文件，那么这些文件中的记录不再在磁盘上。 辅助数据库服务器会提示您从磁带恢复这些文件。在这种情况下，您必须执行步骤 7。

   重要： 在同一会话期间，必须完成步骤 5 到 6。如果必须在步骤 5 后关闭并重新启动辅助数据库服务器，那么必须重新执行步骤 5。

   如果写入主数据库服务器的逻辑日志记录不再在主磁盘上，那么辅助数据库服务器会提示您从磁带备份恢复这些文件。

   如果辅助数据库服务器必须通过网络读取备份逻辑日志文件，那么将辅助数据库服务器上的磁带设备参数设置为正在运行主数据库服务器的计算机上的设备或设置为与主数据库服务器处于相同位置的设备。

   在您恢复磁带上的所有逻辑日志文件后，逻辑复原将完成使用主数据库服务器磁盘上的逻辑日志文件。

表 1. 首次启动带有 RS 辅助服务器的主服务器的步骤

步骤	在主服务器上	在 RS 辅助服务器上
1.	安装 UDR、UDT。 注册 UDR、UDT模块。	安装 UDR、UDT模块。
2.	gadmin 命令 gadmin -wf LOG_INDEX_BUILDS=1	
3.	gadmin 命令 gadmin -d add	

步骤	在主服务器上	在 RS 辅助服务器上
	RSS rss_servername password	
4.	gtape 命令 gtape -s -L 0 ON-Bar 命令 gbackuprestore -b -L 0	
5.		gtape 命令 gtape -p 或 gtape -p -e 当提示您备份日志时回答否。 ON-Bar 命令 gbackuprestore -r -p 或 gbackuprestore -r -p -e
6.		gadmin 命令 gadmin -d RSS primary_servername password 如果自步骤 1 以来写入主数据库服务器的所有逻辑日志记录仍位于主数据库服务器磁盘上，那么辅助数据库服务器将读取这些记录以执行逻辑恢复。（否则，必须执行步骤 8）。
7.		gtape 命令 gtape -l

步骤	在主服务器上	在 RS 辅助服务器上
		ON-Bar 命令 gbackuprestore -r -l 仅当辅助数据库服务器提示您从磁带备份恢复逻辑日志文件时，才需要执行此步骤。

通过备用备份方法来减少设置时间

通过使用 gtape STDIO 功能，可显著提升设置辅助服务器的速度。请参阅使用 gtape STDIO 功能来减少设置时间以获取更多信息。

请参阅《GBase 8s 备份与复原指南》以获取有关使用 STDIO 值的详细信息。

将脱机主服务器转换为 RS 辅助服务器

在执行主服务器到 RS 辅助服务器的计划或意外故障转移后，可以将旧的主服务器转换为 RS 辅助服务器。

例如，假设您有一个名为 srv1 的主服务器，此服务器已故障转移到名为 srv2 的 RS 辅助服务器。 以下步骤显示了如何将旧的主服务器转换为 RS 辅助服务器。

1. 在新的主服务器 (srv2) 上，将旧的主服务器 (srv1) 注册为 RS 辅助服务器。

```
gadmin -d add RSS srv1
```

2. 如果要将旧的主服务器转换为 RS 辅助服务器，但该服务器处于脱机状态，那么请使用下面所示的备份和复原命令初始化该服务器：。或者，也可以通过运行以下命令来初始化旧的主服务器：

```
oninit -PHY
```

   请参阅《GBase 8s 管理员参考》中的 oninit 实用程序以获取更多信息。

3. 使用以下命令将服务器转换为 RS 辅助服务器：

```
gadmin -d RSS srv2
```

延迟应用日志记录

要在灾难恢复场景中进行协助，可以将 RS 辅助服务器配置为先等待指定的时间段，然后应用从主服务器接收的日志。

通过延迟日志文件的应用，可从 RS 辅助服务器复原数据库来迅速地从错误的数据库修改中恢复。也可在指定时间停止 RS 辅助服务器上的日志应用。

例如，假设数据库管理员希望根据行的存在时间从表中删除特定行。表中的每行包含一个时间戳记，指示该行的创建时间。如果数据库管理员不慎将过滤器设置为错误日期，删除

的行可能超出意愿。通过延迟日志文件的应用，这些行仍将保留在 RS 辅助服务器中。 然后，数据库管理员可以从辅助服务器抽取这些行，并将其插入到主服务器。

现在假设需要数据库管理员通过重命名表来对模式执行更改，但是数据库管理员输入了错误的命令并删除了表 orders，而不是将表名更改为 store_orders。 如果配置了 RS 辅助服务器以延迟日志的应用，那么数据库管理员可从辅助服务器恢复 orders 表。

配置了日志文件延迟应用时，达到指定时间段之前，将不应用从主服务器发送的事务。从主服务器接收的日志文件将在 RS 辅助服务器上的指定安全目录中登台，然后在指定时间段之后应用。延迟应用日志文件的方法有两种：

- 在指定时间间隔之后应用登台的日志文件
- 在指定时间停止应用日志文件

可通过在 RS 辅助服务器的 onconfig 文件中设置配置参数来启用日志文件的延迟应用。启用日志文件延迟应用之前，必须通过设置 LOG_STAGING_DIR 配置参数来指定日志文件的登台目录。指定 LOG_STAGING_DIR 配置参数之后，可通过编辑 onconfig 文件或动态使用 gadmin -wf 命令来配置 DELAY_APPLY 或 STOP_APPLY 配置参数。

日志记录的存储位置

服务器将在 LOG_STAGING_DIR 指定的目录内额外创建名为 ifmxlog_## 的目录，其中 ## 是 SERVERNUM 指定的实例。这些目录用于存储逻辑日志，也在恢复 RS 辅助服务器期间使用。如果必须恢复 RS 辅助服务器，而主服务器上已经包装了日志，那么可使用 ifmxlog_## 中的日志来恢复该服务器。ifmxlog_## 内的文件在不再需要时将清除。

延迟触发的条件

BEGIN WORK、COMMIT WORK 和 ROLLBACK WORK 日志记录中的时间值用于计算延迟或停止应用日志文件的时间。将日志页传递到恢复进程之前，将计算这些时间值。

如果发出 BEGIN WORK 语句，事务执行第一个更新活动之前，将不写入 BEGIN WORK 日志记录；因此，发出 BEGIN WORK 语句的时间与写入 BEGIN WORK 日志的时间之间可能存在延迟。

与辅助服务器更新交互

必须注意辅助服务器更新与日志文件延迟应用之间的交互。如果启用了更新，并且更新了辅助服务器，那么达到 DELAY_APPLY 指定的时间量之前，将不应用更新。但是，禁用辅助服务器更新也将禁用“已落实读取”，从而保证检索行时，在表中落实检索到的每个行。

要保留“已落实读取”隔离级别，请考虑使用 UPDATABLE_SECONDARY 配置参数启用辅助服务器更新，但是从连接管理器服务级别协议列表中除去用于延迟应用日志文件的 RS 辅助服务器。也可以考虑将 RS 辅助服务器移动到新 SLA。

有关更多信息，请参阅辅助服务器上的数据库更新和《GBase 8s 管理员参考》。

指定日志登台目录

配置日志登台目录，以指定 RS 辅助服务器上的日志文件在应用于数据库之前，在何处登台。

启用日志文件延迟应用之前，必须为从主服务器发送的日志文件指定登台目录。未定义任何缺省登台目录。服务器将在 LOG_STAGING_DIR 指定的目录内额外创建名为 ifmxlog_## 的目录，其中 ## 是 SERVERNUM 指定的实例。这些目录用于存储逻辑日志，也在恢复 RS 辅助服务器期间使用。登台日志文件在不再需要时将自动除去。如果丢失了 LOG_STAGING_DIR 内的文件，而主服务器覆盖了日志，那么必须重建 RS 辅助服务器。

必须确保 LOG_STAGING_DIR 指定的目录存在，且该目录是安全目录。该目录的所有者必须是用户 gbasedbt，必须属于组 gbasedbt，并且不得具有公共读、写或执行许可权。如果启用了角色隔离，那么 LOG_STAGING_DIR 指定的目录必须属于拥有 $GBASEDBTDIR/etc 的用户或组。如果 LOG_STAGING_DIR 指定的目录不是安全目录，那么不能初始化该服务器。 如果该目录不是安全目录，将在联机消息日志中写入以下消息：

```
日志登台目录 (directory_name) 不安全。
```

还必须确保磁盘包含的空间足以容纳主服务器中的所有日志，并且目录中不包含来自不再使用的早期实例的登台日志。

有关更多信息，请参阅《GBase 8s 管理员参考》。

要设置 LOG_STAGING_DIR，请执行以下操作：

1. 确保要用于存储日志的目录存在，且该目录是安全目录。
2. 编辑 RS 辅助服务器的 onconfig 文件。
3. 按照下面的方式指定登台目录：LOG_STAGING_DIR directory_name，其中，directory_name 是要用于存储日志的目录的名称。
4. 重新启动服务器。

也可通过使用 gadmin -wf 命令，在不重新启动服务器的情况下设置 LOG_STAGING_DIR 配置参数；但是，运行该命令时，日志文件的延迟应用不得处于活动状态。

延迟应用 RS 辅助服务器上的日志记录

可在 RS 辅助服务器上延迟应用日志记录，以准备灾难恢复方案。

可通过设置 DELAY_APPLY 配置参数来启用日志文件的延迟应用。可手动编辑 onconfig 文件并重新启动服务器，也可使用 gadmin -wf 命令动态更改该值。设置 DELAY_APPLY 的值时，还必须设置 LOG_STAGING_DIR。如果配置了 DELAY_APPLY，但未将 LOG_STAGING_DIR 设置为有效的安全目录，那么就不能初始化服务器。

请同时使用数目和修饰符来设置 DELAY_APPLY。数目最多可包含 3 个数字，用于指示修饰符单位的数量。 修饰符是以下项之一：

- D（或 d），代表天
- H（或 h），代表小时

- M（或 m），代表分钟
- S（或 s），代表秒

有关更多信息，请参阅《GBase 8s 管理员参考》。

要在 RS 辅助服务器上延迟应用日志文件 4 个小时，请运行以下命令：

```
gadmin -wf DELAY_APPLY=4H
```

要延迟应用日志文件 1 天，请运行以下命令：

```
gadmin -wf DELAY_APPLY=1D
```

要禁用日志文件的延迟应用，请运行以下命令：

```
gadmin -wf DELAY_APPLY=0
```

停止应用日志记录

可在 RS 辅助服务器上停止应用日志记录，以准备灾难恢复方案。

可通过设置 STOP_APPLY 配置参数来停止在 RS 辅助服务器上应用日志文件。可手动编辑 onconfig 文件并重新启动服务器，也可使用 gadmin -wf 命令动态更改该值。设置 STOP_APPLY 的值时，还必须设置 LOG_STAGING_DIR。如果配置了 STOP_APPLY，但未将 LOG_STAGING_DIR 设置为有效的安全目录，那么就不能初始化服务器。

有关更多信息，请参阅《GBase 8s 管理员参考》。

要立即在 RS 辅助服务器上停止应用日志文件，请运行以下命令：

```
gadmin -wf STOP_APPLY=1
```

要在 2009 年 4 月 15 日晚上 11 点停止应用日志文件，请运行以下命令：

```
gadmin -wf STOP_APPLY="2009:04:15-23:00:00"
```

恢复正常应用日志文件

```
gadmin -wf STOP_APPLY=0
```

6.2.5 共享磁盘辅助服务器

这些主题概述了在高可用性环境中设置和配置 SD（共享磁盘）辅助服务器。SD 辅助服务器选项可用于标准版 GBase 8s。

SD 辅助服务器

共享磁盘 (SD) 辅助服务器参与了高可用性集群配置。 在这样的配置中，主服务器和 SD 辅助服务器共享同一个磁盘或磁盘阵列。

SD 辅助服务器不在自己的磁盘空间上保存物理数据库副本。而是与主服务器共享磁盘。

必须将 SD 辅助服务器配置为访问允许并行访问的共享磁盘设备。不要配置使用操作系统缓冲（如 NFS 交叉安装的文件系统）的 SD 辅助服务器。如果 SD 辅助服务器实例和主服务器实例位于一台机器上，那么这两台服务器都可访问本地磁盘。如果 SD 辅助服务器和主服务器位于不同的物理机器上，那么必须这两台服务器必须都配置为访问显示为本地连接的共享磁盘设备，如 Veritas 或 GPFS™。

SD 辅助服务器可与 HDR 辅助服务器、RS 辅助服务器和 Enterprise Replication 结合使用。

可以非常快速地将 SD 辅助服务器添加到高可用性环境，因为它们不需要单独的磁盘副本。因为 SD 服务器共享主服务器的磁盘存储资源，所以建议您提供一些其他磁盘备份方法，例如磁盘镜像，或者使用 RS 辅助服务器或 HDR 辅助服务器。

以下限制影响属于共享磁盘辅助服务器的数据库服务器实例的提升：

- SD 辅助服务器不能提升为 RS 辅助服务器。
- SD 辅助服务器不能提升为可存在于主要高可用性环境之外的标准服务器。

SD 辅助服务器的磁盘需求

除了磁盘需求（与主服务器共享）之外，硬件和软件需求通常与 HDR 辅助服务器的需求相同（请参阅针对特定受支持平台的机器说明®）。 此外，具有数据库服务器的计算机之间必须共享主磁盘系统。这表示从 SD 辅助服务器到数据库空间的路径必须与主服务器的数据库空间路径相同。请参阅配置集群。

设置共享磁盘辅助服务器

设置共享磁盘辅助服务器要先设置 SDS_ENABLE 配置参数。 接下来，将使用 gadmin 实用程序来设置 SD 辅助服务器用于连接到主服务器的主服务器别名。然后，修改 SD 辅助服务器上的配置文件以包含相应选项。最后，运行 oninit 实用程序以启动 SD 辅助服务器。

1. 在主服务器上，设置 onconfig 文件中的 SDS_TIMEOUT 配置参数：

   ```
   SDS_TIMEOUT x
   ```

   SDS_TIMEOUT 指定主服务器将等待从 SD 辅助服务器发送的日志位置确认的时间量（以秒为单位）。有关 SDS_TIMEOUT 配置参数的信息，请参阅《GBase 8s 管理员参考》。

2. 在主服务器上，配置 SD 主服务器的别名：

   ```
   gadmin -d set SDS primary <alias>
   ```

   由 <alias> 指定的服务器名称将成为共享磁盘环境的主服务器和 SD 辅助服务器的日志源。

3. 在 SD 辅助服务器上，设置配置文件中的以下配置参数：

   ```
   SDS_ENABLE 1
   SDS_PAGING <path 1>,<path 2>
   SDS_TEMPDBS <dbsname>,<dbspath>,<pagesize>,<offset>,<size>
   ```

SDS_ENABLE 在辅助服务器上必须设置为 1（启用）以便支持对共享磁盘环境。SDS_PAGING 指定两个文件的路径，这两个文件用于保存检查点之间可能需要清空的页。 每个文件都作为任意页大小块的临时磁盘存储器。 SDS_TEMPDBS 用于定义 SD 辅助服务器使用的临时数据库空间。 此数据库空间在启动服务器时会动态创建（不是通过运行 gspaces 而创建）。 请参阅《GBase 8s 管理员参考》以获取有关这些参数的其他信息。

4. 在 SD 辅助服务器上，设置以下配置参数以与主服务器上的那些参数匹配：

- ROOTNAME
- ROOTPATH
- ROOTOFFSET
- ROOTSIZE
- PHYSFILE
- LOGFILES
- LOGSIZE

映射其他配置参数以使主服务器的那些参数与 DBSERVERALIASES、DBSERVERNAME 和 SERVERNUM 的异常匹配。

对于高可用性集群中的 HDR、RSS 和 SDS 辅助服务器，必须通过将 TEMPTAB_NOLOG 配置参数设置为 1 来始终禁用对临时表的逻辑日志记录。

5. 在 SD 辅助服务器上，如果要启用客户机应用程序以在辅助服务器上执行更新，插入和删除操作，可以选择将 UPDATABLE_SECONDARY 配置参数设置为正整数。

6. 向主服务器的 sqlhosts 文件添加条目：

```
primary_dbservername  nettype  primary_hostname  servicename
```

7. 使用 oninit 命令启动 SD 辅助服务器。

在主服务器处于活动状态之后才能启动 SD 辅助服务器。

辅助服务器启动时，必须首先使用快速恢复方式处理任何已打开的事务。仅当启动检查点时打开的所有事务均落实或回滚之后，客户机应用程序才可连接到服务器。处理了打开的事务之后，客户机应用程序可正常连接到服务器。必须检查辅助服务器上的 online.log 文件，以验证是否已完成处理打开的事务。

下表说明了前面的步骤，这样您可以清楚地确定在主服务器上执行的步骤和在辅助服务器上执行的步骤。

表 1. 首次启动 SD 辅助服务器的步骤

步骤	在主服务器上	在辅助服务器上

步骤	在主服务器上	在辅助服务器上
1.	设置 onconfig 文件中的 SDS_TIMEOUT 配置参数： SDS_TIMEOUT x	
2.	配置 SD 主服务器的别名： gadmin -d set SDS primary <alias>	
3		设置配置参数： SDS_ENABLE 1 SDS_PAGING <path 1>,<path 2> SDS_TEMPDBS <dbsname>,<dbspath>, <pagesize>,<offset>, <size>
4		设置配置参数以便与主服务器上的那些参数匹配： • ROOTNAME • ROOTPATH • ROOTOFFSET • ROOTSIZE • PHYSFILE • LOGFILES • LOGSIZE
5		（可选）将 UPDATABLE_SECONDARY 配置参数设置为正整数。
6		向主服务器的 sqlhosts 文件添加条目：

步骤	在主服务器上	在辅助服务器上
		dbservername nettype hostname servicename
7		启动 SD 辅助服务器 oninit

添加辅助服务器时，LGR 内存池中的内存使用量将增加。

请参阅《GBase 8s 管理员参考》 以获取有关配置参数的信息。

获取 SD 辅助服务器统计信息

使用 gstat 实用程序或系统监视接口 (SMI) 表来查看 SD 辅助服务器统计信息。

使用 gstat -g sds 来查看 SD 辅助服务器统计信息。gstat 实用程序的输出取决于实用程序是在主服务器还是在辅助服务器上运行。

查询 **syssrcsds** 表可获取有关主服务器上共享磁盘统计信息的信息。

查询 **systrgsds** 表可获取有关辅助服务器上共享磁盘统计信息的信息。

有关 gstat 和 SMI 表的信息，请参阅《GBase 8s 管理员参考》。

SD 辅助服务器配置

将 SD 辅助服务器升级为主服务器

通过在 SD 辅助服务器上发出以下命令，将 SD 辅助服务器转换为主服务器：

```
gadmin -d set SDS primary <alias>
```

SD 辅助服务器不能转换为标准服务器。

将主服务器转换为标准服务器

通过在主服务器上使用下列命令，可以将主服务器转换为标准服务器，并断开与共享磁盘环境的连接。

```
gadmin -d clear SDS primary <alias>
```

SD 辅助服务器安全性

SD 辅助服务器支持与 HDR 类似的加密规则。请参阅集群的数据库服务器配置需求以获取详细信息。

可启用或禁用任何主服务器和辅助服务器对之间的加密。即，可以加密主服务器与某台 SD 辅助服务器之间的流量，而对主服务器与其他 SD 辅助服务器之间的流量不进行加密。

有关设置和配置主服务器与 SD 辅助服务器之间加密的其他信息，请参阅服务器多路复用器组 (SMX) 连接主题。

6.3 集群管理

本章描述用于监视和维护集群的各种管理任务（其中部分可选）。例如，用于优化性能的负载均衡，以及确保安全性。

6.3.1 数据复制的工作原理

这些主题描述了数据库服务器用于对辅助服务器执行数据复制的机制。有关如何设置、启动和管理各种辅助服务器的指示信息，请参阅下表。

表 1. 辅助服务器设置信息

辅助服务器类型	相关指南
HDR 辅助服务器	请参阅高可用性集群配置，有关使用外部备份与复原启动 HDR 对的信息，请参阅《GBase 8s 备份与复原指南》。
RS 辅助服务器	请参阅远程独立辅助服务器。
SD 辅助服务器	请参阅共享磁盘辅助服务器

数据初始复制的工作原理

HDR 辅助服务器和 RS 辅助服务器使用存储空间备份和逻辑日志备份（备份至磁带的和磁盘上的均包括在内）执行从主数据库服务器到辅助数据库服务器的数据初始复制。

SD 辅助服务器不需要来自主服务器的备份与复原，因为 SD 辅助服务器与主服务器共享同一磁盘。

要复制数据，请执行以下操作：

1. 在两个数据库服务器上均安装用户定义的类型和用户定义的例程。
2. 仅在主数据库服务器上注册用户定义的类型和用户定义的例程。
3. 要使两个数据库服务器管理的数据同步，请在主数据库服务器上创建所有存储空间的 0 级备份。
4. 复原数据复制对中辅助数据库服务器上备份的所有存储空间。

   然后，最近步中从存储空间备份复原的辅助数据库服务器从主数据库服务器读取自该备份以来所生成的所有逻辑日志记录。

数据库服务器首先从所有不再在磁盘上的备份逻辑日志文件读取逻辑日志记录，然后从所有在磁盘上的逻辑日志文件读取。

有关复制数据的详细信息，请参阅首次启动 HDR。《GBase 8s 备份与复原指南》说明了如何使用 ON-Bar 启动复制。

您必须对存储空间备份执行初始备份。您无法使用诸如 gload 和 gunload 之类的数据迁移实用程序来复制数据，因为每个数据库服务器上的物理页布局必须相同，以便数据复制工作。

主服务器数据到辅助服务器的复制

所有辅助服务器类型都使用日志来复制主服务器复制数据。主服务器将其整个逻辑日志发送至 HDR 和 RS 辅助服务器，但仅将日志页面的位置发送至 SD 辅助服务器。

索引页面日志记录可供所有辅助服务器使用，但对于到 RS 辅助服务器的复制而言是必需的。

数据库必须使用事务日志记录才能进行复制。

到 HDR 辅助服务器的复制

主数据库服务器可使用三种同步方式将数据复制到 HDR 辅助服务器：

完全同步方式，其中事务需要 HDR 辅助服务器上的完成确认，然后才能完成。

在您使用完全同步方式时，数据完整性最高，但是如果客户机应用程序使用无缓冲日志记录并且有许多小事务，那么系统性能可能会受到负面影响。

- 完全同步方式，其中事务需要 HDR 辅助服务器上的完成确认，然后才能完成。

  在您使用完全同步方式时，数据完整性最高，但是如果客户机应用程序使用无缓冲日志记录并且有许多小事务，那么系统性能可能会受到负面影响。
- 异步方式，其中事务无需 HDR 辅助服务器上的已接收或已完成确认即可完成。

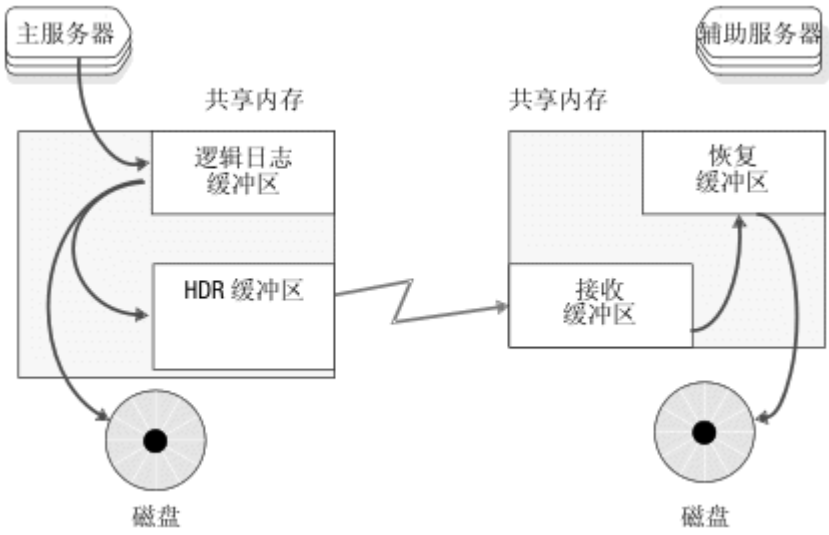
  在使用异步方式时，系统性能最佳，但是如果发生服务器故障，那么数据可能丢失。
- 接近同步方式，其中事务需要 HDR 辅助服务器上的已接收确认，然后才能完成。

  接近同步方式可提供比完全同步方式更优异的性能，比异步方式更优异的数据完整性。如果用于无缓冲日志记录，那么 SYNC 方式（在 DRINTERVAL 设置为 -1 时开启）与接近同步方式相同。

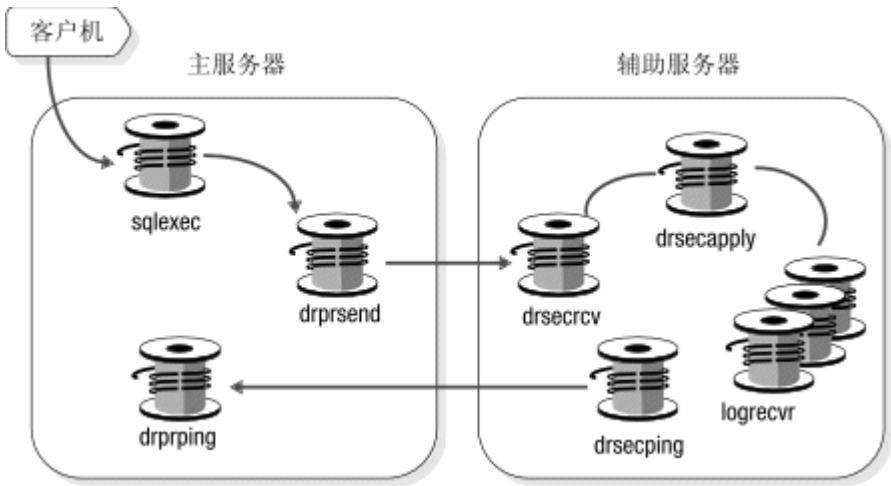
同步方式由 DRINTERVAL 配置参数值、HDR_TXN_SCOPE 配置参数值和数据库日志记录类型的组合进行控制。

以下两张图演示了从主服务器至 HDR 辅助服务器的复制。

图： 从主服务器至 HDR 辅助服务器的数据复制方式



图： 管理数据复制的线程



主服务器的逻辑日志缓冲区的内容将复制到共享内存的数据复制缓冲区并清空到磁盘。如果主服务器使用的是完全同步或接近同步方式，那么它必须接收到来自 HDR 辅助服务器的确认，然后才能完成逻辑日志清空。 主服务器启动 drprsend 线程以将数据复制缓冲区通过网络传输到辅助服务器的 drsecrcv 线程，然后该线程会将数据写入共享内存的接收缓冲区。drsecapply 线程将接收缓冲区复制到恢复缓冲区。HDR 和 RS 辅助服务器都使用 logrecvr 线程来为逻辑日志记录应用其数据库空间。您可以通过更改 OFF_RECVRY_THREADS 配置参数的值来调整 logrecvr 线程的数量。

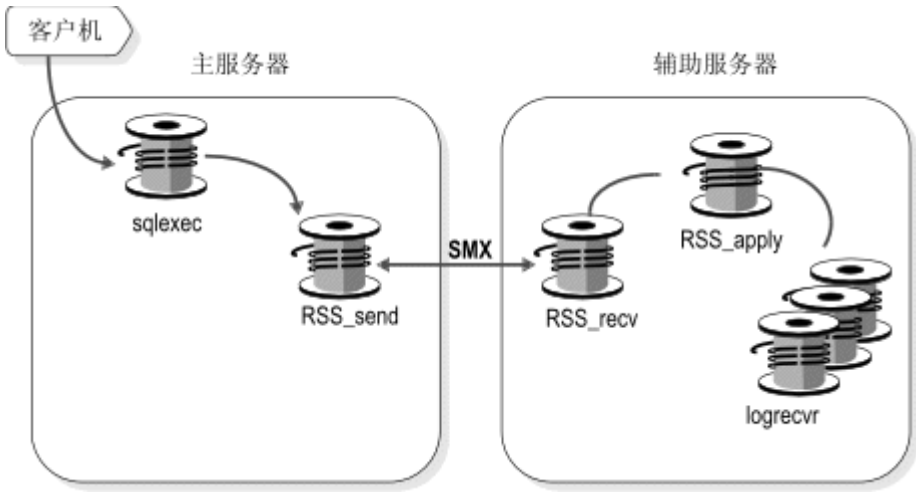
drprping 和 **drsecping** 线程发送并接收消息以监视两个服务器之间的连接。

到 RSS 辅助服务器的复制

因为主服务器和 RS 辅助服务器之间的检查点是异步的，因此 RS 辅助服务器需要索引页面日志记录。

下图演示了从主服务器至 RS 复制服务器的复制。

图： 管理 RS 辅助服务器的数据复制的线程



如果主服务器可以验证其已连接到 RS 辅助服务器，那么 RSS_send 线程将页面从磁盘或逻辑日志缓冲区复制到数据复制缓冲区。**RSS_Send** 线程使用“服务器多路复用器组”(SMX)连接以将数据复制缓冲区发送至 RS 辅助服务器的 **RSS_recv** 线程。然后，**RSS_recv** 线程将数据写入接收缓冲区。**RSS_apply** 线程将接收缓冲区复制到恢复缓冲区。

不同于 HDR 完全同步方式或接近同步方式，主服务器在发送下一个缓冲区之前不需要来自辅助服务器的确认。在 **RSS_send** 线程等待 **RSS_Recv** 线程接收来自 RS 辅助服务器的确认之前，主服务器最多发送 32 个未确认的数据复制缓冲区。

到 SD 辅助服务器的复制

SD 辅助服务器从磁盘读取逻辑日志页，然后将数据应用于其内存数据缓冲区。

HDR 复制的完全同步方式

HDR 完全同步方式确保主服务器上已落实的任何事务也在 HDR 辅助服务器上落实，这可以在发生故障时保护事务一致性。

主数据库服务器将逻辑日志缓冲区内容写入 HDR 缓冲区后，会将缓冲区中的记录发送至 HDR 辅助数据库服务器。仅当主数据库服务器接收到来自 HDR 辅助数据库服务器的确认（已收到记录）之后，主数据库服务器上的逻辑日志缓冲区清空才会完成。

为跟踪同步，主服务器和 HDR 辅助服务器都在其保留的页面中存储以下信息：

- 包含最近完成的检查点的逻辑日志文件的标识
- 逻辑日志文件中检查点记录的位置
- 上次发送或接收的逻辑日志文件的标识

- 上次发送或接收的逻辑日志记录的页码

要查看此信息，请运行 gstat -g dri ckpt 命令。

HDR 复制对中的数据库服务器之间的检查点是同步的。主服务器会等待 HDR 辅助服务器确认收到了检查点日志记录后，再完成其检查点。如果检查点未在 DRTIMEOUT 配置参数指定的时间内完成，那么主数据库服务器假定发生了故障。

HDR 完全同步方式具有以下需求：

- 主服务器和 HDR 辅助服务器上的 DRINTERVAL 配置参数必须设置为 0。
- 主服务器和 HDR 辅助服务器上的 DRTIMEOUT 配置参数必须设置为相同值。

如果主服务器和 HDR 辅助服务器上的操作系统时间是同步的，那么管理可以更轻松。

要开启完全同步数据复制，请将 DRINTERVAL 配置参数设置为 0，然后使用以下某种方法：

- 将 HDR_TXN_SCOPE 配置参数设置为 FULL_SYNC。
- 运行 SET ENVIRONMENT HDR_TXN_SCOPE 'FULL_SYNC';

日志记录将按照其接收顺序进行应用。在日志传输缓冲区包含许多日志记录时，在 HDR 辅助服务器上应用这些日志记录将需要更多时间，并且性能可能受到负面影响。如果发生此情况，请考虑将接近同步方式用于 HDR 数据复制。

HDR 复制的接近同步方式

在使用 HDR 复制的接近同步方式时，主服务器在收到有关 HDR 辅助服务器已接收所传输事务的确认后，将逻辑日志缓冲区清空至磁盘。主服务器不会等待有关在 HDR 辅助服务器上已落实事务的确认。

在日志传输缓冲区包含许多日志记录时，在 HDR 辅助服务器上应用这些日志记录将需要更多时间。HDR 复制的接近同步方式提供比完全同步方式更优异的性能，比异步方式更优异的数据完整性。

主服务器在其保留的页面中存储以下接近同步信息：

- 已排队至 drprsend 线程的未处理数据复制缓冲区的数量。
- 日志唯一值，最近分页的日志的页码。
- 线程控制块 (TCB) 的指针、带括号的线程标识以及该线程已执行的落实的日志序号 (LSN)。
- 等待 HDR 辅助服务器上“已接收”确认的落实的 LSN。

要查看此信息，请运行 gstat -g dri que 命令。

复制对中的数据库服务器间的检查点是同步的。主服务器会等待 HDR 辅助服务器确认收到了检查点日志记录后，再完成其检查点。如果检查点未在 DRTIMEOUT 配置参数指定的时间内完成，那么主数据库服务器假定发生了故障。

HDR 接近同步方式具有以下需求：

- 主服务器和 HDR 辅助服务器上的 DRINTERVAL 配置参数必须设置为 -1，或者主服务器上的 DRINTERVAL 配置参数必须设置为 0。
- 主服务器和 HDR 辅助服务器上的 DRTIMEOUT 配置参数必须设置为相同值。
- 主服务器和 HDR 辅助服务器上的操作系统时间必须是同步的。

要开启接近同步数据复制，请将 DRINTERVAL 配置参数设置为 0，然后使用以下某种方法：

- 将 HDR_TXN_SCOPE 配置参数设置为 NEAR_SYNC。
- 运行 SET ENVIRONMENT HDR_TXN_SCOPE 'NEAR_SYNC';

HDR 复制的异步方式

异步 HDR 复制意味着主服务器不等待来自 HDR 辅助服务器的响应，就将逻辑日志清空至磁盘。异步 HDR 复制可提高复制速度，但可能会丢失事务。

可通过多种方式开启 HDR 复制的异步方式：

- 将 DRINTERVAL 配置参数设置为正整数值。
- 将 DRINTERVAL 配置参数设置为 0，并将 HDR_TXN_SCOPE 配置参数设置为 ASYNC。
- 运行以下语句：

```
SET ENVIRONMENT HDR_TXN_SCOPE 'ASYNC';
```

在异步方式中，主数据库服务器在将逻辑日志缓冲区的内容复制到数据复制缓冲区后，将逻辑日志清空至磁盘。在发生以下某个条件时，主数据库服务器将通过网络发送 HDR 缓冲区的内容：

- HDR 缓冲区变满。
- 自记录发送至 HDR 辅助数据库服务器以来的时间间隔超出主服务器的 DRINTERVAL 配置参数值。

要降低使用异步复制的集群中丢失事务的风险，请为所有数据库使用无缓冲日志记录。 无缓冲日志记录缩短了事务记录等待和传输之间的时间量。如果主服务器使用缓冲日志记录，并且您收到 error -7350 Attempt to update a stale version of a row 消息，那么请切换至无缓冲日志记录。

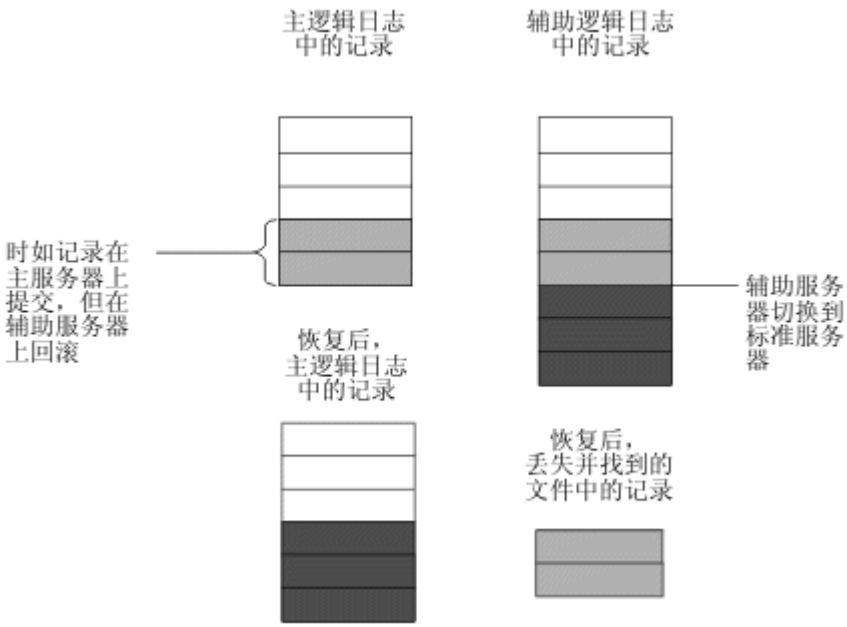
如果发生故障转移，但是主服务器重新启动后带有数据复制，那么主服务器上已落实但辅助服务器上未落实的事务将存储在 DRLOSTFOUND 配置参数指定的文件中。

失而复得的事务

使用异步更新时，在主数据库服务器上落实的事务可能未在辅助数据库服务器上复制。如果在主数据库服务器将落实记录复制到 HDR 缓冲区之后但在主数据库服务器将该落实记录发送至辅助数据库服务器之前发生故障，就会产生这种情况。

如果在主数据库服务器故障后将辅助数据库服务器更改为标准数据库服务器，它会回滚所有打开的事务。这些事务包括任何在主数据库服务器上落实但辅助数据库服务器未接收到其落实记录的事务。因此，事务在主数据库服务器上但未在辅助数据库服务器上落实。当您在故障后重新启动数据复制，那么数据库服务器在主数据库服务器的逻辑恢复期间将丢失的事务中的所有逻辑日志记录放置在某文件（DRLOSTFOUND 配置参数指定）中。下图说明了该过程。

图： 使用失而复得的文件



如果在重新启动数据复制后的运行主数据库服务器的计算机上创建不失而复得的文件，那么表示已丢失事务。数据库服务器无法重新应用失而复得文件中的事务记录，因为在辅助数据库服务器充当标准数据库服务器时可能发生了有冲突的更新。

为了减少不用同步方式运行数据复制而丢失事务的风险，请为所有数据库使用未缓冲日志记录。该方法减少了从主数据库服务器到辅助数据库服务器的写入和事务记录传送之间所需的时间量。

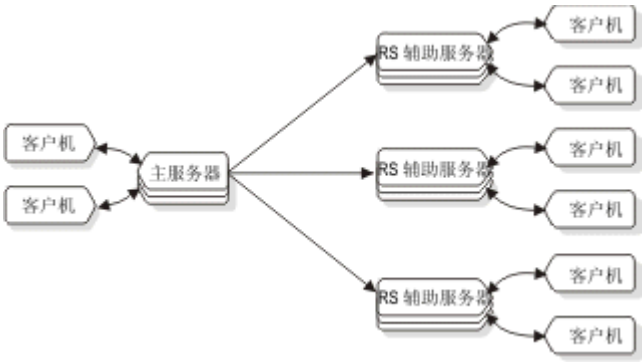
数据复制配置示例

这些主题描述了如何配置数据复制环境的一些示例。

远程独立辅助配置示例

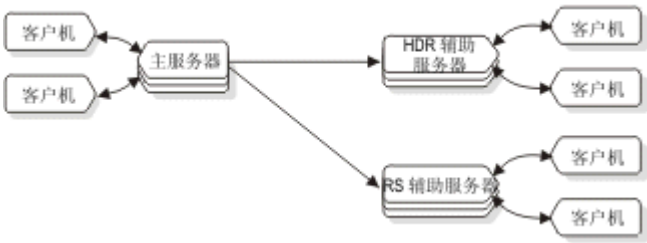
下图说明了由多个 RS 辅助服务器组成的配置的示例。当主服务器距离 RS 辅助服务器很远或主服务器和 RS 辅助服务器之间的网络速度较慢或不稳定时，该配置可能有用。因为 RS 辅助服务器使用全双工通信协议，而且不需要同步检查点处理，因此主服务器的性能通常不受影响。

图： 具有三台 RS 辅助服务器的主服务器



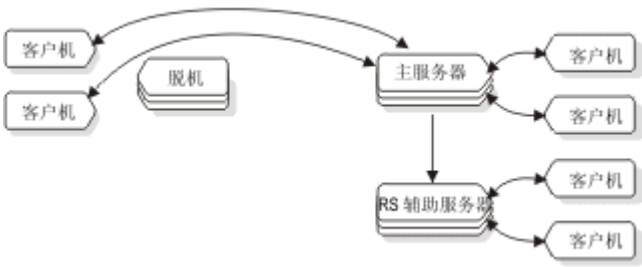
下一幅图显示了配置 RS 辅助服务器以及 HDR 辅助服务器的示例。 在该示例中，如果主服务器和 HDR 辅助服务器均丢失，那么 HDR 辅助服务器提供高可用性，而 RS 辅助服务器提供额外的灾难恢复功能。 RS 辅助服务器在地理位置上可距离主服务器和 HDR 辅助服务器很远，因此区域性的破坏（如地震或洪水）不会对 RS 辅助服务器造成影响。

图： 具有 HDR 辅助服务器和 RS 辅助服务器的主服务器



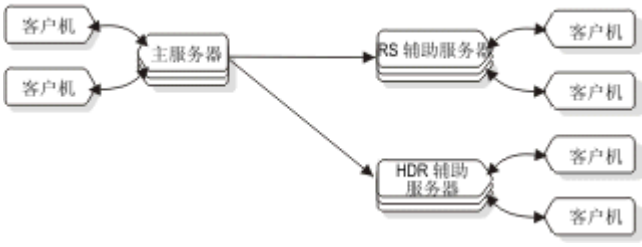
如果主数据库服务器发生故障，那么可将现有 HDR 辅助服务器转换为主服务器，如下图中所示：

图： 从主服务器故障转移到 HDR 辅助服务器



如果原始主服务器将处于脱机状态的时间过长，那么可将 RS 辅助服务器转换为 HDR 辅助服务器。当原始主服务器恢复为联机状态时，再将其配置为 RS 辅助服务器，如下图所示：

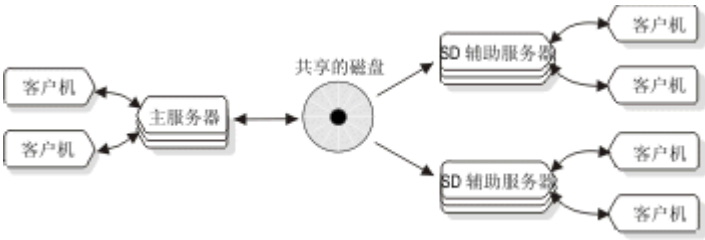
图： 充当 HDR 辅助服务器角色的 RS 辅助服务器



共享磁盘辅助配置示例

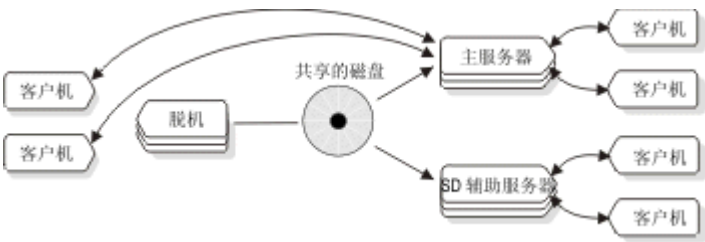
下图显示了带有两台 SD 辅助服务器的主服务器的示例。在这种情况下，主服务器的角色可以转换为两台 SD 辅助服务器中的任意一个。无论是因为计划的中断还是因为主服务器故障而导致主服务器必须停止服务均是如此。

图： 配置有两台 SD 辅助服务器的主服务器



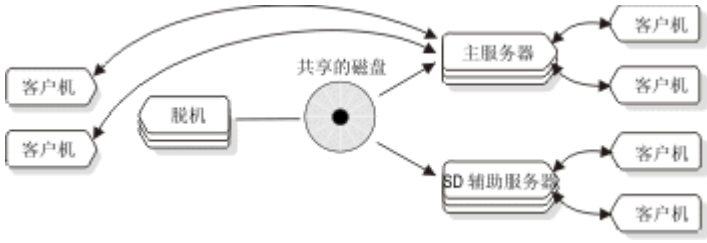
因为两台 SD 辅助服务器都是从同一磁盘子系统进行读取，因此它们都同样能充当主服务器角色。下图说明了主服务器处于脱机状态的情境。

图： 充当主服务角色的 SD 辅助服务器



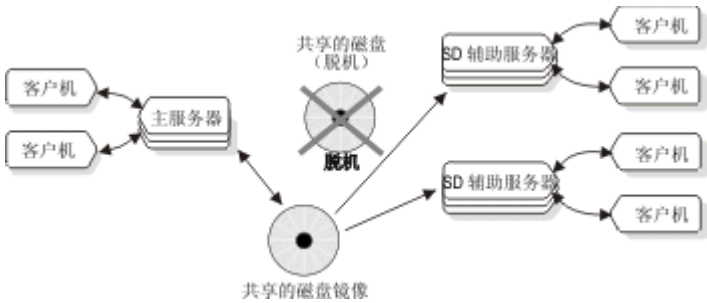
有多种方法来防止共享磁盘发生硬件故障。可能最普通的方法就是基于 RAID 技术（例如 RAID-5）配置磁盘阵列。防止磁盘故障的另一种方法是使用 SAN（存储区域技术）来包含远程磁盘镜像的某些格式。因为 SAN 磁盘可位于距离主磁盘及其镜像较近的位置，因此这可为服务器或磁盘子系统的计划中断和意外中断提供高度可用性。下图描述了这样的配置：

图： 具有镜像磁盘的主服务器和 SD 辅助服务器



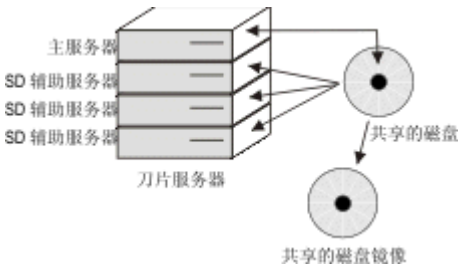
如果发生磁盘故障，那么可按下图中的描述重新配置服务器：

图： 主共享磁盘发生故障之后的共享磁盘镜像



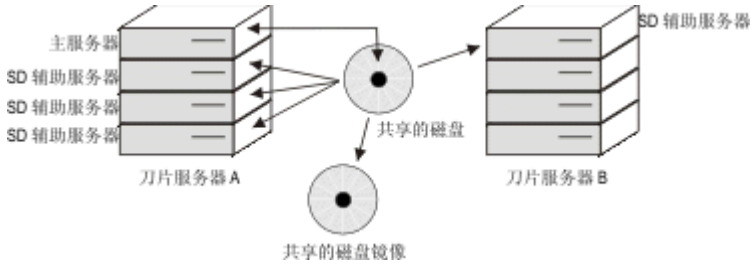
除了按前一图中的描述配置镜像磁盘子系统外，您可能想要配置其他服务器。 例如，您可能想要在单个刀片服务器机柜中使用主服务器和两台 SD 辅助服务器。通过将服务器组放置在单台刀片服务器中，该刀片服务器本身可成为故障点。必须定期增强读处理能力时（例如，执行大型报告任务时），下图中的配置是一种极具吸引力的解决方案。

图： 刀片服务器中的主服务器和 SD 辅助服务器



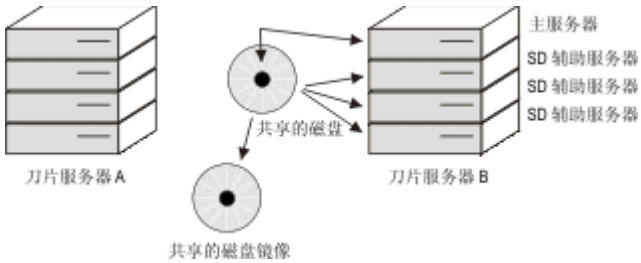
您可以决定通过使用多台刀片服务器来避免单台刀片服务器可能引起的故障点，如下图中所示。

图： 用于防止单点故障的多台刀片服务器配置



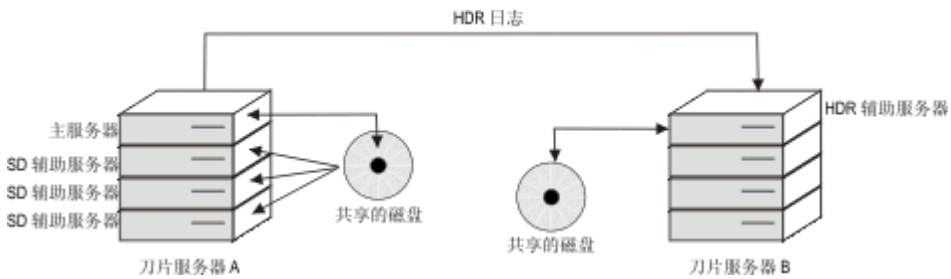
在前一图中，如果刀片服务器 A 发生故障，那么它可以将主服务器角色转换为刀片服务器 B 上的 SD 辅助服务器。因为它可以快速地使附加 SD 辅助服务器联机，所以它可以动态地将附加 SD 辅助服务器添加到刀片服务器 B，如下图所示。

图： 刀片服务器发生故障后进行故障转移



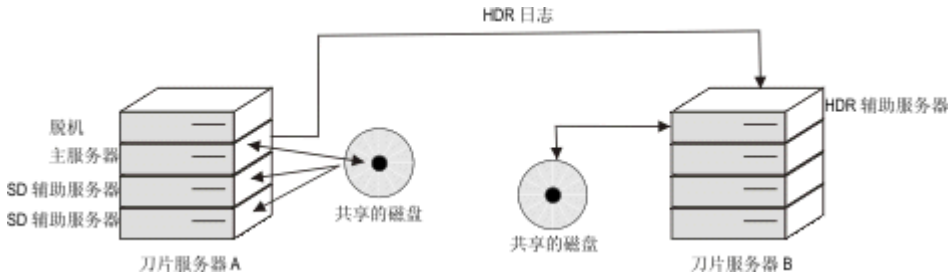
因为主服务器与磁盘镜像可支持的镜像磁盘之间的距离存在限制，所以您可能需要考虑使用共享磁盘和依靠共享磁盘镜像来提供磁盘可用性。例如，您可能希望磁盘子系统的两个副本之间距离较大。在这种情况下，可能选择使用 HDR 辅助服务器或 RS 辅助服务器来维护磁盘子系统的辅助副本。如果网络连接非常快（即，如果对辅助服务器执行 ping 操作的时间短于 50 毫秒），那么必须考虑使用 HDR 辅助服务器。对于较慢的网络连接，请考虑使用 RS 辅助服务器。下图显示了刀片服务器配置中 HDR 辅助服务器的示例。

图： 刀片服务器配置中的 HDR 辅助服务器



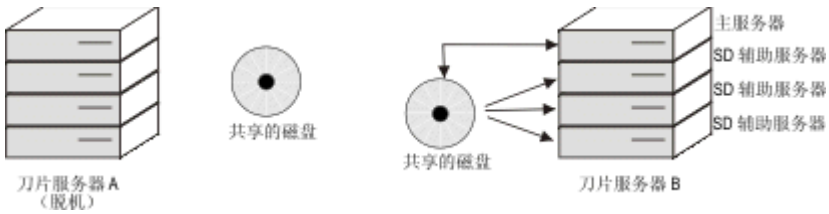
在前一图中显示的配置中，如果主节点发生故障，但是共享磁盘未损坏而且刀片服务器仍然可运行，那么可以将主服务器角色从刀片服务器 A 中的第一台服务器转换为同一刀片服务器中的另一台服务器。 更改主服务器会导致远程 HDR 辅助服务器的源自动重新路由至新的主服务器，如下图中所示：

图： 刀片服务器配置中从主服务器故障转移到 SD 辅助服务器



但是，假设前一图中描述的故障不是刀片服务器中的某个刀片故障，而是整台刀片服务器的故障。在这种情况下，可能需要故障转移到 HDR 辅助服务器。因为 SD 辅助服务器的启动非常快速，所以可以轻易地添加更多 SD 辅助服务器。请注意，SD 辅助服务器仅可处理主节点；当主服务器转换为刀片服务器 B，那么还可能启动刀片服务器 B 上的 SD 辅助服务器，如下图所示。

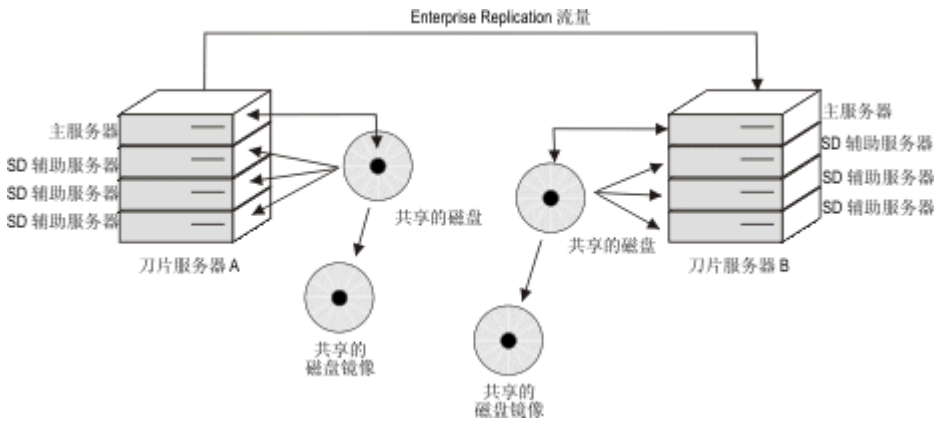
图： 整台刀片服务器发生故障



Enterprise Replication 作为可恢复组的一部分

虽然 Enterprise Replication 不支持操作的 SYNC（同步）方式，但它有能力支持具有多台活动服务器的环境。在故障转移事件期间，Enterprise Replication 可以协调两台服务器之间的数据库差异。必须将 Enterprise Replication 视为提高服务器之间同步的方法，因为每个 Enterprise Replication 系统都维护一个独立的日志记录系统。下图中显示了使用 Enterprise Replication 的配置。

图： 将 Enterprise Replication 配置为可恢复组的一部分



使用 Enterprise Replication 的高可用性集群配置示例

假设在两个高可用性服务器集群之间需要 Enterprise Replication，请如下所示进行配置：

集群 1：

- 主服务器
- HDR 辅助服务器
- SD 辅助服务器 1
- SD 辅助服务器 2
- RS 辅助服务器 1
- RS 辅助服务器 2

集群 2：

- 主服务器
- HDR 辅助服务器
- SD 辅助服务器 1
- SD 辅助服务器 2
- RS 辅助服务器 1
- RS 辅助服务器 2

进一步假设每台服务器按照以下约定命名：

- 前三个字符：企业的名称
- 第四个字符：主机的简短数字
- 第 5、第 6 和第 7 个字符：集群编号
- 第 8、第 9 和第 10 个字符：服务器类型：“pri”表示主服务器，“sec”表示辅助服务器
- 第 11、第 12 和第 13 个字符：连接类型：“shm”或“tcp”

例如，名为 srv4_1_pri_shm 的服务器描述如下：

- srv = 企业的名称
- 4 = 主机简短编号
- _1_ = 集群编号
- pri = 这是一个主服务器
- shm = 连接类型使用共享内存通信

sqlhosts 文件中的以下条目将支持上面的配置：

```
srv4_1_pri_shm onipcshm sun-mach4 srv4_1_pri_shm
srv4_1_sec_shm onipcshm sun-mach4 srv4_1_sec_shm
srv5_1_rss_shm onipcshm sun-mach5 srv5_1_rss_shm
```

```
srv5_1_sds_shm onipcshm sun-mach5 srv5_1_sds_shm
srv6_1_rss_shm onipcshm sun-mach6 srv6_1_rss_shm
srv6_1_sds_shm onipcshm sun-mach6 srv6_1_sds_shm
srv_1_cluster group - - i=1
srv4_1_pri_tcp ontlitcp sun-mach4 21316 g=srv_1_cluster
srv4_1_sec_tcp ontlitcp sun-mach4 21317 g=srv_1_cluster
srv5_1_rss_tcp ontlitcp sun-mach5 21316 g=srv_1_cluster
srv5_1_sds_tcp ontlitcp sun-mach5 21317 g=srv_1_cluster
srv6_1_rss_tcp ontlitcp sun-mach6 21316 g=srv_1_cluster
srv6_1_sds_tcp ontlitcp sun-mach6 21317 g=srv_1_cluster

srv4_2_pri_shm onipcshm sun-mach4 srv4_2_pri_shm
srv4_2_sec_shm onipcshm sun-mach4 srv4_2_sec_shm
srv5_2_rss_shm onipcshm sun-mach5 srv5_2_rss_shm
srv5_2_sds_shm onipcshm sun-mach5 srv5_2_sds_shm
srv6_2_rss_shm onipcshm sun-mach6 srv6_2_rss_shm
srv6_2_sds_shm onipcshm sun-mach6 srv6_2_sds_shm
srv_2_cluster group - - i=2
srv4_2_pri_tcp ontlitcp sun-mach4 21318 g=srv_2_cluster
srv4_2_sec_tcp ontlitcp sun-mach4 21319 g=srv_2_cluster
srv5_2_rss_tcp ontlitcp sun-mach5 21318 g=srv_2_cluster
srv5_2_sds_tcp ontlitcp sun-mach5 21319 g=srv_2_cluster
srv6_2_rss_tcp ontlitcp sun-mach6 21318 g=srv_2_cluster
srv6_2_sds_tcp ontlitcp sun-mach6 21319 g=srv_2_cluster
```

复杂故障转移恢复策略的示例

本主题描述了用于在出现较大区域范围灾难时，实现最大可用性的三层服务器方法。

通常，HDR 辅助服务器提供 SD 辅助服务器的备份，并支持在地理位置上距离主系统较远的高度可用系统。RS 辅助服务器为 HDR 辅助服务器提供附加可用性，并视为灾难可用性解决方案。如果必须使用 RS 辅助服务器确保可用性，那么将强制您通过执行备份与复原来手动重新构建其他系统，以恢复正常运行。要进一步理解此内容，将提供出现较大范围灾难的场景，例如风暴。

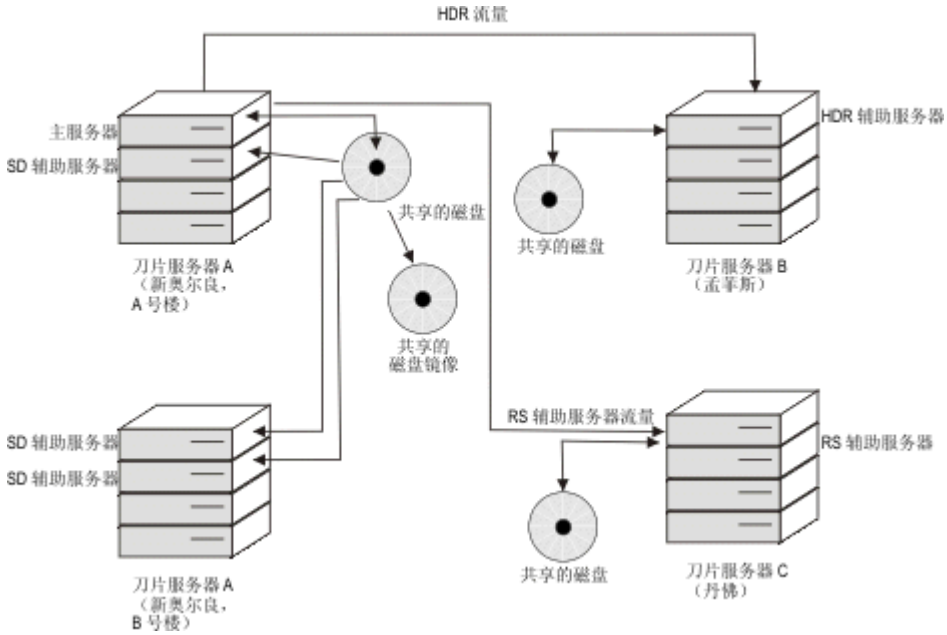
要提供最大可用性以免受区域性灾难的影响，需要分层可用性。第一层提供的可用性解决方案可处理短时间的本地故障。例如，这可能包括将一些刀片服务器连接到运行 SD 辅助服务器的单个磁盘子系统。将 SD 辅助服务器放置在校园内的多个位置可在发生本地中断时提供无缝故障转移。

您可能想要添加第二层，通过包含带有其自身磁盘副本的备用位置来提高可用性。要防护较大范围的灾难，您可能需要考虑配置位于较远位置（可能几百英里）的 HDR 辅助服务

器。也可能想要使用远程系统作为刀片服务器或某些其他多服务器系统。如果发生故障转移并且远程 HDR 辅助服务器成为主服务器，那么通过提供第二层可轻松启动远程站点上的 SD 辅助服务器。

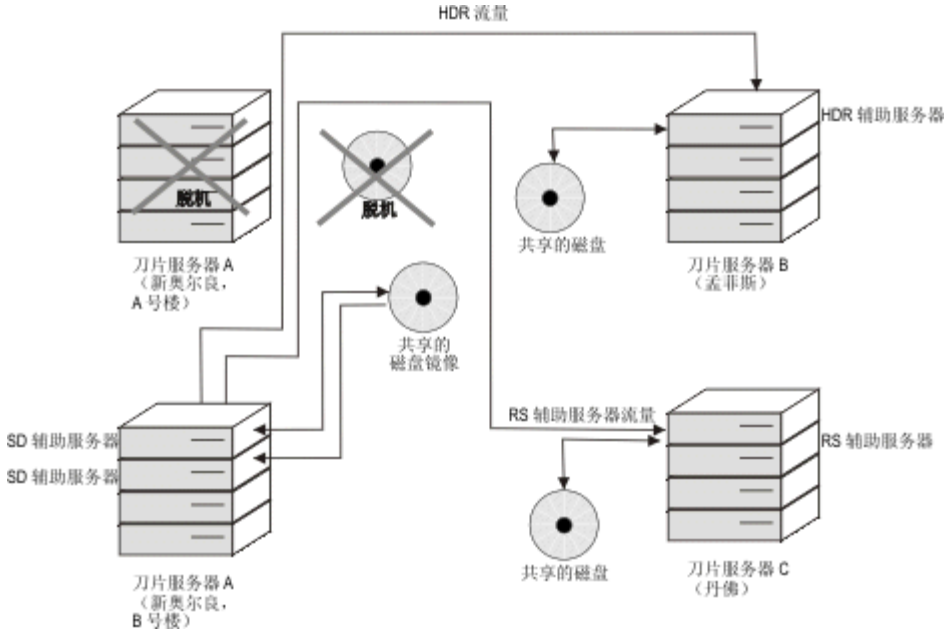
但是，即使是两层的方法也可能不够用。某个区域内的飓风会在几百公里外生成龙卷风。要防护这种情况，请考虑添加第三层保护，例如位于数千英里的 RS 辅助服务器。此三层方法提供了可很大程度减少中断风险的额外冗余。

图： 三层服务器可用性的配置



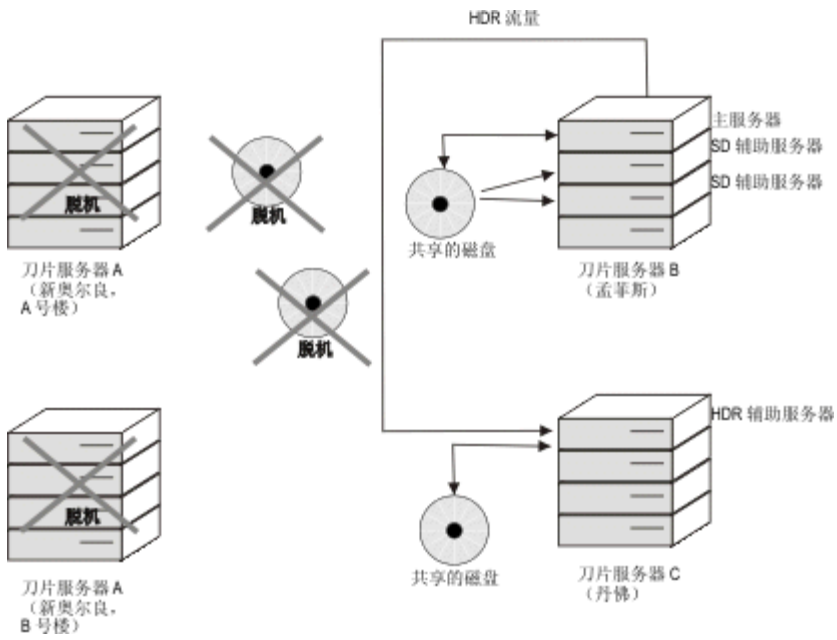
现在假设新奥尔良校园的建筑物 A 中发生了本地中断。可能是机房内的水管破裂使水对刀片服务器和共享磁盘子系统的主副本造成了损害。通过运行 gadmin -d 以使主服务器名位于建筑物 B 中的刀片服务器上运行的某台 SD 辅助服务器上来将主服务器的角色切换为建筑物 B。这将导致其他所有辅助节点自动连接到新的主节点。

图： 第一层保护



如果新奥尔良发生了区域性中断，导致建筑物 A 和建筑物 B 均丢失，那么您可以将主服务器角色切换至孟菲斯。此外，您也可能使丹佛进入到 HDR 辅助服务器，并可将附加 SD 辅助服务器添加到孟菲斯中的机器。

图： 第二层保护



影响两个站点的更大型中断应需要切换至最远的系统。

图： 第三层保护

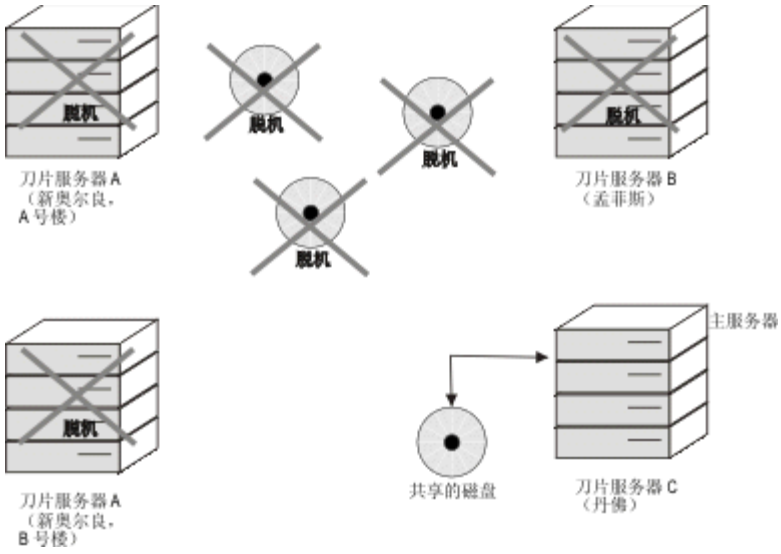


表 1. 各种需求的建议配置

要求	建议配置
必须定期增大报告容量	请使用 SD 辅助服务器
您使用的是提供足够磁盘硬件可用性的 SAN 设备，但是担心发生服务器故障	请使用 SD 辅助服务器
您使用的是提供足够磁盘硬件镜像的 SAN 设备，但是也需要当主操作丢失（并且镜像磁盘的限制不是问题）时能够恢复联机状态的第二组服务器	考虑使用在两个站点上运行 SD 辅助服务器的两台刀片服务器中心
您需要具有距离适中的备份站点，但不能容忍故障转移期间出现任何数据丢失	考虑使用两台刀片服务器中心，SD 辅助服务器在主刀片服务器中心上，HDR 辅助服务器在远程刀片服务器上。
您需要具有未曾丢失事务的高度可用系统，但是还必须在世界的另一面上设置远程系统	考虑使用针对数据复制运行完全同步方式或接近同步方式的本地 HDR 辅助服务器，并同时在世界另一端使用 RS 辅助服务器。
您需要具有高可用性解决方案，但是因为您所在区域中的网络，ping 的最佳响应时间为大约 200 ms	考虑使用 RS 辅助服务器

要求	建议配置
您需要备份站点，但不具有与备份站点之间的任何直接通信	考虑使用带有备份与复原的“连续日志复原”
只要数据最终能够到达目的地，您就能够忍受数据交付过程中的延迟；但是在任何情况下都必须具有快速故障转移	考虑使用硬件磁盘镜像与 ER 相结合的 SD 辅助服务器。
您需要其他写处理能力，能够容忍这些写操作交付中有些延迟，需要高度可用的事务并能够分割工作负载	考虑使用带有 SD 辅助服务器的 ER

高可用性集群环境故障诊断

高可用性集群环境与独立服务器环境相比，需要很少甚至不需要额外的故障诊断。 本主题说明了用于描述高可用性集群环境的术语，并提供了一些常见的故障诊断过程。

可以使用诊断工具显示高可用性环境的配置，并验证是否正确设置辅助服务器来更新数据。服务器会快速处理事务。gstat 命令只在运行瞬间显示状态信息。

为了更新辅助服务器上的数据，GBase 8s 在主数据库服务器和辅助数据库服务器上创建了代理分发器。为每个代理分发器都分配了一个在集群内唯一的标识。 代理分发器负责将 DML 更新请求从辅助服务器发送至主服务器。 辅助服务器根据辅助服务器的 onconfig 文件中的 UPDATABLE_SECONDARY 设置确定要创建的代理分发器的实例数。

对于高可用性集群环境中的可更新辅助服务器，从可更新辅助服务器到主服务器的加密需要 SMX 加密。要加密从可更新辅助服务器发送到主服务器的数据，请在辅助服务器上设置 ENCRYPT_SMX 配置参数。请参阅启用 SMX 加密以获取更多信息。

在高可用性集群中初始化可更新的辅助服务器时，该服务器将保持快速恢复方式，直到所有打开的事务（包括已打开和已准备好的 XA 事务）完成为止。处于快速恢复方式时，应用程序不能连接到该服务器。在所有已打开的事务落实或回滚之前，服务器将保持快速恢复方式。

在主服务器上使用 gstat -g proxy 命令查看有关高可用性集群中所有代理分发器的信息。

```
gstat -g proxy

Secondary       Proxy       Reference       Transaction   Hot Row
Node             ID         Count           Count         Total
serv2           392          0               2            112
serv2           393          0               2            150
```

检查前一示例中 gstat 命令的输出，两个代理分发器的标识为 392 和 393. 事务计数显示每个代理分发器当前正在处理的事务数。

在辅助服务器上运行 gstat -g proxy，以便查看有关能够为来自辅助服务器的更新请求提供服务的代理分发器的信息。

```
gstat -g proxy

Primary      Proxy       Reference       Transaction   Hot Row
Node          ID          Count           Count         Total
serv1        392          0               2             112
serv1        393          0               2             150
```

在此示例中，服务器是共享磁盘 (SD) 辅助服务器，并且已配置为更新数据。 另外，服务器被连接到命名为 serv1 的主服务器。并且具有两个代理分发器，每个的事务计数为 2。

在主服务器上使用 gstat -g proxy all，以便显示有关代理分发器和代理线程的信息。代理分发器创建了一个或多个代理线程以用于处理来自辅助服务器的数据更新。

```
gstat -g proxy all

Secondary   Proxy       Reference       Transaction   Hot Row
Node          ID          Count           Count         Total
serv2        392          0               2             1
serv2        393          0               2             0

TID        Flags          Proxy   Source    Proxy     Current    sqlerrno iserrno
                                             ID        SessID     TxnID    Seq
63         0x00000024 392      22          1         5          0        0
64         0x00000024 392      19          2         5          0        0
62         0x00000024 393      23          1         5          0        0
65         0x00000024 393      21          2         5          0        0
```

在 gstat -g proxy all 命令输出中，TID 表示在主服务器上运行的代理线程的标识。 Source SessID 表示辅助服务器上的用户会话标识。Proxy TxnID 显示当前事务的序号。 每个 Proxy TxnID 对于代理分发器而言都是唯一的。Current® Seq 表示正在处理的事务中当前操作的序号。每个发送至辅助服务器的数据库事务都被从内部分成一个或多个操作，然后将这些操作发送至主服务器。 最后两个字段 sqlerrno 和 iserrno 显示在事务中遇到的任何 SQL 或 ISAM/RSAM 错误。 错误数 0 表示完成且没有错误。

在辅助服务器上运行 gstat -g proxy all 会显示有关当前可更新辅助服务器上数据的所有会话的信息。

```
gstat -g proxy all

Primary     Proxy        Reference       Transaction   Hot Row
Node         ID           Count           Count         Total
serv1       3466          0               0             1
serv1       3465          0               1             0

Session   Proxy      Proxy      Proxy      Current   Pending   Reference
                      ID         TID        TxnID     Seq       Ops        Count
19        3465       67         1          23        0         1
```

在辅助服务器上运行的 gstat -g proxy all 命令生成的输出中，Session 表示该辅助服务器上的用户会话标识。Proxy ID 和 Proxy TID 与在主服务器上显示的一样。Pending Ops 显示正在等待传送至主服务器的操作数。 Reference Count 显示该事务正在使用的线程数。
当 Reference Count显示 0 时，表示事务处理已完成。

要显示由给定分发器执行的当前工作的详细信息，请使用：

```
gstat -g proxy <proxy id> [proxy transaction id] [operation number]
```

代理事务标识和操作编号都是可选参数。 如果已提供，那么第一个编号将视为代理事务标识。 如果其后有第二个编号，那么会将此编号解释为操作编号。 如果没有代理事务标识，该命令执行的任务与以下命令相同：gstat -。与此类似，如果针对给定代理事务标识不存在此类操作编号，那么该命令执行的任务与以下命令相同：gstat -。

使用以下命令显示服务器是否配置为允许对数据进行更新的相关信息。 该命令可以在主服务器或辅助服务器上运行：

```
gstat -g <server_type>
```

示例：

```
gstat -g rssonstat -g sds
gstat -g dri
```

使用来自于主服务器的设置来配置连接管理器

这常常有助于通过与主服务器上相同的配置参数来配置连接管理器实例。 这在现有应用程序指向主服务器，且不希望将这些应用程序重新编译为指向连接管理器时很有用。 通过使用接下来的过程，配置连接管理器实例，以便现有应用程序连接到连接管理器而不是主服务器。

对于此示例：

- 尚未配置连接管理器实例。
- 您拥有主数据库服务器、HDR 辅助服务器、SD 辅助服务器和 RS 辅助服务器。
- 主服务器的 DBSERVERNAME 为 myserv

- 服务名称为 5656

要将所有服务器迁移至新的主服务器 DBSERVERNAME：

1. 使用 DBSERVERNAME 和新的主服务器的服务名称更新所有辅助服务器上的 sqlhosts 文件。

   现有 sqlhosts 文件：

   ```
   #dbservername      nettype         hostname        servicename
   options
   myserv             protocol        host            5656
   ```

   修改的 sqlhosts 文件：

   ```
   #dbservername      nettype         hostname        servicename
   options
   myserv_pri         protocol        host            5656
   ```

2. 在连接管理器服务器上，按照如下所示编辑 sqlhosts 文件：

   ```
   #dbservername      nettype         hostname        servicename     options
   myserv             protocol        host            5656
   ```

3. 在连接管理器服务器上，编辑连接管理器配置文件以设置连接管理器实例的名称：

   ```
   NAME myserv
   SLA ...
   LOG ...
   LOGFILE ...
   ```

   请注意连接管理器“名称”和相应的 sqlhosts 文件值与原先主服务器的相同。 这使客户机应用程序可以连接到连接管理器而无需重新编译应用程序。

4. 启动连接管理器：oncmsm

5. 关闭 SD 辅助服务器和主服务器。

6. 在主服务器上，更新 onconfig 文件中的 DBSERVERNAME 配置参数：
   原有的：

   ```
   DBSERVERNAME   myserv
   ```

   新的：

   ```
   DBSERVERNAME   myserv_pri
   ```

7. 使用以下命令启动主服务器：oninit -SDS=myserv_pri

8. 使用 oninit 命令启动 SD 辅助服务器。

9. 在 RS 辅助服务器上，关闭服务器，然后以物理恢复方式启动：oninit -PHY

10. 在 RS 辅助服务器上，通过以下命令连接到主服务器：gadmin -d RSS myserv_pri

对于 HDR 辅助服务器，不需要更多设置，因为会自动重新建立 HDR 对。 然而，可以执行步骤 6 和 7，以便在 HDR 辅助服务器中注册新的主服务器名称。

设计数据复制组客户机

本主题说明了针对连接至正在运行数据复制的数据库服务器的客户机的各种设计注意事项。

另请参阅辅助服务器上的隔离级别，以获取有关辅助服务器上的 COMMITTED READ 和 COMMITTED READ LAST COMMITTED 隔离级别的信息。

用于排序的临时数据库空间和临时表的使用

即使辅助数据库服务器处于只读方式，当它必须排序或创建临时表时也会进行写操作。临时数据库空间说明数据库服务器在何处找到临时空间以在排序期间使用或用于临时表。

要防止辅助数据库服务器写入处于逻辑恢复方式的数据库空间，您必须采取以下操作：

1. 确保一个或多个临时数据库空间存在。

   有关创建临时数据库空间的指示信息，请参阅创建使用缺省页大小的数据库空间。

2. 执行以下操作之一：

- 将辅助数据库服务器的 onconfig 文件中的 DBSPACETEMP 参数设置为临时数据库空间或数据库空间。
- 将客户机应用程序的 DBSPACETEMP 环境变量设置为临时数据库空间或数据库空间。

在辅助服务器（SD 辅助服务器、RS 辅助服务器和 HDR 辅助服务器）上创建的临时表必须通过 WITH NO LOG 选项来创建。或者，在辅助服务器上将 TEMPTAB_NOLOG 配置参数设置为 1，以便将临时表的缺省日志记录方式更改为无日志记录。 在启用日志记录的情况下创建的表将导致 ISAM 错误。

对于 SD 辅助服务器，设置 SDS_TEMPDBS 配置参数以配置 SD 辅助服务器使用的临时数据库空间。

对于 SD 辅助服务器，不需要显式添加临时数据库空间，因为启动辅助服务器时，该服务器会分配由 SDS_TEMPDBS 指定的块。只需要准备用于接受块的设备。

如果高可用性集群中的主服务器发生故障，并且 SD 辅助服务器作为主服务器接管，那么 SD 辅助服务器上为 SDS_TEMPDBS 配置参数设置的值将用于临时数据库空间，直到服务器重新启动。必须确保 SD 辅助服务器上为 SDS_TEMPDBS 配置参数指定的值与主服务器上指定的值不同。重新启动 SD 辅助服务器之后，将使用 DBSPACETEMP 配置参数。

6.3.2 执行基本管理任务

这些主题包含有关系统正在运行 HDR 时如何执行数据库服务器管理任务的指示信息。

更改数据库服务器配置参数

一些配置参数必须在复制对中的两个数据库服务器上均设置为相同的值（如集群的数据库服务器配置需求中所列）。其他 GBase 8s 配置参数可设置为不同的值。

要对 onconfig 文件进行更改，请执行以下操作：

1. 使用 gadmin -k 选项使每个数据库服务器脱机。

   如果 DRAUTO 设置为 RETAIN_TYPE 或 REVERSE_TYPE，那么可以更加轻松地先使辅助数据库服务器脱机。

2. 更改每个数据库服务器上的参数。
3. 从已脱机的最后一个数据库服务器开始，将每个数据库服务器恢复联机。

   例如，如果您最后一次将辅助数据库服务器变为脱机，那么首先将辅助数据库服务器恢复联机。表 1 列出了将主数据库服务器和辅助数据库服务器恢复联机的过程。

如果配置参数无需在复制对中的每个数据库服务器上具有相同值，那么可以分别在主数据库服务器或辅助数据库服务器上更改该值。

备份存储空间和逻辑日志文件

当您使用 HDR 时，您必须仅在主数据库服务器上备份逻辑日志文件和存储空间。但要准备在数据库服务器的类型更改为标准类型的情况下在辅助数据库服务器上执行存储空间和逻辑日志备份。

您必须在两个数据库服务器上使用相同的备份与复原工具。

所用的块大小和磁带大小（用于存储空间备份和逻辑日志备份）必须在主数据库服务器和辅助数据库服务器上均相同。

您可以使用 gtape 将磁带大小设置为 0 以自动使用磁带的全部物理容量。

更改数据库的记录方式

当您正在使用 HDR 时，您无法打开主数据库服务器上数据库的事务日志记录。您可以为数据库关闭日志记录；但是，随后对该数据库所作的更改将不会复制到辅助数据库服务器上。

要打开数据库日志记录，请执行以下操作：

1. 要关闭 HDR，请关闭辅助数据库服务器。
2. 打开数据库日志记录。

在您为数据库打开日志记录后，如果您启动数据复制但未在主数据库服务器上执行 0 级备份，然后在辅助数据库服务器上进行复原，那么主数据库服务器和辅助数据库服务器上的数据库可能有不同数据。 这种情况可能会导致数据复制问题。

3. 在主数据库服务器上执行 0 级备份并在辅助数据库服务器上复原。

在首次启动 HDR 中描述了该过程。

添加和删除块与存储空间

您仅可从主数据库服务器执行磁盘布局操作（如添加或删除块和数据库空间）。在辅助数据库服务器上复制该操作。 这种安排将确保复制对中两个数据库服务器上的磁盘布局保持一致。

块的目录路径名或实际文件在您创建块之前就必须存在。请确保在主数据库服务器上创建块之前路径名和偏移量（后者如果适用）存在于辅助数据库服务器上，否则数据库服务器会关闭数据复制。

提示： 在具有一个或多个 SD 辅助服务器的高可用性集群的主服务器上添加数据库空间时，某个 SD 辅助服务器的 online.log 可能显示以下错误：“Assert Failed: Page Check Error”。如果发生该情况，请关闭并重新启动该 SD 辅助服务器。在重新启动该 SD 辅助服务器后，新添加的数据库空间将可用并完全发挥作用。

重命名块

如果为块路径名使用了符号链接，那么可以在 HDR 运行时对块重命名。有关重命名块的指示信息，请参阅《GBase 8s 备份与复原指南》。

如果没有为块路径名使用符号链接，那么在重命名块时，必须使两个数据库服务器保持脱机，直到完成数据库服务器冷复原。

要在发生故障的 HDR 服务器上重命名块，请执行以下操作：

1. 将未损坏的服务器的方式更改为标准方式。
2. 对标准服务器进行 0 级备份。
3. 关闭标准服务器。
4. 在从新的 0 级归档进行冷复原期间重命名标准服务器上的块（有关指示信息，请参阅《GBase 8s 备份与复原指南》）。
5. 启动标准服务器。
6. 对标准服务器再次进行 0 级归档。请确保服务器处于标准方式。
7. 用新的 0 级备份复原发生故障的服务器并重新建立 HDR 对。

在辅助数据库服务器上保存块状态

对于数据复制对，如果在辅助数据库服务器上更改块的状态（脱机、联机），并且该辅助服务器在检查点完成前重新启动，那么不保存更新的块状态。

要确保新的块状态清空到辅助数据库服务器上的保留页，可在主数据库服务器上强制执行检查点并验证检查点也已在辅助数据库服务器上完成。即使辅助数据库服务器重新启动，此时也将保留新的块状态。

如果辅助数据库服务器上的主块脱机，您可以从镜像块恢复该主块。

要从镜像块恢复主块，请执行以下操作：

1. 在辅助数据库服务器上运行 gspaces -s 以使主块联机。

   您也可使用 Server Administrator使主块联机。

2. 在主数据库服务器上运行 gadmin -c 以强制执行检查点。
3. 在主数据库服务器上运行 gadmin -m 以验证检查点已实际执行。
4. 在辅助据库服务器上运行 gadmin -m 以验证检查点也已在辅助数据库服务器上完成。

完成这些步骤后，重新启动辅助数据库服务器时，主块将联机。

使用和更改块的镜像

在您可以添加镜像块之前，必须已经在主数据库服务器和辅助数据库服务器上均分配了该块的磁盘空间。如果您希望对复制对中一个数据库服务器上的数据库空间制作镜像，您必须在两个数据库服务器上均为该数据库空间创建镜像块。有关分配磁盘空间的一般信息，请参阅分配磁盘空间。

不将 MIRROR 配置参数设置为 1 除非您正在使用镜像。

您仅可从主数据库服务器执行磁盘布局操作。因此，您可以仅从主数据库服务器添加或删除镜像块。您向主数据库服务器添加的或从中删除的镜像块也添加到辅助数据库服务器或从辅助数据库服务器中删除。您必须对辅助数据库服务器上新添加的镜像块执行镜像恢复。有关更多信息，请参阅恢复镜像块。

当您从主数据库服务器删除块时，GBase 8s 自动在辅助数据库服务器上删除相应块。这对主块和镜像块均适用。

当您为主数据库服务器上的数据库空间关闭镜像时，GBase 8s 不会为辅助数据库服务器上的相应数据库空间关闭镜像。要为辅助数据库服务器上的数据库空间关闭镜像而不依赖主服务器，请使用 gspaces -r。有关关闭镜像的更多信息，请参阅结束镜像过程。

您可以在主数据库服务器或辅助数据库服务器上将镜像块脱机或恢复镜像块。这些进程对于 HDR 是透明的。

管理物理日志

物理日志的大小必须在两个数据库服务器上都相同。如果您更改了主数据库服务器上物理日志的大小和位置，那么此更改将复制到辅助数据库服务器。 辅助数据库服务器上的 ONCONFIG 值将自动更新。

有关更改物理日志的大小和位置的信息，请参阅管理物理日志。

管理逻辑日志

逻辑日志的大小必须在两个数据库服务器上都相同。按管理逻辑日志文件中所述，您可以用 glogadmin 实用程序添加或删除逻辑日志文件。GBase 8s 将在辅助数据库服务器上复制此更改；但是辅助数据库服务器上的 LOGFILES 参数将不会进行更新。因此，当您从主数据库服务器发出 glogadmin 命令后，必须将 LOGFILES 参数手动更改为辅助数据库服务器上的相应值。最后，为使更改生效，您必须对主数据库服务器上的根数据库空间执行 0 级备份。

如果您向主数据库服务器添加逻辑日志文件，该文件在您执行 0 级备份时就立即可供使用并标记为 F。辅助数据库服务器上新的逻辑日志文件仍然标志为 A。但是，这种情况不会阻止辅助数据库服务器写入文件。

管理虚拟处理器

虚拟处理器的数目对数据复制没有影响。您可以分别配置和调整复制对中的每个数据库服务器。

管理共享内存

如果对一台数据库服务器上的共享内存 ONCONFIG 参数进行了更改，那么必须同时对另一数据库服务器上的共享内存 ONCONFIG 参数进行相同更改。有关进行该更改的过程，请参阅更改数据库服务器配置参数。

设置来自主服务器的响应的等待时间

可使用 IFX_SMX_TIMEOUT 和 IFX_SMX_TIMEOUT_RETRY 这两个环境变量来处理高可用性复制 (HDR)、远程独立 (RS) 或共享磁盘 (SD) 辅助服务器等待来自主服务器的响应的时间量。

使用：

- IFX_SMX_TIMEOUT 环境变量可以指定辅助服务器等待来自主服务器的消息的最大秒数。
- IFX_SMX_TIMEOUT_RETRY 环境变量以指定在未接收到来自主服务器的响应时，辅助服务器重复 IFX_SMX_TIMEOUT 环境变量指定的等待周期的次数。

设置服务器之间 SMX 活动的等待时间

您可以设置 SMX_PING_INTERVAL 和 SMX_PING_RETRY 配置参数以调整高可用性集群中辅助服务器等待主服务器活动的时间间隔。

使用 SMX_PING_INTERVAL 配置参数可指定超时时间间隔秒数，辅助服务器在此期间通过“服务器多路复用器组”(SMX) 连接等待主服务器的活动。

使用 SMX_PING_RETRY 配置组参数可指定在未收到来自主服务器的响应时，辅助服务器重复 SMX_PING_INTERVAL 配置参数指定的超时时间间隔的最大次数。 如果达到最大

次数而无响应，那么辅助服务器将在 online.log 中输出错误消息并关闭“服务器多路复用器组”(SMX) 连接。

将索引复制到 HDR 辅助数据库服务器

如果启用索引页日志记录，索引会自动复制到 HDR 辅助数据库服务器上（请参阅索引页日志记录）。 如果禁用索引页日志记录，并且 HDR 辅助数据库服务器上的索引已损坏而必须重建，那么您可以使用以下两种方法之一：

- 手动地将索引从主服务器复制到辅助服务器中。
- 使辅助服务器自动复制索引（如果您使辅助服务器能够执行此操作）。

要使辅助数据库服务器能够自动复制索引，请执行以下操作之一：

- 将 gadmin -d idxauto 设置为 on 。
- 将 DRIDXAUTO 配置参数的值设置为 1 。

在您设置这些值的任意一个之后，当辅助数据库服务器上的某个线程检测到损坏的索引时，索引将自动复制到辅助数据库服务器中。重新启动索引复制可能需要 DRTIMEOUT 配置参数中指定的时间（以秒计）。

有时，您可能希望手动复制索引，例如：当由于表被锁定您因此希望延迟索引修复时。如果您希望能够在 HDR 辅助服务器上手动复制索引，那么关闭自动复制功能。

要关闭自动索引复制功能，请执行以下操作之一：

- 将 gadmin -d idxauto 设置为 off 。
- 将 DRIDXAUTO 配置参数设置为 0 。

如果 gadmin -d idxauto 设置为 off 或 DRIDXAUTO 设置为 0，并且辅助服务器检测到损坏的索引，那么可以通过发出以下格式的 gadmin -d index 命令在 HDR 辅助服务器上手动复制索引：gadmin -d index database:[ownername].table#index

例如：gadmin -d index cash_db:user_dx.table_12#index_z

如果分段索引具有一个损坏分段，那么 gadmin -d idxauto 选项只会转移单个受影响的分段，而 gadmin -d index 选项将会转移整个索引。

重要：当打开或关闭自动索引复制功能时，您可以 gadmin 命令或 DRIDXAUTO 配置参数。如果使用 gadmin 命令，那么无需停止并重新启动数据库服务器。当您使用 DRIDXAUTO 参数时，数据库服务器将使用您指定的设置重新启动。gadmin 命令不会更改 DRIDXAUTO 值。如果您使用 gadmin 命令，那么您必须手动更改 DRIDXAUTO 的值。

辅助服务器生成的 online.log 文件包含有关所有已复制索引的信息。

加密 HDR 数据库服务器之间的数据通信

要使用已加密的 HDR 连接与通信支持模块 (CSM) 客户机/服务器加密一起使用，必须配置两个网络端口：

- 其中一个网络端口必须配置用于 HDR。

- 另一网络端口必须配置用于 CSM 客户机/服务器连接。

可以使用 GBase 8s 服务器加密选项来加密 HDR 对的数据库服务器之间的数据流量。当您想要确保安全的数据传输时，请如此操作。

当您启用加密后，HDR 对中的首个数据库服务器将在数据发送到对中另一服务器之前加密数据。接收数据的服务器收到数据即开始解密数据。

对于高可用性集群环境中的可更新辅助服务器，从可更新辅助服务器到主服务器的加密需要 SMX 加密。要加密从可更新辅助服务器发送到主服务器的数据，请在辅助服务器上设置 ENCRYPT_SMX 配置参数。请参阅启用 SMX 加密以获取更多信息。

限制：您无法在网络连接上启动 HDR，该连接是配置以使用客户机/服务器连接的 CSM 加密的。

可能需要附加的缓冲区或较大的缓冲区以满足已加密数据的大小。

要加密两台 HDR 数据库服务器之间的数据流量,请执行以下操作：

1. 在 HDR 对中的首台服务器上设置以下配置参数。
   - ENCRYPT_HDR，该参数启用或禁用 HDR 加密
   - ENCRYPT_CIPHERS，它指定用于加密的密码和方式
   - ENCRYPT_MAC，它控制消息认证代码 (MAC) 生成的级别
   - ENCRYPT_MACFILE，它指定了 MAC 密钥文件的完整路径名列表
   - ENCRYPT_SWITCH，它指定自动重新协商密码和密钥之间的分钟数

要更改这些参数，请遵循更改数据库服务器配置参数中的指示信息。

2. 在辅助服务器上设置加密配置参数。

   ENCRYPT_HDR、ENCRYPT_CIPHERS、ENCRYPT_MAC 和 ENCRYPT_SWITCH 配置参数必须具有主服务器上对应配置参数的相同值。ENCRYPT_MACFILE 配置参数可以在每台服务器上具有不同的值，但是文件中必须包含相同的 MAC 密钥。

例如，指定以下有关 HDR 对中的主服务器和辅助服务器的信息：

配置参数	主服务器上的样本设置	辅助服务器上的样本设置
ENCRYPT_HDR	1	1
ENCRYPT_CIPHERS	all	all
ENCRYPT_MAC	medium	medium
ENCRYPT_MACFI	/vobs/tristan/sqldist/etc/mac1.	vobs/tristan/sqldist/etc/mac2.

配置参数	主服务器上的样本设置	辅助服务器上的样本设置
LE	dat	dat
ENCRYPT_SWITCH	60,60	60,60

在本例中，ENCRYPT_MACFILE 路径中主服务器的文件名是 mac1.dat，而 ENCRYPT_MACFILE 路径中辅助服务器的文件名是 mac2.dat。否则，所有的设置在两台服务器上是一样的。

仅使用这些配置参数来指定 HDR 的加密信息。不能通过使用 sqlhosts 文件中的 CSM 选项来指定 HDR 加密信息。

HDR 加密与 Enterprise Replication 加密一起使用，并控制启用或禁用 Enterprise Replication 加密。 当 HDR 和 Enterprise Replication 相互联合使用时，可共享同样的 ENCRYPT_CIPHER、ENCRYPT_MAC、ENCRYPT_MACFILE 和 ENCRYPT_SWITCH 配置参数。

有关这些配置参数的更多信息，请参阅《GBase 8s 管理员参考》 。

调整 HDR 服务器对中的 LRU 清空和自动调节

当为 HDR 配置了服务器，由辅助服务器触发的检查点是非分块的。检查点的这些类型将会陆续发生。 如果辅助服务器触发了非阻塞检查点，那么将阻塞主服务器上的事务，以确保辅助服务器的完整性不受影响。如果系统上的辅助服务器触发非分块检查点，那么必须将主服务器上的 LRU 清空调整得更加积极以减少事务分块。

要增加 LRU 清空，请减少 BUFFERPOOL 配置参数中 lru_min_dirty 和 lru_max_dirty 的值。

可以独立地在每个 HDR 节点上打开或关闭自动 LRU 调整。每台 HDR 数据库服务器上的设置可以不同。有关关闭自动 LRU 调整的信息，请参阅打开或关闭自动 LRU 调整。

快速克隆主服务器

可使用 gclone 实用程序来执行一步式服务器安装，从而使用最低设置或配置来克隆高可用性集群中的主服务器。

使用 gclone 实用程序可以创建独立的 GBase 8s 服务器，或创建 RS 辅助服务器。通过使用 gclone 实用程序，数据库管理员可快速、轻松、安全地通过正在运行的 GBase 8s 实例创建克隆服务器，而无需备份源服务器上的数据以及将其传输并复原到克隆服务器上。使用 gclone 实用程序可同时启动备份与复原进程，因此无需对磁盘或磁带执行数据读写。

通过使用加密的服务器多路复用器组 (SMX) 连接，可以将数据通过网络从源服务器传输到目标服务器。

通过从脚本调用 gclone 实用程序，可以自动创建克隆实例。

创建主服务器的克隆

可使用 gclone 实用程序创建主服务器的克隆。

下面是创建服务器克隆的一般步骤：

1. 在目标服务器上设置以下环境变量：
   - GBASEDBTDIR
   - GBASEDBTSERVER
   - ONCONFIG
   - GBASEDBTSQLHOSTS
2. 在目标服务器上，创建源服务器上包含的所有块。执行以下步骤来创建这些块：

   a. 在源服务器上，运行 gstat -d 命令以显示块列表：

   ```
   gstat -d
   ```

   b. 在目标服务器上，以用户 gbasedbt 的身份登录，并使用命令 touch、chown 和 chmod 创建这些块。例如，要创建名为 /usr/gbasedbt/chunks/rootdbs.chunk 的块，请执行以下步骤：

   ```
   $ su gbasedbt
   Password:
   $ touch /usr/gbasedbt/chunks/rootdbs.chunk
   $ chown gbasedbt:gbasedbt /usr/gbasedbt/chunks/rootdbs.chunk
   $ chmod 660 /usr/gbasedbt/chunks/rootdbs.chunk
   ```

   c. 对 gstat -d 命令报告的每个块重复执行上一步中的所有命令。
3. 保持以用户 gbasedbt 的身份登录的状态下，在启动克隆服务器的目标系统上运行带适当参数的 gclone 实用程序。
4. 可以选择在目标服务器上创建 onconfig 和 sqlhosts 文件。

使用以下步骤，使用源服务器上的 ONCONFIG 和 GBASEDBTSQLHOSTS 配置文件来克隆服务器。

在本示例中，省略了 -L 选项，使 gclone 实用程序从源服务器检索必要的配置信息。这些配置文件用作创建目标服务器配置的模板。通过让 gclone 实用程序创建配置文件，可以节约时间，并减少在配置文件中产生错误的可能性。

在本示例中，假定源服务器 (Amsterdam) 将 sqlhosts 文件配置如下：

```
#Server     Protocol HostName          Service Group
Amsterdam onsoctcp 192.168.0.1 123       -
```

还必须具有目标服务器的名称、IP 地址和端口号。本示例使用了以下信息：

- 源服务器名称：Amsterdam
- 源 IP 地址：192.168.0.1
- 源端口：123
- 目标服务器名称：Berlin
- 目标 IP 地址：192.168.0.2
- 目标端口：456

1. 在目标服务器上，以用户 gbasedbt 的身份登录，并使用 touch、chown 和 chmod 命令创建块、更改其所有者并更改其许可权。块路径必须与块在源服务器上的路径匹配。
2. 以用户 gbasedbt 的身份运行 gclone 实用程序：

```
gclone -T -S Amsterdam -I 192.168.0.1 -P 123 -t Berlin
          -i 192.168.0.2 -p 456
```

gclone 实用程序将修改源服务器上的 sqlhosts 文件，并在新目标服务器上创建该文件的副本。目标服务器上的 sqlhosts 文件与源服务器上的相同：

```
#Server     Protocol HostName          Service Group
Amsterdam onsoctcp 192.168.0.1 123       -
Berlin      onsoctcp 192.168.0.2 456
```

使用 -L (– useLocal) 选项在远程主机上创建服务器的克隆：-L 选项用于将源 onconfig 文件配置信息与目标 onconfig 文件合并。此选项还将把源 sqlhosts 文件复制到目标服务器。

- 源服务器名称：Amsterdam
- 源 IP 地址：192.168.0.1
- 源端口：123
- 目标服务器名称：Berlin
- 目标 IP 地址：192.168.0.2
- 目标端口：456

1. 在目标计算机上创建 onconfig 和 sqlhosts 文件并设置环境变量。
2. 在目标服务器上，以用户 gbasedbt 的身份登录，并使用 touch、chown 和 chmod 命令创建块、更改其所有者并更改其许可权。块路径必须与块在源服务器上的路径匹配。
3. 以用户 gbasedbt 的身份运行 gclone 实用程序：

```
gclone -T -L -S Amsterdam -I 192.168.0.1 -P 123 -t Berlin
          -i 192.168.0.2 -p 456
```

要向现有高可用性集群添加 RS 辅助服务器，请执行以下操作：

- 源服务器名称：Amsterdam
- 源 IP 地址：192.168.0.1
- 源端口：123
- 目标服务器名称：Berlin
- 目标 IP 地址：192.168.0.2
- 目标端口：456

1. 在目标计算机上创建 onconfig 和 sqlhosts 文件并设置环境变量。
2. 在目标服务器上，以用户 gbasedbt 的身份登录，并使用 touch、chown 和 chmod 命令创建块、更改其所有者并更改其许可权。块路径必须与块在源服务器上的路径匹配。
3. 在源服务器上（如有必要），通过以用户 gbasedbt 的身份运行以下命令来启用 LOG_INDEX_BUILDS 配置参数：

```
gadmin -wf LOG_INDEX_BUILDS=1
```

4. 在源服务器上，以用户 gbasedbt 的身份运行以下命令，以将目标服务器添加为 RS 辅助服务器：

```
gadmin -d add RSS Berlin
```

5. 以用户 gbasedbt 的身份运行 gclone 实用程序：

```
gclone -T -L -S Amsterdam -I 192.168.0.1 -P 123 -t Berlin
          -i 192.168.0.2 -p 456 -s medium -d RSS
```

辅助服务器上的数据库更新

可以启用与辅助服务器相连的应用程序来更新数据库数据。如果在辅助服务器上启用写操作，那么会将 DELETE、INSERT、MERGE 和 UPDATE 操作传播到主服务器。

使用 UPDATABLE_SECONDARY 配置参数可控制辅助服务器是否可更新数据，并可配置更新操作使用的连接数。

辅助服务器上同时支持数据定义语言 (DDL) 语句和数据操作语言 (DML) 语句。

所有可更新的辅助服务器上都支持 **dbimport** 实用程序。

辅助服务器上不支持 **dbimport** 实用程序。

仅当满足以下条件时，远程独立 (RS) 辅助服务器上才支持 **dbexport**：

- STOP_APPLY 配置参数设置为非零的有效值。
- UPDATABLE_SECONDARY 配置参数设置为非零的有效值。
- USELASTCOMMITTED 配置参数设置为 COMMITTED READ、DIRTY READ 或 ALL。

USELASTCOMMITTED 会话环境设置可覆盖 USELASTCOMMITTED 配置参数设置。

如果配置了 UPDATABLE_SECONDARY 配置参数和 STOP_APPLY 配置参数，那么所有只读辅助服务器上都支持 dbschema 实用程序。

所有可更新的辅助服务器上都支持 dbschema 实用程序。

只读辅助服务器上也支持 dbschema 实用程序。但是，dbschema 实用程序在这些服务器上运行时，会显示一条警告消息。

使用 DDL 或 DML 的大多数应用程序可在高可用性集群中的任一辅助服务器上运行；但是，不支持以下 DDL 语句：

- CREATE DATABASE（无日志记录）
- CREATE EXTERNAL TABLE
- CREATE RAW TABLE
- CREATE TEMP TABLE（有日志记录）
- CREATE XADATASOURCE
- CREATE XADATASOURCE TYPE
- DROP XADATASOURCE
- DROP XADATASOURCE TYPE
- UPDATE STATISTICS

在集群环境中，可更新的辅助服务器上不支持 SET CONSTRAINTS、SET INDEXES 和 SET TRIGGERS 语句。 对于针对辅助服务器数据库中表对象的 UPDATE 操作，不会重定向 SET Database Object Mode 语句指定的任何会话层索引、触发器或约束方式。

仅当辅助服务器映像与主服务器映像匹配时，客户机应用程序才可以在辅助服务器上插入、更新和删除行。 支持以下数据类型：

- BIGINT
- BIGSERIAL
- BLOB
- BOOLEAN
- BYTE（存储在表中）
- CHAR
- CLOB
- DATE
- DECIMAL
- DATETIME
- FLOAT
- INT
- INT8
- INTERVAL

- MONEY
- NCHAR
- NVCHAR
- SERIAL
- SERIAL8
- SMALLFLOAT
- SMALLINT
- TEXT（存储在表中）
- VARCHAR

不支持存储在 BLOB 空间中的 BYTE 和 TEXT 数据类型，因为不能复制 BLOB 空间数据。

如果下列数据类型不接收指向另一个分区的指针引用，那么也支持它们：

- COLLECTION
- LIST
- LVARCHAR
- MULTISET
- ROW
- SET
- UDTVAR

主服务器映像与辅助服务器映像之间的任何差异都会导致 SQL 错误并回滚所有更改。

您不能在 HDR 辅助服务器、远程独立 (RS) 辅助服务器或共享磁盘 (SD) 辅助服务器上使用以下实用程序：

- archecker
- dbload
- High-Performance Loader (HPL)
- gdblogmode
- gload
- ON-Monitor
- glogadmin
- onperf
- onsnmp
- gspaces
- gunload

此外，不能在 HDR 辅助服务器或共享磁盘 (SD) 辅助服务器上使用 dbexport 实用程序。在辅助服务器中，仅远程独立辅助 (RS) 服务器支持 dbexport 执行的写操作，并且只能与

上述 STOP_APPLY、UPDATABLE_SECONDARY 和 USELASTCOMMITTED 配置参数设置一起使用。

为更新配置的辅助服务器不支持字节范围锁定。 辅助服务器上的字节范围锁不能提升为全对象锁。

复制智能大对象

使用可更新的辅助服务器时，可能会收到下面一条或多条错误消息：

- 12014
- 12015
- 12233

这些错误通常表明智能大对象文件描述符存在问题。以下任一条件均可导致这些错误：

- 落实事务之前，将智能大对象标识传递到其他事务或进程。因为所有对象（包括智能大对象）在落实事务之前都不会落实，所以不允许其他事务使用智能大对象。特别是脏读取可以访问锁定的智能大对象。
- 智能大对象在打开之后未关闭。在事务结束时，必须关闭辅助服务器上的所有智能大对象，尤其是已创建并随后回滚了事务的智能大对象。如果将智能大对象文件描述符保持打开状态，将导致在终止会话之前，内存始终保持已分配状态。
- 另一个进程删除了主服务器上的智能大对象。共享锁定不会自动从辅助服务器传播到主服务器，因此其他辅助服务器可能会访问主服务器上实际已删除的智能大对象。辅助服务器上重放包含该删除操作的日志记录或主服务器更新辅助服务器之前，这些访问将一直有效。

处理脏读取信息时，可能还会再返回三个错误代码：

- -126 (ISAM error: bad row id)
- -244 (SQL error: Could not do a physical-order read to fetch next row)
- -937

如果收到以上任一代码，请重试查询。

辅助服务器上的 LOCK TABLE 语句行为

可以从高可用性集群中的可更新辅助服务器设置对表的互斥锁定。对于从辅助服务器请求的互斥方式锁定，会话可读取表，但不能更新表。 此行为类似辅助服务器上的共享访问方式；即，如果一个会话以给定表上具有互斥锁定，那么其他任何会话均不能获取该表上的共享或互斥锁定。

辅助服务器上的隔离级别

所有类型的辅助服务器上均支持以下语句：

```
Set isolation to committed read

Set isolation to committed read last committed
```

设置已落实读取隔离的辅助服务器可以在本地读取已落实的数据。 如果主服务器上的已落实数据在辅助服务器上可用并已落实，它们也可以读取这些数据。 连接至辅助服务器的应用程序会接收当前已在辅助服务器上落实的数据。 请参阅设计数据复制组客户机，以获取有关针对连接至正在运行数据复制的数据库服务器的客户机设计注意事项的更多信息。

辅助服务器上的缺省隔离级别是 DIRTY READ；然而，设置显式隔离级别将启用正确的隔离级别：DIRTY READ、COMMITTED READ 或 COMMITTED READ LAST COMMITTED。

不支持 REPEATABLE READ 和 CURSOR STABILITY 隔离级别。 在忽略 CURSOR STABILITY 和 REPEATABLE READ 级别的情况下使用 SET ISOLATION 语句。

启动辅助服务器后，只有在启动检查点打开的所有事务已落实或回滚时，客户机应用程序才与服务器连接。

如果禁用了 UPDATABLE_SECONDARY 配置参数（通过取消设置或设置为零），辅助数据复制服务器将为只读。在这种情况下，辅助服务器上只能使用 DIRTY READ 或 READ UNCOMMITTED 事务隔离级别。

如果启用了 UPDATABLE_SECONDARY 参数（通过将其设置为大于零的有效连接数），辅助数据复制服务器可支持 COMMITTED READ、COMMITTED READ LAST COMMITTED 或 COMMITTED READ 事务隔离级别，或 USELASTCOMMITTED 会话环境变量。只有 SQL 的 DML 语句（DELETE、INSERT、UPDATE 和 MERGE 语句）以及 **dbexport** 实用程序可支持对可更新辅助服务器执行写操作。（除 UPDATABLE_SECONDARY 之外，还必须设置 STOP_APPLY 和 USELASTCOMMITTED 配置参数，才能在辅助数据复制服务器上启用 dbexport 执行的写操作。）

使用 gstat -g ses 或 gstat -g sql 可查看隔离级别设置。请参阅《GBase 8s 管理员参考》 以获取更多信息。

设置锁定方式

在辅助服务器上发出 SET LOCK MODE TO WAIT 或 SET LOCK MODE TO WAIT n 语句，像在主服务器上一样为该会话设置锁定等待超时值。 当执行来自辅助服务器的更新时，在主服务器上为当前会话创建的代理线程将使用 SET LOCK MODE 设置的值。如果 SET LOCK MODE 的值大于 DEADLOCK_TIMEOUT 的 ONCONFIG 参数值，就使用 DEADLOCK_TIMEOUT 的值。

高可用性集群辅助服务器上的瞬态类型

不论辅助服务器是只读还是可更新，都可以在高可用性集群辅助服务器上使用瞬态未命名复杂数据类型（ROW、SET、LIST 和 MULTISET）。辅助服务器上支持以下使用瞬态类型的操作类型：

- 使用瞬态类型的 SQL 查询

- 使用派生表、集合子查询以及 XML 功能的 SQL 查询（这些语句隐式地使用瞬态类型）
- 由使用瞬态类型的 CREATE TEMP 语句创建的临时表

请参阅《GBase 8s SQL 指南：参考》和《GBase 8s SQL 指南：语法》以获取有关复合数据类型的信息。

行版本控制

使用行版本控制可确定是否已更改了行并检测冲突。启用行版本控制之后，会对表的每一行进行配置以使其包含校验和及版本号。当首次插入行时，会自动生成校验和，并且版本设置为 1。每次更新该行时，版本都会加一，但校验和值保持不变。 如果删除一行并在表中重新插入另一行，那么通过行版本控制可识别出新插入的行不同于删除的行。 通过比较辅助服务器和主服务器之间的行校验和与行版本，可以检测出数据冲突。

Web 应用程序可以使用版本列以确定之前检索对象中包含的信息是否仍处于最新状态。例如，Web 应用程序可能会向客户显示待售商品。 当客户决定购买某个产品时，应用程序会检查该产品行的版本列以便确定是否有关该产品的信息发生了变化。

如果在环境中客户机应用程序可更新辅助服务器上的数据，请使用行版本控制来尽可能减少网络使用，特别是表中包含大量行的情况。否则，辅助服务器上的所有行将与主服务器上的所有行进行比较，以确定是否发生了更新。

要将行版本控制添加到现有表，可使用下列语法：

```
ALTER TABLE tablename add VERCOLS;
```

同样，您可以使用以下语法来删除表中的行版本控制：

```
ALTER TABLE tablename drop VERCOLS;
```

要创建带行版本控制的新表，可使用以下语法：

```
CREATE TABLE tablename (
    Column Name     Datatype
    Column Name     Datatype
    Column Name     Datatype
    ) with VERCOLS;
```

启用了行版本控制之后，每次更新行，ifx_row_version 都会递增 1；但是，Enterprise Replication 执行的行更新不会递增行版本。要使用 Enterprise Replication 更新服务器上的行版本，必须在复制参与者定义中包含 ifx_row_version 列。

使用高可用性集群备份和复原

大多数备份与复原操作都不能在辅助服务器上执行。

必须先在服务器上执行冷复原，然后才能将该服务器建立为 HDR（高可用性数据复制）或 RS（远程独立）辅助服务器。

将服务器设置为 HDR 或 RSS 辅助服务器之后，只能执行以下备份与复原操作：

- 设置高可用性集群时，可在 HDR 或 RS 辅助服务器上执行逻辑复原。
- 可在 RS 辅助服务器上执行外部备份。有关更多信息，请参阅《GBase 8s 备份与复原指南》。

冷复原主服务器期间，必须关闭 SD 辅助服务器，但是 SD 辅助服务器在关闭并重新启动之后，可以在热复原期间联机。

更改数据库服务器方式

如果更改高可用性集群中数据库服务器的方式，复制将停止。

要更改数据库服务器方式，请使用 gadmin 实用程序。

下表总结了更改主数据库服务器方式产生的影响。

表 1. 主数据库服务器上的方式更改

在主服务器上	在辅助服务器上	重新启动 HDR
脱机的任何方式 (gadmin -k)	辅助服务器显示：DR: Receive error。 HDR 关闭。 方式保持为只读。 如果 DRAUTO 设置为 0 (OFF)，那么方式将保持为只读。 如果 DRAUTO 设置为 1 (RETAIN_TYPE)，那么辅助服务器将切换为标准类型并且可以接受更新。（如果 DRAUTO 设置为 2 (REVERSE_TYPE)，那么当旧的主服务器发生故障时，在连接结束时，辅助数据库服务器将立即成为主数据库服务器。）	将其作为主服务器的故障一样来处理。根据在主数据库服务器脱机时您对辅助数据库服务器所进行的操作不同，可能有两种不同的情况： 请参阅以下部分以获取信息： ● 辅助服务器未更改为主服务器 ● 辅助服务器已自动更改为主服务器 请参阅主服务器发生故障时重新启动。

在主服务器上	在辅助服务器上	重新启动 HDR
联机、静默或管理 (gadmin -s / gadmin -u) (gadmin -j)	辅助服务器不会接收到错误。 HDR 保持打开。 方式保持为只读。	在主服务器上使用 gadmin -m。

下表总结了更改辅助数据库服务器方式产生的影响。

表 2. 辅助数据库服务器上的方式更改

在辅助服务器上	在主服务器上	重新启动 HDR
只读脱机 (gadmin -k)	主服务器显示：DR: Receive error。 HDR 关闭。	将其作为辅助服务器的故障一样来处理。请遵循辅助服务器发生故障时重新启动 HDR 或 RS 集群中的过程。

管理方式在 HDR 辅助数据库服务器上的运行方式与其在主数据库服务器上运行的方式相同。

更改数据库服务器类型

您可以更改主数据库服务器或辅助数据库服务器的类型。

您可以更改主数据库服务器或辅助数据库服务器的类型。

仅当在辅助数据库服务器上关闭 HDR 时，您才能将数据库服务器类型从辅助更改为标准。当数据复制到主数据库服务器的连接断开或辅助数据库服务器上的数据复制失败时，关闭 HDR。当您使标准数据库服务器脱机并随后将其恢复联机时，它不会尝试连接至复制对中的另一数据库服务器。

使用以下命令切换类型：

- hdrmksec.[sh|bat] 和 hdrmkpri.[sh|bat] 脚本

要使用 hdrmkpri 和 hdrmksec 脚本切换数据库服务器类型，请执行以下操作：

1. 关闭主数据库服务器 (ServerA)：gadmin -ky
2. 在辅助数据库服务器 (ServerB) 联机时，运行 hdrmkpri.sh 脚本（在 UNIX™ 上）。现在 ServerB 是主数据库服务器。
3. 对于 ServerA，运行 hdrmksec.sh 脚本（在 UNIX 上）。现在 ServerA 是辅助数据库服务器。

4. 使 ServerB（主数据库服务器）联机。

还可以使用以下命令来切换服务器类型：

1. 通过运行以下命令，将 ServerA 更改为主服务器：

```
gadmin -d make primary ServerA
```

此命令将使 ServerA 成为主服务器，并将集群中的其他任何辅助服务器重定向为指向新的主服务器。 此命令还会关闭原有 HDR 主服务器 (ServerB)，这是因为高可用性环境中只能存在一个主服务器。

2. 通过运行以下命令将 ServerB 初始化为 HDR 辅助服务器：

   - 在 UNIX 系统上：

```
$GBASEDBTDIR/bin/hdrmksec.sh ServerB
```

防止 HDR 服务器上的检查点阻塞

在 HDR 辅助服务器上，检查点处理必须等待缓冲池清仓完成。可在 HDR 辅助服务器上配置非阻塞检查点，这样在完成检查点处理之前，将把从主服务器发送的日志数据存储（也称为登台）到目录中。

可通过设置 LOG_STAGING_DIR 和 LOG_INDEX_BUILDS 配置参数，在 HDR 辅助服务器上配置非阻塞检查点。配置了非阻塞检查点之后，从主服务器发送的日志数据将登台到 LOG_STAGING_DIR 配置参数指定的目录中。HDR 辅助服务器完成检查点的处理后，将读取并应用登台区域内存储的日志数据。如果登台目录为空，HDR 辅助服务器将读取和应用从主服务器接收的日志数据。

可通过在 HDR 辅助服务器上设置 LOG_STAGING_DIR 配置参数，并同时在主服务器和 HDR 辅助服务器上设置 LOG_INDEX_BUILDS 来启用非阻塞检查点。
LOG_INDEX_BUILDS 的值在主服务器和 HDR 辅助服务器上必须相同。

HDR 辅助服务器遇到检查点时，将进入缓冲方式。在缓冲方式中，辅助服务器把来自主服务器的任何日志页数据登台到登台目录内的文件中。

HDR 辅助服务器完成检查点处理后，将进入漏出方式。 在这种方式中，HDR 辅助服务器从登台文件读取数据，同时从主服务器接收新数据。登台区域为空之后，HDR 辅助服务器将恢复正常运行。

日志记录在 HDR 服务器上的存储位置

HDR 辅助服务器将在 LOG_STAGING_DIR 指定的目录内额外创建名为 ifmxhdrstage_## 的目录，其中 ## 是 SERVERNUM 指定的实例。这些目录用于存储处理检查点期间从主服务器发送的逻辑文件。 ifmxhdrstage_## 内的文件在不再需要时将清除。

非阻塞检查点与辅助服务器更新交互

必须注意辅助服务器更新与 HDR 辅助服务器上的非阻塞检查点之间的交互。如果 HDR 辅助服务器接收到更新请求，在该 HDR 辅助服务器处理相应的日志记录之前，将不应用

更新。如果在 HDR 辅助服务器上启用了非阻塞检查点，可能会在辅助服务器上发生数据应用延迟，这是因为检查点处理导致日志文件在辅助服务器上登台。

查看 HDR 服务器上非阻塞检查点的统计信息

可使用 gstat 实用程序查看有关主服务器上和 HDR 辅助服务器上的非阻塞检查点的信息。

要查看有关登台日志的信息，请使用 gstat -g dri ckpt 命令。

有关 gstat -g ckpt 输出的示例，请参阅《GBase 8s 管理员参考》中有关 gstat 实用程序的信息。

监视 HDR 状态

监视数据库服务器的 HDR 状态以确定以下信息：

数据库服务器类型（主类型、辅助类型或标准类型）

复制对中另一数据库服务器的名称

HDR 是否打开

HDR 参数的值

命令行实用程序

每次运行 gstat 时显示的头信息均有字段指示数据库服务器正在作为主数据库服务器还是辅助数据库服务器运行。

以下示例为作为复制对中的主数据库服务器并且处于联机方式的数据库服务器显示头信息：

```
GBase 8s Database Server V8.5      -- online(Prim) -- Up 45:08:57
```

以下示例显示作为复制对中的辅助数据库服务器并且处于只读方式的数据库服务器：

```
GBase 8s Database Server V8.5      -- Read-Only (Sec) -- Up 45:08:57
```

以下示例显示不包含在 HDR 中的数据库服务器的标题。该数据库服务器的类型为标准类型。

```
GBase 8s Database Server V8.5       -- online -- Up 20:10:57
```

gstat -g dri 选项

要获得完整的 HDR 监视信息，请运行 gstat -g dri 选项。显示以下字段：

- 数据库服务器类型（主类型、辅助类型或标准类型）
- HDR 状态（打开或关闭）
- 成对的数据库服务器
- 最后一台 HDR 检查点

- HDR 配置参数的值

有关 gstat -g dri 输出的示例，请参阅《GBase 8s 管理员参考》。

gcheck -pr 选项

如果您的数据库服务器正在运行 HDR，那么保留页 PAGE_1ARCH 和 PAGE_2ARCH 将保存 HDR 用于同步主数据库服务器和辅助数据库服务器的检查点信息。以下示例中提供了相关 gcheck -pr 输出的示例。

```
Validating GBase 8s Database Server reserved pages - PAGE_1ARCH &
PAGE_2ARCH
            Using archive page PAGE_1ARCH.

    Archive Level                   0
    Real Time Archive Began          01/11/95 16:54:07
    Time Stamp Archive Began          11913
    Logical Log Unique Id           3
    Logical Log Position            b018

    DR Ckpt Logical Log Id          3
    DR Ckpt Logical Log Pos          80018
    DR Last Logical Log Id          3
    DR Last Logical Log Page         128
```

SMI 表

sysdri 表提供了有关数据库服务器的高可用性数据复制状态的信息。

在 GBase 8s 管理员参考 中有关 sysmaster 数据库的这些主题中描述的 sysdri 表包含以下各列。

type

HDR 服务器类型

state

HDR 服务器状态

name

数据库服务器名

intvl

HDR 缓冲区清空时间间隔

timeout

网络超时

lostfound

HDR 失而复得的路径名

使用 ON-Monitor 监视 HDR 状态 (UNIX™)

选择**状态** > **复制**可查看有关 HDR 的信息。

该选项与 gstat -g dri 选项显示相同的信息。

6.3.3 获取 RS 辅助服务器统计信息

使用 gstat 命令，显示有关 RS 辅助服务器的状态信息。 要显示 RS 辅助服务器的数量、有关已发送到 RS 辅助服务器的数据的信息，以及有关 RS 辅助服务器已确认接收的内容的信息，请使用 gstat -g rss 命令。

此命令具有显示有关单台服务器或多个辅助服务器的扩展信息的选项。 有关 gstat -g rss 输出的示例，请参阅《GBase 8s 管理员参考》中有关 gstat 实用程序的信息。

6.3.4 除去 RS 辅助服务器

可通过发出以下命令，从高可用性集群中除去 RS 辅助服务器：

```
gadmin -d delete RSS rss_servername
```

6.3.5 RS 辅助服务器安全性

RS 辅助服务器支持与 HDR 类似的加密规则。请参阅加密 HDR 数据库服务器之间的数据通信以获取详细信息。

有关设置和配置服务器与 RS 辅助服务器之间加密的其他信息，请参阅服务器多路复用器组 (SMX) 连接。

创建或更改 RS 辅助服务器上的密码

可以为 RS 辅助服务器创建密码，以便在建立集群时，提供主服务器与辅助服务器之间的认证。密码是可选的。密码仅在主服务器和辅助服务器首次互连时有效。

密码可防止网络中的意外实例初始化和接受来自主服务器的事务数据。 密码与 gbasedbt 用户密码无关。

使用用于指定 RS 辅助服务器名称的相同命令创建密码。每个 RS 辅助服务器的名称和密码可在主服务器的 0 级备份之前或之后定义。

要设置 RS 辅助服务器名称和密码，请在主服务器上运行 gadmin -d add RSS rss_servername password 命令。辅助服务器设置期间，在辅助服务器上运行 gadmin -d RSS rss_servername password 命令时，请包含密码。

仅当服务器未连接到高可用性环境时，您才可以更改密码。尝试更改已连接的 RS 辅助服务器的密码将返回错误。

要在连接 RS 辅助服务器之前，更改该服务器上的密码，请在主服务器上使用 gadmin -d change RSS password 命令。

示例

以下命令建立 RS 辅助服务器 ServerB 并使用密码 s#cure。

在主服务器上：

```
gadmin -d add RSS ServerB s#cure
```

在辅助服务器上：

```
gadmin -d RSS ServerB s#cure
```

以下命令将未连接到主服务器的 RS 辅助服务器 ServerB 的密码更改为 s@fest：

```
gadmin -d change RSS ServerB s@fest
```

6.3.6 在集群故障转移期间完成事务

可在高可用性集群环境中配置服务器，以便在执行了主服务器故障转移之后继续处理事务。

失败的主服务器以外的任何服务器上运行的事务将继续运行。请配置集群环境，以便：

- 在辅助服务器上运行的事务不受影响。
- 在成为主服务器的辅助服务器上运行的事务不受影响。
- 在失败的主服务器上运行的事务将终止。

目前对智能大对象、XA 事务发生故障转移之后，以及在辅助服务器上运行 DDL 语句时发生故障转移之后，均不支持继续完成事务。

如果发生了故障转移，集群中的辅助服务器将临时暂挂正在运行的用户事务，直到新主服务器开始运行。故障转移之后，辅助服务器会将保存的事务重新发送到新的主服务器。新主服务器将恢复执行来自尚存的辅助服务器的事务。

运行分布式事务（跨多个数据库服务器的事务）时，出现故障时在主服务器上运行的任何事务都将终止。

不管故障转移是自动的（使用连接管理器）还是手动的（通过指定服务器来充当新主服务器），故障转移服务器都必须是具有集群中所有尚存服务器的最高级日志重放位置的服务器。如果故障转移服务器没有最高级日志重放位置，那么集群中的所有事务都将终止并回滚。

要实现最佳性能，请使用缺省的连接管理器故障转移配置：SDS+HDR+RSS,0（请参阅连接管理）。

建议故障转移到 SD 辅助服务器，因为主服务器和 SD 辅助服务器从同一个物理磁盘读取数据。

如果故障转移服务器是 HDR 辅助服务器，那么 SD 辅助服务器将关闭。

配置服务器以使事务在故障转移后完成

可使用 FAILOVER_TX_TIMEOUT 配置参数来配置高可用性集群中的服务器，以便在故障转移之后完成事务。

FAILOVER_TX_TIMEOUT 的值指示主服务器出现故障之后，回滚事务之前服务器等待的最大秒数。请在集群中所有服务器上都设置相同的 FAILOVER_TX_TIMEOUT 值。例如，要对事务完成指定 20 秒，请将 onconfig 文件中 FAILOVER_TX_TIMEOUT 配置参数的值设置为 20。

要禁用故障转移之后完成事务，请在集群中的所有服务器上将 FAILOVER_TX_TIMEOUT 配置参数设置为 0。

6.4 连接管理

可以使用连接管理器来监视和维护客户机连接，以及将客户机连接请求定向到连接单元中的适当服务器。

oncmsm 实用程序用于启动连接管理器，该管理器根据系统管理员配置的服务级别协议来管理并定向客户机连接请求。连接管理器提供负载均衡，并将客户机连接请求定向到一个或多个连接单元。连接单元是在网格、高可用性集群、复制集或服务器集配置中排列的一个或多个数据库服务器的集合。

表 1. 连接单元的类型和描述

连接单元类型	描述
CLUSTER	高可用性集群是由主服务器和一个或多个辅助服务器构成的一组数据库服务器。集群中的服务器是同类服务器，即所有服务器均使用相同的硬件和软件配置。主服务器上的数据将复制到所有辅助数据库服务器。一个集群中至少包含一个主服务器和一个辅助服务器。集群中可以包含一个主服务器、一个 HDR 辅助服务器、零或更多个共享磁盘辅助服务器（SDS 服务器），以及零或多个远程独立辅助服务器（RSS 服务器）。
REPLSET	复制集是与 Enterprise Replication (ER) 链接的一组数据库服务器。ER 支持在按地理分布的数据库服务器上执行异步数据复

连接单元类型	描述
	制。可以使用 ER 复制整个数据库，也可以复制数据库与表的子集。使用 ER 链接的服务器可以是异类服务器；即这些服务器可能使用不同的硬件和软件配置。复制定义复制参与者和数据的复制方法，而复制集则组合多个复制，以构成可作为一个单元来一起管理的集合。域是 ER 已知的所有服务器的集合。ER 域中的节点可包含高可用性集群，从而可以有由多个集群构成的集群。
GRID	网格是 Enterprise Replication (ER) 域中的任意一组互连服务器，包括集群、复制集和服务器集。网格简化了大型数据库服务器组的管理。例如，在一台服务器上创建表时，将在网格内的所有服务器上创建该表，并且将自动同步数据。
SERVERSET	服务器集是一台或一组由第三方复制应用程序管理的服务器。这种数据库服务器必须具有相同的数据库名称和模式，以便客户机应用程序连接。连接管理器仅对服务器集提供可用性和负载均衡。

连接管理器是一个守护程序，它接受来自客户机应用程序的连接请求，然后将客户机连接到数据库服务器。连接管理器从连接单元中的每台服务器收集工作负载统计信息，并将客户机连接定向到最合适的服务器。

当连接管理器必须在多台服务器中进行选择，以便连接客户机请求时，将根据系统管理员确定的预配置策略来决定要连接到哪台服务器。 指定的策略根据服务器数据等待时间、服务器故障状态或工作负载容量来定向连接请求。

连接管理器程序配置为使用 sqlhosts 文件，配置方法与 GBase 8s 数据库服务器相同。可以配置多个连接管理器实例，以便在连接管理器发生故障时，允许从一个连接管理器实例故障转移到另一个连接管理器实例。要避免连接管理器成为单个故障点，配置多个连接管理器实例尤为重要。有关配置多个连接管理器实例的示例，请参阅用于故障转移的连接管理器冗余。

因为连接单元可能包含以网格、高可用性集群、Enterprise Replication 复制集和服务器集形式排列的多台数据库服务器，所以客户机应用程序必须可以连接到服务器的任何成员。如果有大量服务器，可能难以确定要连接到哪台服务器。此外，也很难确定哪台服务器有足够的可用资源可用于执行给定任务。最后，难以（如果并非不可能）了解服务器何时可能遇到问题。使用连接管理器可解决这些问题。

连接管理器通过将客户机应用程序定向到活动量最少的服务器来均衡工作负载。连接管理器实用程序还执行故障转移仲裁。可以配置连接管理器，以确保在高可用性集群中的主服务器发生故障时，另一台服务器自动接管该主服务器的角色。

连接管理器执行三种角色：

- 基于规则的连接重定向
- 连接单元负载均衡
- 集群故障转移

基于规则的连接重定向

应用程序连接到连接管理器，正如连接到数据库服务器。应用程序连接到连接管理器时，将在通信层执行连接重定向，而应用程序无需执行其他任何操作。

为了配置连接管理器，将启动名为 oncmsm 的守护程序，该程序使用名为服务级别协议 (SLA) 的定制重定向规则。 连接管理器配置并初始化之后，将接受来自客户机应用程序的连接请求，并根据 SLA（重定向规则）将这些请求重定向到适当的服务器。

连接单元负载均衡

连接管理器可以执行负载均衡，其中重定向基于服务级别协议中设置的配置。连接管理器连接到每个连接单元中的每台服务器，并收集有关服务器类型、未使用的工作负载容量及服务器当前状态的统计信息。 根据这些信息，连接管理器能够将客户机连接重定向到可用容量最高的服务器。

使用 SLA 定义中的 POLICY 参数可为连接单元设置负载均衡策略。对于网格和复制集，必须启用数据质量 (QOD)，才能利用 LATENCY、FAILURE 和 WORKLOAD 策略。如果未设置 QOD，或者如果未定义该策略，重定向策略将仅基于工作负载。

对于高可用性集群和服务器集，该策略仅基于工作负载。

自动故障转移

可以使用连接管理器配置自动故障转移。在高可用性集群中，如果连接管理器检测到主服务器发生故障，并且在确定超时周期期间主服务器未执行任何操作时，会将最合适的辅助服务器转换为主服务器。

使用连接管理器配置文件中的 FOC 选项可配置连接管理器故障转移参数。如果使用多个连接管理器来管理集群故障转移，可以通过在集群的主服务器上设置 HA_FOC_ORDER 配置参数来强制执行一致的故障转移策略。HA_FOC_ORDER 配置参数的值会替换连接到主服务器的每个连接管理器的配置文件中 FOC ORDER= 的值。

连接管理器监视连接单元中的服务器，并帮助客户机应用程序连接到最合适的服务器。然而，这些角色是独立的；连接管理器将客户机连接到服务器后，不会再重定向该客户机。 如果应用程序连接的数据库服务器遇到问题，那么应用程序必须再次通过连接管理器请求连接。

6.4.1 配置连接管理器的步骤

要配置连接管理器，必须设置加密的密码文件、服务级别协议和故障转移参数，并配置 sqlhosts 文件。

先决条件：

仅限 UNIX：只有用户 **gbasedbt** 才能运行 oncmsm 命令。如果为用户 **root** 或 DBSA 组的成员授予了连接到 sysadmin 数据库的特权，那么用户 root 或 DBSA 组的该成员也可运行 oncmsm。

所有操作系统： 仅对于代理方式服务级别协议 (SLA)，设置操作系统允许的最大文件描述符数（例如，在 UNIX™ 系统上通过使用 ulimit 进行设置）。

GBase 8s Client Software Development Kit (Client SDK) V3.50 或更高版本中随附了连接管理器。配置和使用连接管理器之前，必须安装 Client SDK。 请参阅《GBase 8s 客户机产品安装指南》。

要配置和启动连接管理器，请执行以下操作：

1. 创建密码文件并将其加密。
2. 设置 GBASEDBTDIR 环境变量。
3. 在要运行连接管理器的计算机上创建连接管理器配置文件。

   配置文件定义了连接管理器实例名称、服务级别协议和连接管理选项。
4. 通过编辑 sqlhosts 文件（在 UNIX 上），为主服务器和所有 SLA 创建条目。

创建要由连接管理器使用的加密密码文件

如果连接单元中的任何服务器不属于安全网络环境，那么必须创建加密密码文件，以便连接管理器可创建与集群中的每台服务器的安全连接。

要创建加密的密码文件，请执行以下操作：

1. 使用文本编辑器创建 ASCII 文本文件，其中包含集群内所有服务器的服务器名称、用户名和密码。

   对于高可用性集群和 Enterprise Replication 域，如果要通过在 sqlhosts 文件中设置 s=6 选项来使用安全端口，并且客户机应用程序使用 DBSERVERALIASES 配置参数来指定备用数据库服务器名称列表，那么必须在密码文件中为这些安全端口定义备用服务器别名。
2. 要保护加密文件，请指定密钥。

   密钥可包含任何顺序的字符或数字，但是不得包含空格。

   重要： 要在以后解密密码文件，必须提供用于加密该文件的相同密钥。
3. 通过指定密钥名称和密码文件名称，使用 onpassword 实用程序加密密码文件：

```
onpassword -k key -e ./password_file
```

修改加密的密码文件

如果在连接单元中添加或除去了服务器，更改了密码或密钥，那么您可能希望修改加密的密码文件。

要编辑加密的密码文件，请执行以下操作：

1. 通过指定密钥和密码文件名，使用 onpassword 实用程序对密码文件解密：

```
onpassword -k EncryptKey -d ./password_file
```

2. 如果必要，请使用文本编辑器编辑该文件，以便执行必需的更改。

如果必要，确定一个新密钥。该密钥可包含任何字符或数字序列，但是不能包含空格。为了在以后解密密码文件，必须提供用于加密该文件的相同密钥。

3. 通过指定密钥和密码文件名，使用 onpassword 实用程序对密码文件加密：

```
onpassword -k EncryptKey -e ./password_file
```

为连接管理器配置环境

启动连接管理器之前，GBASEDBTDIR 环境变量必须指向连接管理器的安装目录。

如果使用的是 UNIX™ C shell (csh)，请使用 setenv 命令来设置该环境变量。对于其他 shell，请使用适合该 shell 的方法。

```
setenv GBASEDBTDIR path
```

要使用除 sqlhosts 以外的文件来指定连接管理器设置，请将 GBASEDBTSQLHOSTS 环境变量设置为该文件的名称。

修改连接管理器的 sqlhosts 文件

必须修改连接管理器的 sqlhosts 文件，才能定义网络连接信息。

连接管理器和数据库服务器实例在 sqlhosts 文件中以类似方法进行配置。sqlhosts 文件中的每个条目表示一个服务级别协议 (SLA) 名称或一个数据库服务器。

连接管理器需要连接到并监视的每台服务器都必须列在连接管理器 sqlhosts 文件中。如果要监视高可用性集群，连接管理器 sqlhosts 文件必须包含主服务器和所有辅助服务器。

可以使用 GBASEDBTSQLHOSTS 环境变量来指定 sqlhosts 文件的位置。

1. 编辑将要运行连接管理器的服务器上的 sqlhosts 文件。
2. 添加一行，其中包含名称、网络类型、主服务器的主机名和服务名称，并为连接管理器需要管理的每个辅助服务器和每个 Enterprise Replication 服务器添加单独的行。
3. 为每个服务级别协议名称添加一行。

   例如，以下条目用于创建 SLA 名称 oltp 和 report：

```
#dbservername      nettype         hostname        servicename         options
```

```
oltp        onsoctcp      cmhost1 cmport1
report      onsoctcp        cmhost1 cmport3
```

客户机通过连接管理器使用 SLA 名称连接到服务器。cmhost1 表示正在运行连接管理器的服务器名称。 cmport1 和 cmport3 是客户机连接到的端口。

为高可用性集群设置 sqlhosts、连接管理器和密码文件的示例

此示例显示如何为较小的高可用性集群设置连接管理器。

您有由以下三个服务器组成的高可用性集群配置：

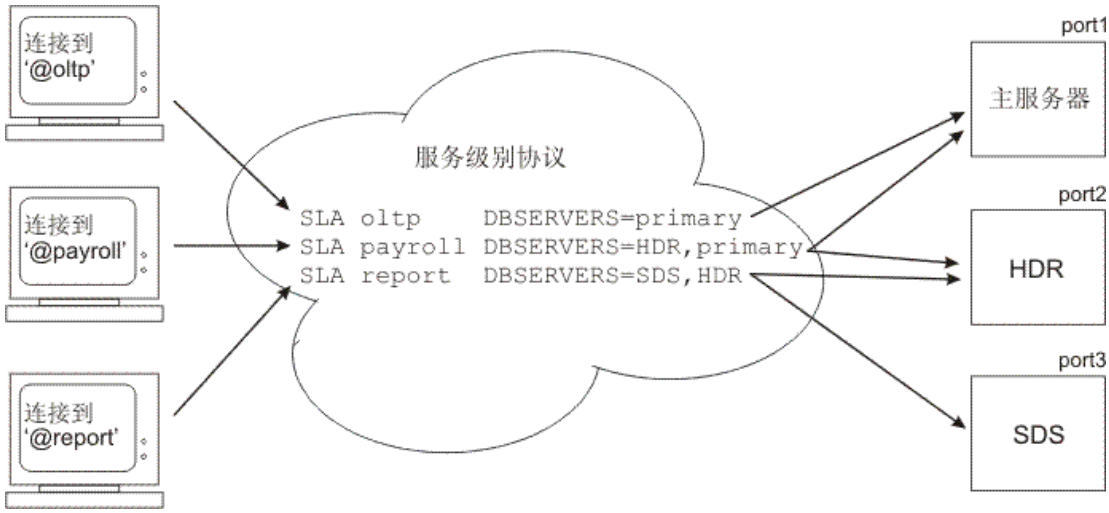
- 主服务器
- HDR 辅助服务器
- SD（共享磁盘）辅助服务器

您希望集群提供以下三种不同的服务来支持不同的应用程序：

- 联机事务处理 (OLTP)，可以在主服务器上运行
- Payroll 服务，可以在 HDR 辅助服务器或主服务器上运行
- 报告服务，可以在 HDR 辅助服务器或 SD 辅助服务器上运行

下图说明了连接管理器将客户机连接请求路由到相应服务器时所需的服务级别协议。

图： 连接管理器配置



连接管理器计算机和客户端计算机上的 sqlhosts 文件具有以下条目：

```
#dbservername      nettype        hostname        servicename         options
cluster_1 group    -               -               i=10
ifx                onsoctcp host1  port1           g=cluster_1
ifx_hdr            onsoctcp host2  port2           g=cluster_1
ifx_sds            onsoctcp host3  port3           g=cluster_1
```

```
oltp        onsoctcp       cmhost1 cmport1
report      onsoctcp cmhost1    cmport2
payroll     onsoctcp cmhost1    cmport3
```

每个数据库服务器上的 sqlhosts 文件具有以下条目：

```
#dbservername      nettype          hostname         servicename         options
ifx         onsoctcp host1       port1
ifx_hdr onsoctcp host2       port2
Ifx_sds onsoctcp host3       port3
```

连接管理器的 sqlhosts 文件定义了一个名为 cluster_1 的服务器组。该服务器组已配置为确保在主服务器发生故障且重新启动了连接管理器时，连接管理器可以重新连接到集群。该服务器组包含可用作故障转移目标的所有服务器节点（ifx、ifx_hdr 和 ifx_sds）。

要配置和启动连接管理器，请执行以下操作：

1. 创建密码文件并将其加密。对于此示例，请创建包含以下条目的文件 passwords.txt：

   ```
   ifx          ifx          gbasedbt password1
   ifx_hdr    ifx_hdr     gbasedbt password2
   ifx_sds    ifx_sds     gbasedbt password3
   ```

   运行以下命令来加密该文件：

   ```
   onpassword -k SecretKey -e ./passwords.txt
   ```

2. 设置 GBASEDBTDIR 环境变量以指向安装了 Client SDK 的目录。

3. 创建连接管理器配置文件。对于此示例，在要运行连接管理器的计算机上，创建包含以下条目的文件 cmconfig：

   ```
   NAME          cm_example
   LOG           1
   LOGFILE ${GBASEDBTDIR}/etc/cmlog

   CLUSTER cluster_1
   {
     GBASEDBTSERVER ifx
     SLA    oltp       DBSERVERS=primary
     SLA    payroll DBSERVERS=HDR,primary
     SLA    report    DBSERVERS=SDS,HDR
     FOC    ORDER=ifx_sds,ifx_hdr \
            TIMEOUT=10 \
            RETRY=1
   }
   ```

此配置文件指定了以下信息和行为：

- 连接管理器实例的名称为 cm_example。
- 已启用日志记录。
- 日志文件的名称为 cmlog，并且该日志文件位于 $GBASEDBTDIR/etc 目录中。
- 连接管理器管理的是名为 cluster1 的集群。
- 连接管理器启动时，将在 sqlhosts 文件中搜索 ifx 条目，并连接到该数据库服务器。
- CONNECT TO @oltp 连接请求将定向到主服务器。
- CONNECT TO @payroll 连接请求将定向到 HDR 辅助服务器。如果 HDR 辅助服务器不可用，CONNECT TO @payroll 连接请求将定向到主服务器。
- CONNECT TO @report 连接请求将定向到 SD 辅助服务器。如果 SD 辅助服务器不可用，CONNECT TO @report 连接请求将定向到 HDR 辅助服务器。
- 如果主服务器发生故障，ifx_sds 会成为新的主服务器。如果 ifx_sds 不可用，ifx_hdr 会成为新的主服务器。
- 连接管理器再等待 10 秒，以等待有关连接单元的主服务器故障转移处理开始的事件。
- 连接管理器在主服务器的 HA_FOC_ORDER 配置参数指定的列表中循环 1 次后，自动故障转移将终止。

某些参数和属性不包含在此配置文件中，因此连接管理器具有以下缺省行为：

- 未设置 EVENT_TIMEOUT 参数，因此连接管理器将等待主服务器事件 60 秒，然后再开始故障转移处理。
- 未设置 SECONDARY_EVENT_TIMEOUT 参数，因此连接管理器将等待辅助服务器事件 60 秒，然后再与辅助服务器断开连接。
- 未设置 SQLHOSTS 参数，因此连接管理器将依次检查 ifx 的本地 sqlhosts 文件和远程 sqlhosts 文件以查找 ifx_sds 和 ifx_hdr 的实例。
- 未设置 SLA 参数的 WORKERS 属性，因此将为每个 SLA 分配四个工作程序线程。

4. 编辑主服务器、HDR 辅助服务器和 SD 辅助服务器上的 sqlhosts 文件：

```
#dbservername      nettype         hostname        servicename         options
ifx          onsoctcp host1       port1
ifx_hdr onsoctcp host2       port2
ifx_sds onsoctcp host3       port3
```

5. 在连接管理器计算机上和每个客户端计算机上，编辑 sqlhosts 文件：

```
#dbservername      nettype         hostname        servicename         options
```

```
cluster_1 group      -          -          i=10
ifx          onsoctcp host1      port1     g=cluster_1
ifx_hdr      onsoctcp host2      port2     g=cluster_1
ifx_sds      onsoctcp host3      port3     g=cluster_1
oltp         onsoctcp      cmhost1 cmport1
report       onsoctcp cmhost1    cmport2
payroll      onsoctcp cmhost1    cmport3
```

6. 在连接管理器计算机上，通过运行 oncmsm 命令启动连接管理器：

```
oncmsm -c cmconfig
```

7. 检查日志文件以验证连接管理器是否正确启动。

启动连接管理器

可以使用 oncmsm 实用程序来启动连接管理器。

要启动连接管理器，请运行以下命令：

- UNIX™

使用以下命令在 UNIX 系统上启动连接管理器：

```
oncmsm -c configuration_file
```

6.4.2 连接管理器配置文件的格式和示例

要设置连接管理器，您必须创建连接管理器配置文件。

大多数连接管理器选项是在配置文件中指定的。配置文件由两部分组成：

- 头，其中包含连接管理器配置选项，用于指定实例名称和其他可选参数。
- 主体，其中包含一个或多个连接单元部分，用于定义连接单元的类型和名称、服务级别协议、故障转移配置和故障转移警报程序。

以下代码块显示了连接管理器配置文件的格式：

```
# *** HEADER ***
NAME connection_manager_instance_name

# Optional Parameters
MACRO name_1=server_list_1
MACRO name_2=server_list_2
MACRO name_n=server_list_n
.
.
```

```
.
LOCAL_IP ip_list
LOG value
LOGFILE path_and_filename
DEBUG value
CM_TIMEOUT seconds
EVENT_TIMEOUT seconds
SECONDARY_EVENT_TIMEOUT seconds
SQLHOSTS value

# *** BODY ***
# Connection Unit 1unit_type unit_name_1
{
    GBASEDBTSERVER server_list
    SLA sla_name_1 DBSERVERS=value \
    \
                    \ #Optional SLA Attributes
                       MODE=value \
                       USEALIASES=value \
                       POLICY=value \
                       WORKERS=number_of_threads \
                       HOST=host_name \
                       NETTYPE=network_protocol \
                       SERVICE=service_name \
                       SQLHOSTSOPT=options
    SLA sla_name_2 DBSERVERS=value ...
    SLA sla_name_n DBSERVERS=value ...
.
.
.
    #Optional Failover Parameter and Attributes
    FOC ORDER=value \
          PRIORITY=value \
          TIMEOUT=seconds \
```

```
        RETRY=attempts

    #Optional Failover Alarm Parameter
    CMALARMPROGRAM path_and_filename
}

# Connection Unit 2unit_type unit_name_2
{
    GBASEDBTSERVER server_list
    SLA sla_name_1 DBSERVERS=value ...
    SLA sla_name_2 DBSERVERS=value ...
    SLA sla_name_n DBSERVERS=value ...
    FOC ORDER=value ...
    CMALARMPROGRAM path_and_filename
}
.
.
.
# Connection Unit nunit_type unit_name_n
{
    GBASEDBTSERVER server_list
    SLA sla_name_1 DBSERVERS=value ...
    SLA sla_name_2 DBSERVERS=value ...
    SLA sla_name_n DBSERVERS=value ...
    FOC ORDER=value ...
    CMALARMPROGRAM path_and_filename
}
```

提示： 为了提高可读性，请使用反斜杠 (\) 行接续字符来断开较长的配置文件行。以下示例显示了分成四行的宏定义：

```
MACRO srvlist=node1,node2,node3,node4, \
                node5,node6,node7,node8, \
                node9,node10,node11,node12, \
                node13,node14,node15
```

连接管理器配置文件参数

下表列出了可以在连接管理器配置文件的不同部分中使用的参数。有关每个参数的更多信息，请访问表中该参数的链接。

表 1. 连接管理器配置文件参数

配置文件的组成部分	必需参数	可选参数
头配置文件参数	• NAME	• LOCAL_IP • LOG • LOGFILE • DEBUG • MACRO • 连接单元类型和名称 • CM_TIMEOUT • EVENT_TIMEOUT • SECONDARY_EVENT_TIMEOUT
用于高可用性集群的主体	• SLA • GBASEDBTSERVER • FOC	• CMALARMPROGRAM
用于网格、复制集或服务器集的主体	• SLA • GBASEDBTSERVER	• CMALARMPROGRAM • SQLHOSTS

示例 1：用于高可用性集群的基本配置文件

```
NAME        cm1
LOG         1
LOGFILE     ${GBASEDBTDIR}/tmp/cm1.log

CLUSTER west
{
    GBASEDBTSERVER   ids_w1,ids_w2
    SLA   oltp      DBSERVERS=primary
    SLA report DBSERVERS=HDR,SDS
    FOC ORDER=ENABLED \
        TIMEOUT=5 \
        RETRY=2
```

```
    CMALARMPROGRAM ${GBASEDBTDIR}/etc/CMALARMPROGRAM.sh
}
```

此示例针对高可用性集群配置了连接管理器，并定义了两个 SLA。

配置文件头定义了以下内容：

- cm1 定义为连接管理器实例的名称
- 已启用日志记录
- cm1.log 定义为日志文件的名称
- $GBASEDBTDIR/tmp 定义为日志目录

配置文件主体定义了两个 SLA、故障转移参数以及故障转移处理失败时要调用的程序：

- CONNECT TO @oltp 连接请求将定向到主服务器
- CONNECT TO @report 连接请求将定向到 HDR 辅助服务器。如果 HDR 辅助服务器不可用，CONNECT TO @report 连接请求将定向到任何可用的 SD 辅助服务器。
- 主服务器的 HA_FOC_ORDER 配置参数值在故障转移规则中使用。
- 连接管理器再等待 5 秒，以等待有关连接单元的主服务器故障转移处理开始的事件。
- 连接管理器在主服务器的 HA_FOC_ORDER 配置参数指定的列表中循环 2 次后，自动故障转移将终止并生成警报。
- 如果故障转移处理遇到错误，将调用 $GBASEDBTDIR/etc/CMALARMPROGRAM.sh。

某些参数和属性不包含在此配置文件中，因此连接管理器具有以下缺省行为：

- 未设置 DEBUG 参数，因此将禁用调试。
- 未设置 EVENT_TIMEOUT 参数，因此连接管理器将等待主服务器事件 60 秒，然后再开始故障转移处理。
- 未设置 SECONDARY_EVENT_TIMEOUT 参数，因此连接管理器将等待辅助服务器事件 60 秒，然后再与辅助服务器断开连接。
- 未设置 CM_TIMEOUT 参数，因此在下一个可用连接管理器成为故障转移仲裁器之前，数据库服务器将等待 60 秒以从故障转移仲裁器连接管理器接收事件。
- 未设置 SQLHOSTS 参数，因此连接管理器将依次检查本地 sqlhosts 文件和远程 sqlhosts 文件以查找 ids_w1 和 ids_w2 的实例。
- 未设置 SLA 参数的 MODE 属性，因此连接管理器会将连接请求重定向到 oltp 和 report，而不是充当代理服务器。
- 未设置 report SLA 的 POLICY 属性，因此连接管理器会将连接请求定向到工作负载最低的辅助服务器。

- 未设置 SLA 参数的 WORKERS 属性，因此将为每个 SLA 分配四个工作程序线程。
- 未设置 SLA 参数的 HOST、NETTYPE、SERVICE 和 SQLHOSTSOPT 属性，因此连接管理器将使用 ids_w1 和 ids_w2 的 sqlhosts 值。

示例 2：用于多个连接单元类型的复杂配置文件

```
NAME cm2
MACRO NY=(ny1,ny2,ny3)
MACRO CA=(ca1,ca2,ca3)
LOG            1
LOGFILE ${GBASEDBTDIR}/tmp/cm2.log

CLUSTER west
{
    GBASEDBTSERVER ids_w1,ids_w2
    SLA oltpw DBSERVERS=primary
    SLA reportw DBSERVERS=(HDR,SDS)
    FOC ORDER=ENABLED \
          TIMEOUT=5 \
          RETRY=1
    CMALARMPROGRAM /etc/CMALARMPROGRAM.sh
}

CLUSTER east
{
    GBASEDBTSERVER ids_e1,ids_e2
    SLA oltpe DBSERVERS=primary
    SLA reporte DBSERVERS=RSS,HDR
    FOC ORDER=ENABLED \
          TIMEOUT=5 \
          RETRY=1
    CMALARMPROGRAM ${GBASEDBTDIR}/etc/CMALARMPROGRAM.sh
}

REPLSET erset
{
    GBASEDBTSERVER g_er1,g_er2
    SLA repl1_any    DBSERVERS=ANY
```

```
    SLA repl1_ca DBSERVERS=${CA} \
                    POLICY=WORKLOAD
    SLA repl1_ny DBSERVERS=${NY}
}

GRID grid1
{
    GBASEDBTSERVER node1,node2,node3
    SLA grid1_any DBSERVERS=ANY \
                    POLICY=LATENCY
    SLA grid1_avail DBSERVERS=${NY},${CA}
}

GRID grid2
{
    GBASEDBTSERVER node4,node5
    SLA grid2_any DBSERVERS=ANY \
                    POLICY=LATENCY
    SLA grid2_avail DBSERVERS=${CA},${NY}
}

SERVERSET ss
{
    GBASEDBTSERVER ids1,ids2,ids3
    SLA ssavail DBSERVERS=ids1,ids2,ids3 \
                    HOST=apollo \
                    SERVICE=9600 \
                    NETTYPE=onsoctcp
    SLA ssany DBSERVERS=(ids1,ids2,ids3) \
                HOST=apollo \
                SERVICE=9610 \
                NETTYPE=onsoctcp
}
```

此示例将连接管理器配置为支持两个高可用性集群、一个复制集、两个网格和一个服务器集。

配置文件头定义了以下内容：

- cm2 定义为连接管理器实例的名称

- 定义了两个宏
  - NY，由 ny1、ny2 和 ny3 组成
  - CA，由 ca1、ca2 和 ca3 组成
- 已启用日志记录
- cm2.log 定义为日志文件的名称
- $GBASEDBTDIR/tmp 定义为日志目录

配置文件主体定义了用于六个连接单元的服务级别协议和故障转移处理选项。配置文件头中定义的宏在某些 SLA 中使用。

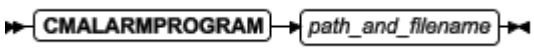
某些参数和属性不包含在此配置文件中，因此连接管理器具有以下缺省行为：

- 未设置 DEBUG 参数，因此将禁用调试。
- 未设置 EVENT_TIMEOUT 参数，因此连接管理器将等待主服务器事件 60 秒，然后再开始故障转移处理。
- 未设置 SECONDARY_EVENT_TIMEOUT 参数，因此连接管理器将等待辅助服务器事件 60 秒，然后再与辅助服务器断开连接。
- 未设置 CM_TIMEOUT 参数，因此在下一个可用连接管理器成为故障转移仲裁器之前，数据库服务器将等待 60 秒以从故障转移仲裁器连接管理器接收事件。
- 未设置 SQLHOSTS 参数，因此连接管理器将依次检查本地 sqlhosts 文件和远程 sqlhosts 文件以查找数据库服务器实例。

连接管理器配置文件的参数

要设置连接管理器，您必须创建连接管理器配置文件。

CMALARMPROGRAM 参数



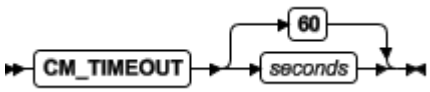
指定在故障转移处理遇到错误时要运行的程序或脚本的路径和文件名。如果故障转移处理失败，那么连接管理器将调用 CMALARMPROGRAM 指定的程序。

在以下示例中，如果故障转移处理失败，将调用 cmalarmprogram.sh：

```
${GBASEDBTDIR}/etc/cmalarmprogram.sh
```

${GBASEDBTDIR} 是 GBASEDBTDIR 环境变量的值。

CM_TIMEOUT 参数



指定在下一个可用连接管理器成为故障转移仲裁器之前，数据库服务器等待从故障转移仲裁器连接管理器接收事件的秒数。 如果未指定 CM_TIMEOUT，超时为 60 秒。

DEBUG 参数

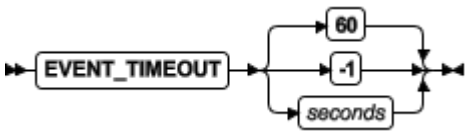
指定是启用还是禁用调试方式。

- 1 启用 SQL 和 ESQL/C 错误消息的日志记录。
- 0 禁用 SQL 和 ESQL/C 错误消息的日志记录。

如果连接管理器配置文件中未指定 DEBUG，将禁用调试方式。

不能从命令行启用调试方式。

EVENT_TIMEOUT 参数

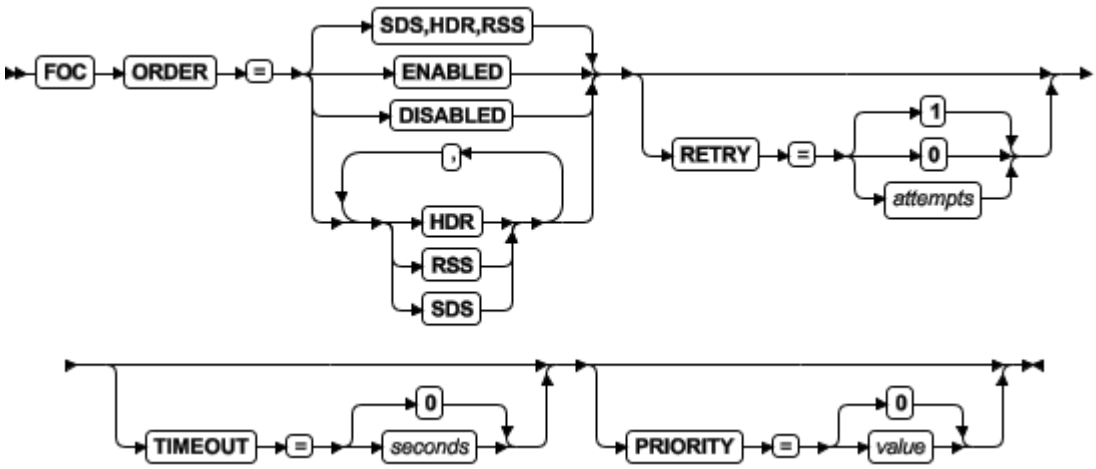


指定连接管理器在开始故障转移处理之前，等待主服务器事件的秒数。主服务器事件由主服务器发出，用于表明该服务器仍在运行，如性能统计信息、节点更改或管理消息。

- -1 表示连接管理器无限期等待来自主服务器的事件，或等到收到某个辅助服务器有关主服务器已脱机的通知为止。
- 值 0 到 30 会读取为 30。

  如果连接管理器配置文件中未指定 EVENT_TIMEOUT，那么连接管理器在开始故障转移处理之前，将等待主服务器事件 60 秒。

FOC 参数和属性



指定连接管理器用于确定在主服务器发生故障时由哪台服务器进行接管的故障转移配置。FOC 参数对于连接单元类型 REPLSET、GRID 和 SERVERSET 无效。

重要： 如果指定了 FOC 参数并且故障转移处理开始，那么在故障转移处理完成之前，不能手动重新启动发生故障的主服务器。

表 1. FOC 参数的属性

FOC 参数的属性	描述
ORDER	指定服务器类型的逗号分隔列表，或指定服务器类型列表是否来自集群的主服务器的 HA_FOC_ORDER 配置参数。 要配置多个连接管理器实例来管理高可用性集群，请在主服务器的 onconfig 文件中设置 HA_FOC_ORDER 配置参数。 • ENABLED 指定在故障转移仲裁时会考虑连接管理器，并使用主服务器的 HA_FOC_ORDER 配置参数值来确定节点故障转移顺序。 • DISABLED 禁止连接管理器进行故障转移处理。如果禁用了故障转移处理，连接管理器将忽略故障转移请求。 • HDR 指定高可用性数据复制辅助服务器。 • RSS 指定远程独立辅助服务器 • SDS 指定共享磁盘辅助服务器。 如果未指定 ORDER 属性，并且未设置主服务器的 HA_FOC_ORDER 配置参数，那么故障转移的顺序依次为 SDS、HDR 和 RSS。 重要： 如果 FOC 参数的 PRIORITY 属性设置为正整数，那么 ORDER 属性必须设置为 ENABLED。
PRIORITY	指定在应用程序所在相同主机上或在高可用性集群中应用程序服务器上运行的连接管理器的故障转移优先级。PRIORITY 值必须为 0 或者正整数值，并且在配置为管理特定集群的所有连接管理器中必须是唯一的。 • 0 指定在发生集群网络故障后禁用故障转移。

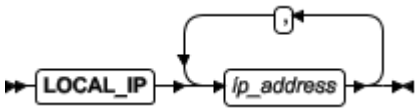
	如果活动的仲裁器连接管理器检测到主数据库服务器处于不活动状态，或者如果与主数据库服务器的连接丢失，那么该仲裁器连接管理器将开始故障转移。 • 1 指定连接管理器具有最高优先级。 **重要：** 如果 PRIORITY 值设置为正整数，那么 FOC 参数的 ORDER 属性必须设置为 ENABLED。 如果未指定 PRIORITY 属性，那么在发生集群网络故障后将禁用连接管理器故障转移。 仅当优先级最高的连接管理器可以保持与新的主数据库服务器的有效连接时，才能执行故障转移。例如，如果 PRIORITY=2 的连接管理器尝试执行故障转移时，PRIORITY=1 的连接管理器与主数据库服务器的连接会丢失，那么将阻塞该故障转移请求。然而，如果 PRIORITY=1 的连接管理器尝试开始故障转移时，PRIORITY=2 的连接管理器与主数据库服务器的连接会丢失，那么将允许该故障转移请求继续。
RETRY	指定连接管理器尝试完成故障转移期间，在 ORDER 属性值中循环的次数。在未成功故障转移的情况下达到 RETRY 限制后，将生成警报，并且连接管理器会终止自动执行的故障转移尝试。 • 0 指示连接管理器无限期地继续故障转移处理。 • 如果未指定 RETRY 属性，连接管理器将在 ORDER 属性值中循环一次。
TIMEOUT	指定连接管理器在开始故障转移处理之前，再等待主服务器事件的秒数。 TIMEOUT 属性值在超出 EVENT_TIMEOUT 参数值后应用。例如，如果 EVENT_TIMEOUT 参数设置为 60，而

	TIMEOUT 值设置为 10，那么必须在经过 70 秒都没有收到任何主服务器事件的情况下，才能开始故障转移。 如果未指定 TIMEOUT 属性，那么故障转移将在超出 EVENT_TIMEOUT 参数值指定的时间量后立即开始。

GBASEDBTSERVER 参数

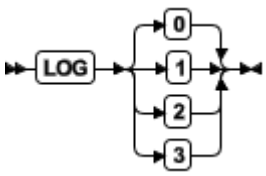
指定初始化期间连接管理器连接到的数据库服务器。

LOCAL_IP 参数



指定运行连接管理器的计算机上要监视的 IP 地址。LOCAL_IP 参数与 FOC 参数的 PRIORITY 属性结合使用。必须列出要监视的每块网络接口卡的 IP 地址。

LOG 参数

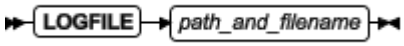


为连接管理器方式指定日志记录。

- 0 指定不进行日志记录。
- 1 指定记录有关 PROXY 和 REDIRECT 服务级别协议的连接信息。
- 2 仅记录 PROXY 方式 SLA 信息，并指定记录客户机和连接管理器之间的数据发送和接收活动。
- 3 仅记录 PROXY 方式 SLA 信息，并指定记录客户机与连接管理器之间的数据内容。

如果未设置 LOG 参数，将禁用日志记录。

LOGFILE 参数



指定连接管理器日志文件的名称和位置。连接管理器启动时会显示该日志文件的路径和文件名，并且在连接管理器运行期间，该日志文件会持续更新状态信息。 该日志文件对于监视和故障诊断用途是必需的。

确保该日志文件的目录存在，并且已为启动连接管理器的用户启用对该文件的访问权。

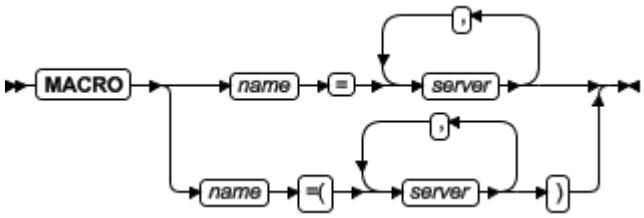
如果未设置 LOGFILE 参数，连接管理器将在 $GBASEDBTDIR/tmp 目录中创建日志文件，名称为connection_manager_name.process_ID.log。

在以下示例中，cm.log 是连接管理器日志文件：

```
${GBASEDBTDIR}/tmp/cm.log
```

${GBASEDBTDIR} 是 GBASEDBTDIR 环境变量的值。

MACRO 参数



指定宏的名称和服务器名称列表。宏在用于 SLA 定义时将成为服务器名称列表。

例如，以下参数定义名为 CA 的宏，其中列出了三台服务器：

```
MACRO CA=ca1,ca2,ca3
```

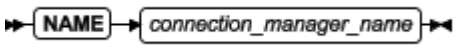
通过用圆括号将服务器列表括起，可以在宏内定义负载均衡（其中连接基于所列数据库服务器的 CPU 使用率进行定向）：

```
MACRO CA=(ca1,ca2,ca3)
```

MACRO 参数可以设置多次以创建多个宏。 例如：

```
MACRO NorthCA=ca1,ca2,ca3
MACRO SouthCA=ca4,ca5,ca6
```

NAME 参数



指定连接管理器实例的名称。 连接管理器实例的名称对于监视、关闭或重新装入连接管理器是必需的。如果未设置 NAME 参数，那么第一个 SLA 的名称将用作连接管理器实例名称。连接管理器实例名称必须唯一。

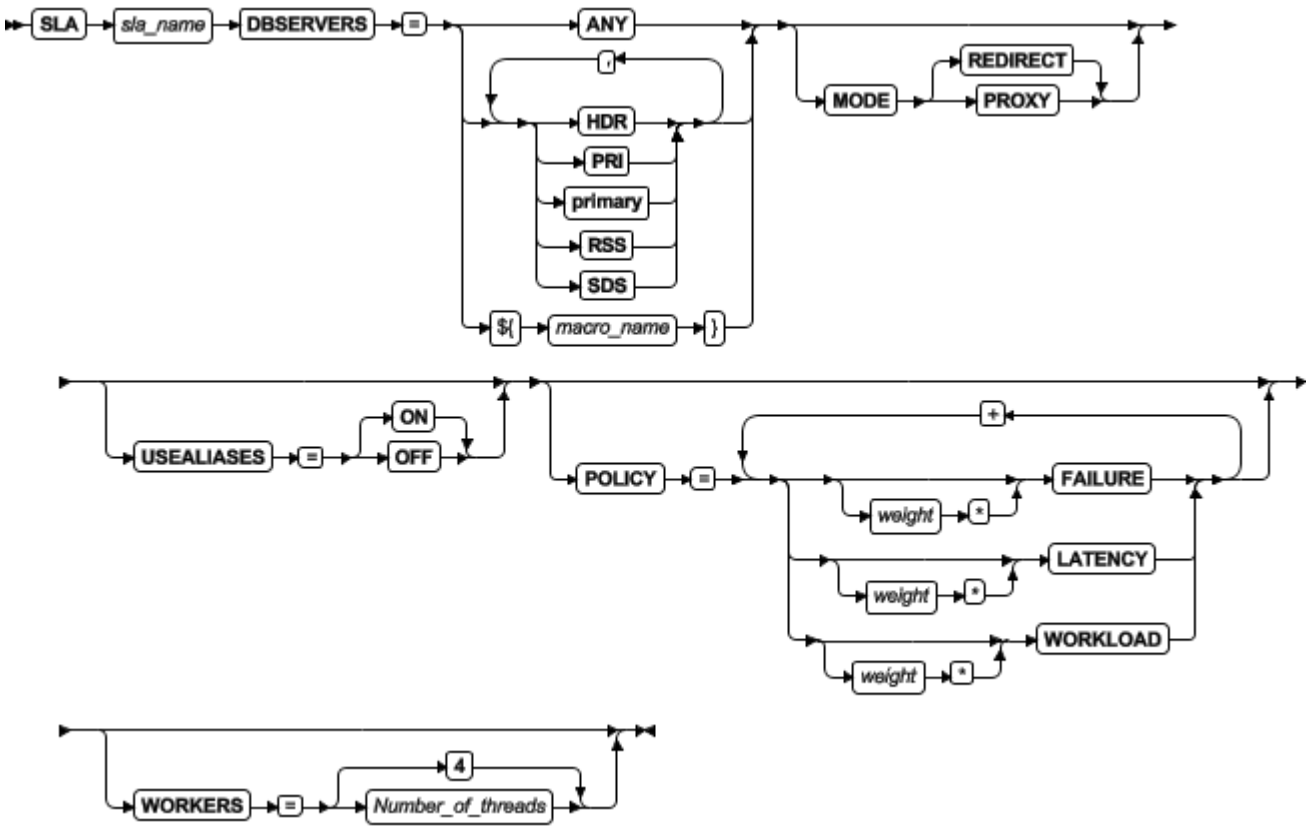
SECONDARY_EVENT_TIMEOUT 参数

指定连接管理器在与辅助服务器断开连接之前，等待辅助服务器事件的秒数。

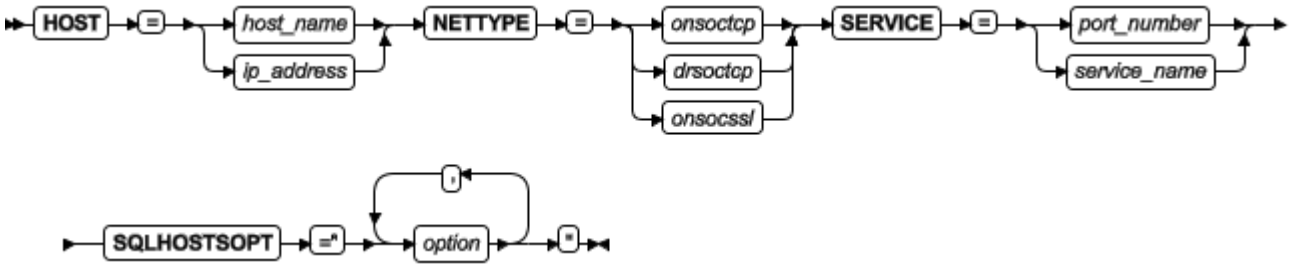
- -1 表示连接管理器无限期等待来自辅助服务器的事件。
- 值 0 到 30 会读取为 30。

如果未设置 SECONDARY_EVENT_TIMEOUT 参数，超时将为 60 秒。

SLA 参数



SQLHOSTS 属性



客户机应用程序使用 SLA 名称连接到 DBSERVERS 属性值指定的数据库服务器或数据库服务器类型。对于每个 SLA，侦听器线程会安装在服务器上的指定端口处，用于检测入局客户机请求。SLA 参数可以在同一配置文件中多次指定；但是，每个 SLA 名称必须唯一。

在一些可选属性可以在 SLA 定义中进行设置：

- 每个 SLA 通常在 sqlhosts 文件中都有一个对应的条目；但是，用户可以改为通过设置 NETTYPE、HOST、SERVICE 和 SQLHOSTSOPT 属性，在 SLA 内部指定 sqlhots 信息。例如，可以在 SLA 中输入以下 sqlhosts 条目中的值：

```
#dbserver   nettype   hostname   servicename   options
oltp1       onsoctcp host_1     sales_1       csm=path
```

SLA 条目：

```
SLA oltp1 DBSERVERS=primary \
          NETTYPE=onsoctcp \
          HOST=host_1 \
          SERVICE=sales_1 \
          SQLHOSTSOPT="cms=path"
```

- MODE 属性指定连接请求是通过连接管理器传递，还是由连接管理器重定向。
- POLICY 属性指定连接管理器如何确定选择列出的哪个数据库服务器来接收连接请求。
- USEALIASES 属性指定连接管理器是否可以将客户机连接请求重定向到 DBSERVERALIASES 配置参数指定的数据库服务器。

表 2. SLA 参数的必需属性和值。

必需属性	值
DBSERVERS	指定服务器、服务器别名、服务器类型或先前定义的用于定向连接请求的宏。 用圆括号将列表括起可启用负载均衡，其中连接将定向到 CPU 利用率最低的服务器。 • ANY 指定连接请求可以发送到任何可用的数据库服务器。 • HDR 指定连接请求可以发送到高可用性数据复制服务器。

必需属性	值
	• PRI 或 primary 指定连接请求可以发送到主数据库服务器。 • SDS 指定连接请求可以发送到共享磁盘辅助服务器。 • RSS 指定连接请求可以发送到远程独立辅助服务器。 • 特定服务器名称或别名指定连接请求可以发送到具有该名称或别名的数据库服务器。 • ${macro} 指定连接请求可以发送到配置文件头中的宏内所定义的数据库服务器。

表 3. SLA 参数的可选属性和值。

可选属性	值
MODE	指定连接请求是通过连接管理器传递，还是由连接管理器重定向。 • PROXY 指定客户机连接将连接管理器用作代理服务器。客户机应用程序无法连接到位于防火墙后面的数据库服务器时，请使用代理方式。 由于代理服务器连接管理器将处理所有客户机请求，因此配置多个连接管理器实例可避免连接管理器成为单个故障点。 • REDIRECT 指定客户机连接使用重定向方式，这将连接管理器配置为向客户机应用程序返回相应服务器节点、IP 地址和端口号。客户机应用程序使用连接管理器返回的 IP 地址和端口号来连接到数据库服务器。 如果未设置 MODE 属性，连接请求将由连接管理器重定向。
POLICY	指定连接管理器如何定向客户机连接请求。指定 LATENCY 并不表示特定等待时间段；而是表示连接管理器将使用采用相对值的公式来确定将客户机连接请求定向到的位置。 • LATENCY 指定连接管理器将连接请求定向到等待时间最短的服务器。 • FAILURE 指定连接请求将定向到应用失败次数最少的服务器。 • WORKLOAD 指定连接请求将定向到工作负载最低的服务器 如果未设置 POLICY 属性，连接管理器会将连接请求定向到工作负载最低的服务器。 可以为策略值提供相对权重。例如，要将客户机请求定向到等待时间最短

	且应用失败次数最少（但等待时间比应用失败次数重要 10 倍）的服务器，请使用以下值： `POLICY=10*LATENCY+FAILURE` 要使用等待时间和失败策略： • 网格必须具有启用复制的表。 • 必须启用数据质量 (qod) 监视。

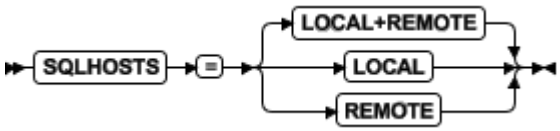
USEALIASES	指定连接管理器是否可以将客户机连接请求重定向到 DBSERVERALIASES 配置参数指定的数据库服务器别名。
	• ON 指定连接管理器可以将客户机连接请求定向到 DBSERVERS 属性指定的服务器别名。
	• OFF 指定连接管理器不将客户机连接请求定向到 DBSERVERS 属性指定的服务器别名。
	如果未设置 USEALIASES 属性，连接管理器可以将客户机连接请求定向到 DBSERVERS 属性指定的服务器别名。
	
	例如，数据库服务器 **srv1** 具有别名 **srv1_alias1** 和 **svr1_alias2**，并且连接管理器包含以下服务级别协议：
	
	`SLA sla1 DBSERVERS=srv1`
	
	`SLA sla2 DBSERVERS=srv1 \`
	
	`USEALIASES=OFF`
	
	`SLA sla3 DBSERVERS=srv1_alias1`
	
	`SLA sla4 DBSERVERS=srv1_alias1 \`
	
	`USEALIASES=OFF`
	
	连接管理器将通过以下方式定向客户机请求：

	• CONNECT TO @sla1 请求可以定向到 srv1、srv1_alias1 和 srv1_alias2。 • CONNECT TO @sla2 请求将定向到 srv1。 • CONNECT TO @sla3 请求可以定向到 srv1、srv1_alias1 和 srv1_alias2。 • CONNECT TO @sla4 请求将定向到 srv1_alias1。
WORKERS	指定分配给 SLA 的工作程序线程数。指定了服务级别协议时，连接管理器将创建 SLA 侦听器进程来拦截客户机连接请求。SLA 侦听器进程可以有一个或多个线程，称为工作程序线程。 如果未设置 WORKERS 属性，将为 SLA 分配 4 个工作程序线程。

表 4. SLA 参数的可选 SQLHOSTS 属性和值。

可选 SQLHOSTS 属性	值
HOST	指定数据库服务器的主机。将使用 SLA 中的值，而不是 sqlhosts 文件中的值。 • host_name 指定数据库服务器的主机名或主机别名。 • ip_address 指定数据库服务器的 TCP/IP 地址。
NETTYPE	指定数据库服务器的网络协议。将使用 SLA 中的值，而不是 sqlhosts 文件中的值。 • onsoctcp 指定使用 TCP/IP 协议的套接字
SERVICE	指定数据库服务器的端口号或服务名称。将使用 SLA 中的值，而不是 sqlhosts 文件中的值。 • port_number 指定端口号。 • service_name 指定服务名称。
SQLHOSTSOPT	指定在 SLA 中所指定的数据库服务器的连接选项。用一对引号将所有连接选项括起。将使用 SLA 中的值，而不是sqlhosts 文件中的值。

SQLHOSTS 参数



要禁止客户机应用程序访问高可用性集群中的一个或多个数据库服务器时，SQLHOSTS 选项会很有用。

连接管理器可以使用本地计算机或远程计算机上的 sqlhosts 文件条目。

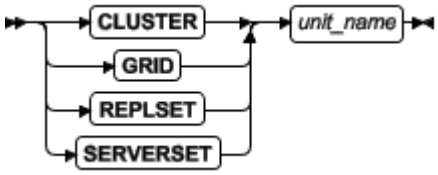
表 5. SQLHOSTS 连接管理器配置参数的值

SQLHOSTS 参数值	描述
LOCAL	连接管理器在 GBASEDBTSQLHOSTS 环境变量指定的本地 sqlhosts 文件中搜索请求的数据库服务器实例。
REMOTE	连接管理器在远程计算机上的 sqlhosts 文件中搜索请求的数据库服务器实例。远程计算机由本地数据库服务器的 GBASEDBTSQLHOSTS 环境变量指定。 远程计算机上的 sqlhosts 文件由远程数据库服务器的 GBASEDBTSQLHOSTS 环境变量指定。
LOCAL+REMOTE（缺省值）	连接管理器依次检查本地 sqlhosts 文件和远程 sqlhosts 文件以查找 GBase 8s 实例。

如果未设置 SQLHOSTS 参数，连接管理器将首先检查本地 sqlhosts 文件以查找 GBase 8s 实例。 如果找不到请求的服务器条目，那么将使用远程服务器上的 DBSERVERNAME 或 DBSERVERALIASES 配置参数。

例如，您有一个 Web 服务器（位于外部网络上）和一个应用程序服务器（位于内部网络上）。 数据库服务器为外部网络连接定义了 TCP/IP port1 和 port2，为内部连接定义了 TCP/IP port3 和 port4。指定 SQLHOSTS local 会使连接管理器只查找连接管理器实例的 sqlhosts 文件中定义的服务器实例。在这种情况下，连接管理器只会将客户机连接请求定向到外部网络上的服务器。

unit_type



连接单元是特定的高可用性服务器配置。connection_unit_name 指定连接管理器识别连接单元所依据的名称。 在连接管理器配置文件中指定组类型和连接单元，以将客户机连接请求定向到相应的数据库服务器。每个名称在配置文件中必须唯一。

在配置文件中指定 CLUSTER 连接单元的名称时，请将与组名相同的名称用于 GBASEDBTSQLHOSTS 文件中的对应集群。

在连接管理器配置文件中指定连接单元名称时，只有 GRID 和 REPLSET 连接单元类型可以使用多字节字符。

连接管理器配置文件中服务级别协议的示例

以下示例显示了如何针对高可用性集群、复制集、网格和服务器集来配置各种服务级别协议 (SLA)。

在以下示例中，可以使用服务器别名或通用服务器类型。例如，RSS 指定连接管理器对集群中的每个 RS 辅助服务器进行评估，以确定客户机应用程序请求的最佳候选项。您还可以指定数据库服务器的网络别名，例如 sds1 或 rss1。

以下配置文件示例显示了各种选项：

示例 1：从服务级别协议进行连接请求重定向

```
NAME cm1
LOG          1
LOGFILE ${GBASEDBTDIR}/tmp/cm1.log

CLUSTER east
{
    GBASEDBTSERVER ids_e1,ids_e2
    SLA secondaryNodes DBSERVERS=SDS,HDR,PRI
    FOC ORDER=ENABLED \
        TIMEOUT=5 \
        RETRY=1
    CMALARMPROGRAM ${GBASEDBTDIR}/etc/CMALARMPROGRAM.sh
}
```

连接管理器 cm1 按如下所示定向 CONNECT TO @secondaryNodes 连接请求：

连接到任何可用 SD 辅助服务器。

如果 SD 辅助服务器不可用，那么连接到 HDR 辅助服务器。

如果 HDR 辅助服务器不可用，那么连接到主服务器。

示例 2：指定多个服务级别协议

```
NAME         cm2
```

```
LOG          1
LOGFILE    ${GBASEDBTDIR}/tmp/cm2.log

CLUSTER west
{
    GBASEDBTSERVER   ids_w1,ids_w2
    SLA   oltp      DBSERVERS=primary
    SLA report DBSERVERS=HDR,SDS
    FOC ORDER=ENABLED \
        TIMEOUT=5 \
        RETRY=2
    CMALARMPROGRAM ${GBASEDBTDIR}/etc/CMALARMPROGRAM.sh
}
```

此示例针对高可用性集群配置了连接管理器，并定义了两个 SLA。

配置文件主体定义了两个 SLA、故障转移参数以及故障转移处理失败时要调用的程序：

CONNECT TO @oltp 连接请求将定向到主服务器

CONNECT TO @report 连接请求将定向到 HDR 辅助服务器。如果 HDR 辅助服务器不可用，CONNECT TO @report 连接请求将定向到任何可用的 SD 辅助服务器。

示例 3：服务级别协议中的负载均衡

要启用服务器间的负载均衡，请用圆括号将服务器类型列表括起。

```
NAME cm3
LOG          1
LOGFILE ${GBASEDBTDIR}/tmp/cm3.log

CLUSTER south
{
    GBASEDBTSERVER ids_s1,ids_s2
    SLA secondary DBSERVERS=(SDS,HDR)
    FOC ORDER=ENABLED \
        TIMEOUT=5 \
        RETRY=1
    CMALARMPROGRAM ${GBASEDBTDIR}/etc/CMALARMPROGRAM.sh
}
```

连接管理器 cm3 将客户机 CONNECT TO @secondary 连接请求定向到 CPU 利用率最低的 SD 或 HDR 辅助服务器。

示例 4：服务级别协议中的代理和重定向方式

在重定向方式下，连接管理器将指定数据库服务器的 IP 地址和端口号发送到客户机应用程序。 然后客户机应用程序使用此 IP 地址和端口号来连接到数据库服务器。在代理方式下，客户机请求将通过连接管理器进行路由。如果未指定 SLA 方式，重定向方式是缺省方式。

```
NAME cm4
LOG          1
LOGFILE ${GBASEDBTDIR}/tmp/cm4.log

CLUSTER north
{
    GBASEDBTSERVER ids_n1,ids_n2
    SLA sla1 DBSERVERS=alpha \
                MODE=REDIRECT
    SLA sla2 DBSERVERS=beta \
                MODE=PROXY
    SLA sla3 DBSERVERS=SDS \
                MODE=REDIRECT
    FOC ORDER=ENABLED \
        TIMEOUT=5 \
        RETRY=1
    CMALARMPROGRAM ${GBASEDBTDIR}/etc/CMALARMPROGRAM.sh
}
```

对于此示例：

主服务器的别名为 alpha

HDR 辅助服务器的别名为 beta

有两个 SD 辅助服务器，别名为 gamma1 和 gamma2

连接管理器 cm4 按如下所示定向客户机连接请求：

CONNECT TO @sla1 连接请求将定向到主服务器 (alpha)。

CONNECT TO @sla2 连接请求将通过连接管理器（充当代理服务器）定向到 HDR 辅助服务器 (beta)。

CONNECT TO @sla3 连接请求将定向到可用资源最多的 SD 辅助服务器。

示例 5：主服务器、辅助服务器和连接管理器的超时

```
NAME cm5
LOG          1
```

```
LOGFILE ${GBASEDBTDIR}/tmp/cm5.log
CM_TIMEOUT 300
EVENT_TIMEOUT 45
SECONDARY_EVENT_TIMEOUT 50

CLUSTER southwest
{
    GBASEDBTSERVER ids_sw1,ids_sw2
    SLA    oltp       DBSERVERS=primary
                  MODE=PROXY
    SLA report DBSERVERS=SDS,RSS,HDR
    SLA primary DBSERVERS=primary
    SLA secondary DBSERVERS=(SDS,RSS,HDR)
    FOC ORDER=ENABLED \
          RETRY=1
    CMALARMPROGRAM ${GBASEDBTDIR}/etc/CMALARMPROGRAM.sh
}
```

如果数据库服务器在 300 秒内未接收到来自连接管理器的任何事件，该数据库服务器将假定连接管理器未运行，而下一个可用的连接管理器将成为故障转移仲裁器。

如果连接管理器在 45 秒内未接收到来自主服务器的任何事件，那么连接管理器将开始故障转移处理。如果为某个连接单元定义了 FOC 参数的 TIMEOUT 属性，那么该参数的值会与 EVENT_TIMEOUT 参数的值相加，所得结果作为该连接单元的故障转移处理开始之前要等待的总时间。

如果连接管理器在 50 秒内未接收到来自辅助服务器的任何事件，那么连接管理器将与该辅助服务器断开连接。

- CONNECT TO @oltp 连接请求将通过连接管理器（充当代理服务器）定向到主服务器。
- CONNECT TO @report 连接请求将定向到第一个可用的 SD 辅助服务器。如果 SD 辅助服务器不可用，CONNECT TO @report 连接请求将发送到任何可用的 RS 辅助服务器。如果 RS 辅助服务器不可用，那么 CONNECT TO @report 连接请求将发送到 HDR 辅助服务器。
- CONNECT TO @primary 连接请求将定向到主服务器。
- CONNECT TO @secondary 连接请求将通过负载均衡定向到可用资源最多的辅助服务器。

示例 6：服务级别协议中的宏和工作负载均衡

可以为服务器组定义宏。可以在宏内或 SLA 内定义负载均衡。

```
NAME cm6
MACRO NY=(ny1,ny2,ny3)
MACRO CA=(ca1,ca2,ca3)
LOG          1
LOGFILE ${GBASEDBTDIR}/tmp/cm6.log

REPLSET erset
{
    GBASEDBTSERVER g_er1,g_er2
    SLA repl1_any DBSERVERS=ANY
    SLA repl1_ca DBSERVERS=${CA}
    SLA repl1_ny DBSERVERS=${NY}
}
```

在此示例中，定义了两个宏：

- NY，由 ny1、ny2 和 ny3 组成，并使用工作负载均衡。
- CA，由 ca1、ca2 和 ca3 组成，并使用工作负载均衡。
- 连接管理器 cm6 按如下所示定向客户机连接请求：
- CONNECT TO @repl1_any 连接请求将定向到任何可用的服务器。
- CONNECT TO @repl1_ca 连接请求将定向到 ca1、ca2 或 ca3 中 CPU 利用率最低者。
- CONNECT TO @repl1_ny 连接请求将定向到 ny1、ny2 或 ny3 中 CPU 利用率最低者。

示例 7：服务级别协议中的重定向策略

```
NAME cm7
MACRO SF=(sf1,sf2,sf3)
MACRO LA=(la1,la2,la3)
LOG          1
LOGFILE ${GBASEDBTDIR}/tmp/cm7.log

GRID grid1
{
    GBASEDBTSERVER node1,node2,node3
    SLA grid1_any DBSERVERS=ANY
                  POLICY=LATENCY
    SLA grid1_avail DBSERVERS=${SF},${LA}
}
```

在此示例中，定义了两个宏：

- SF，由 sf1、sf2 和 sf3 组成，并使用工作负载均衡。
- LA，由 la1、la2 和 la3 组成，并使用工作负载均衡。

连接管理器 cm7 按如下所示定向客户机连接请求：

- CONNECT TO @grid1_any 连接请求可以定向到任何可用的服务器。连接请求将定向到等待时间最短的服务器。
- CONNECT TO @grid1_avail 连接请求将定向到 sf1、sf2 或 sf3 中 CPU 利用率最低者。如果 sf1、sf2 和 sf3 不可用，连接请求将发送到 la1、la2 或 la3 中 CPU 利用率最低者。

示例 8：服务级别协议中的 sqlhosts 连接信息

```
NAME cm8
LOG          1
LOGFILE ${GBASEDBTDIR}/tmp/cm8.log

SERVERSET ss
{
    GBASEDBTSERVER ids1,ids2,ids3
    SLA ssavail DBSERVERS=ids1,ids2,ids3 \
                    HOST=apollo \
                    SERVICE=9600 \
                    NETTYPE=onsoctcp
    SLA ssany DBSERVERS=(ids1,ids2,ids3) \
                HOST=apollo \
                SERVICE=9610 \
                NETTYPE=onsoctcp
}
```

此示例显示了支持连接管理器所支持的服务器集的示例配置。连接管理器使用 HOST、SERVICE 和 NETTYPE 属性的值，而不是服务器的 sqlhosts 文件中的值。

对于 ssavail SLA，ids1、ids2 和 ids3 具有以下 sqlhosts 信息：

```
#dbservername     nettype         hostname        servicename         options
ids1              onsoctcp        apollo          9600
ids2              onsoctcp        apollo          9600
ids3              onsoctcp        apollo          9600
```

对于 ssany SLA，ids1、ids2 和 ids3 具有以下 sqlhosts 信息：

```
#dbservername     nettype         hostname        servicename         options
```

```
ids1            onsoctcp    apollo      9610
ids2            onsoctcp    apollo      9610
ids3            onsoctcp    apollo      9610
```

连接管理器 cm8 按如下所示定向客户机连接请求：

- CONNECT TO @ssavail 连接请求将定向到 ids1。如果 ids1 不可用，CONNECT TO @ssavail 连接请求将发送到 ids2。如果 ids2 不可用，CONNECT TO @ssavail 连接请求将发送到 ids3。
- CONNECT TO @ssany 连接请求将定向到 ids1、ids2 或 ids3 中 CPU 利用率最低者。

连接管理器服务级别协议中故障转移配置的示例

以下示例显示了由连接管理器执行的故障转移的配置示例。

以下示例显示了将连接管理器配置为在原始主服务器遇到问题时将辅助服务器提升为主服务器的多种方法。每个服务级别协议 (SLA) 都应该定义故障转移配置。

如果使用多个连接管理器来管理集群的故障转移，请在集群的主服务器上设置 HA_FOC_ORDER 配置参数。HA_FOC_ORDER 配置参数的值会替换连接到主服务器的每个连接管理器的配置文件中 FOC ORDER= 的值。

对于以下示例，所有服务器上的 HA_FOC_ORDER 配置参数都是缺省值 SDS,HDR,RSS（依次为共享磁盘辅助服务器、高可用性数据复制辅助服务器和远程独立辅助服务器）。

示例 1：立即故障转移

```
FOC ORDER=ENABLED TIMEOUT=0
```

在此示例中，指定了以下行为：

- 如果连接管理器检测到主服务器因在 EVENT_TIMEOUT 值指定的时间量内未发生任何主服务器事件而脱机，那么由于 TIMEOUT 值设置为 0，因此故障转移处理将立即启动。
- 连接管理器首先尝试将最适合的 SDS 服务器转换为主服务器。如果没有任何 SDS 服务器联机，那么连接管理器会尝试将 HDR 服务器转换为主服务器。 如果 HDR 服务器未联机，那么连接管理器会尝试将最适合的 RSS 服务器转换为主服务器。
- 由于未指定 RETRY 属性，因此连接管理器将在 ORDER 属性值中循环一次。

示例 2：包含延长的超时的故障转移

```
FOC ORDER=ENABLED TIMEOUT=10
```

在此示例中，指定了以下行为：

- 如果连接管理器检测到主服务器因在 EVENT_TIMEOUT 值指定的时间量内未发生任何主服务器事件而脱机，那么 TIMEOUT 值会使连接管理器再等待 10 秒，以等待主服务器恢复联机，然后才能开始故障转移处理。

- 故障转移处理期间，连接管理器首先尝试将最适合的 SDS 服务器转换为主服务器。如果没有任何 SDS 服务器联机，那么连接管理器会尝试将 HDR 服务器转换为主服务器。 如果 HDR 服务器未联机，那么连接管理器会尝试将最适合的 RSS 服务器转换为主服务器。
- 由于未指定 RETRY 属性，因此连接管理器将在 ORDER 属性值中循环一次。

示例 3：包含延长的超时、重试和警报的故障转移

```
FOC ORDER=ENABLED TIMEOUT=20 RETRY=2
CMALARMPROGRAM ${GBASEDBTDIR}/etc/cmalarmprogram.sh
```

在此示例中，指定了以下行为：

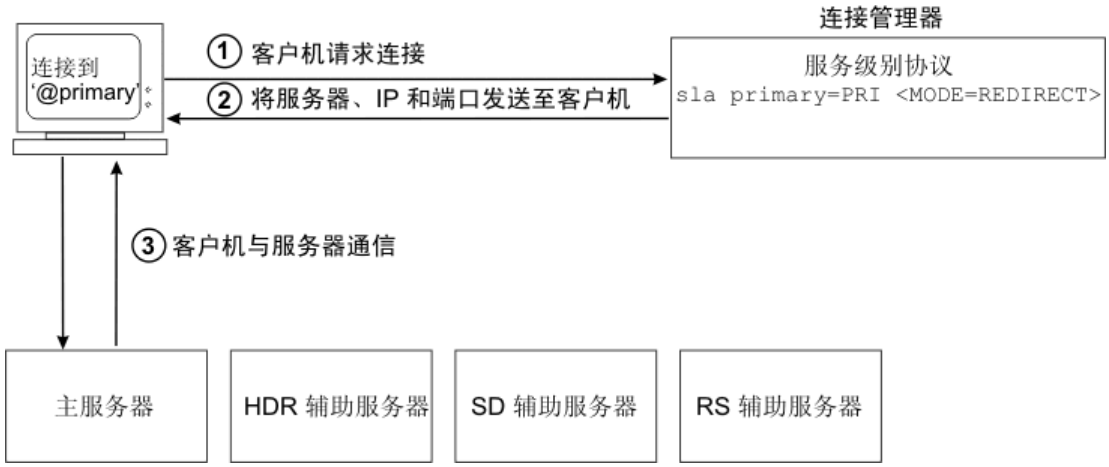
- 如果连接管理器检测到主服务器因在 EVENT_TIMEOUT 值指定的时间量内未发生任何主服务器事件而脱机，那么 TIMEOUT 值会使连接管理器再等待 20 秒，以等待主服务器恢复联机，然后才能开始故障转移处理。
- 故障转移处理期间，连接管理器首先尝试将最适合的 SDS 服务器转换为主服务器。如果没有任何 SDS 服务器联机，那么连接管理器会尝试将 HDR 服务器转换为主服务器。 如果 HDR 服务器未联机，那么连接管理器会尝试将最适合的 RSS 服务器转换为主服务器。
- 连接管理器在等待成功完成故障转移期间，最多会在 ORDER 列表 SDS,HDR,RSS 中循环两次。
- 如果在 ORDER 列表中循环两次后，故障转移仍未完成，那么将调用 cmalarmprogram.sh 程序来生成警报，并且连接管理器会终止故障转移。

6.4.3 连接管理器代理方式和重定向方式

连接管理器可以重定向客户机连接请求或充当代理服务器并处理所有客户机/服务器通信。

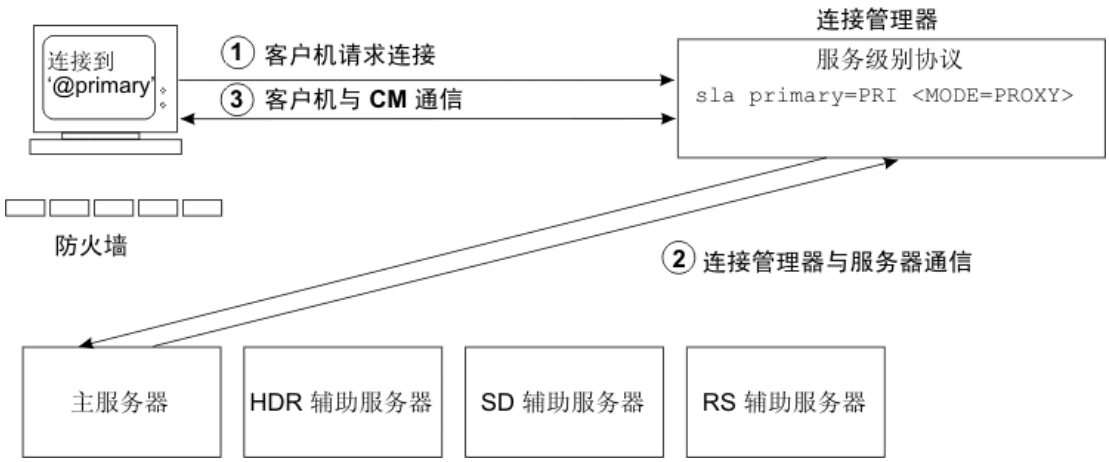
使用服务级别协议 MODE=REDIRECT 或 MODE=PROXY 属性可指定连接管理器如何处理客户机连接请求。

图： 在服务级别协议中设置重定向方式



在重定向方式下，连接管理器将相应服务器节点、IP 地址和端口号返回给发出连接请求的客户机应用程序。然后，客户机应用程序使用连接管理器提供的 IP 地址和端口号来连接到指定的数据库服务器。

图： 服务级别协议中的 MODE=PROXY



在代理方式下，连接管理器充当代理服务器，并管理客户机与数据库服务器的通信。当客户机应用程序无法连接到位于防火墙后面的数据库服务器时，请使用代理方式。要避免连接管理器成为单个故障点，在使用代理方式时，请配置多个连接管理器实例。

6.4.4 用于故障转移的连接管理器冗余

要防止连接管理器成为单个故障点，必须在不同计算机上配置连接管理器的多个实例。

连接管理器可以安装在未运行 GBase 8s 数据库服务器实例的计算机上，以防止连接管理在某个数据库服务器发生故障时发生故障。此外，还可以配置多个连接管理器实例，并使用 sqlhosts 故障转移功能来进一步提高可用性。

如果连接管理器服务级别协议使用的是重定向方式，那么在连接管理器发生故障后，客户机仍将保持与数据库服务器的连接；但是，除非配置了备份连接管理器实例，否则无法建立新客户机连接。如果连接管理器服务级别协议使用的是代理方式，那么连接管理器将充当代理服务器，并路由与指定数据库服务器的所有客户机通信。 如果代理服务器连接管理器发生故障，那么所有客户机连接都将丢失。

下图显示了其中包含两个连接管理器实例的高可用性集群示例。

图： 连接管理器组配置

主服务器和辅助服务器上的 sqlhosts 文件配置如下：

```
#dbservername     nettype         hostname        servicename         options
ifx_cluster       group           -               -                   i=30
ifx_pri           onsoctcp        p_machine       ifxsrv0pri          g=ifx_cluster
ifx_sds           onsoctcp        sds_machine     ifxsrv0sds          g=ifx_cluster
ifx_hdr           onsoctcp        hdr_machine     ifxsrv0hdr          g=ifx_cluster
ifx_rss           onsoctcp        rss_machine     ifxsrv0rss          g=ifx_cluster
```

sqlhosts 文件配置了一个其中包含集群内的每台服务器的服务器组，名称为 ifx_cluster。

运行连接管理器的服务器上和客户端计算机上的 sqlhosts 文件按如下所示进行配置：

```
#dbservername     nettype         hostname        servicename         options
ifx_pri           onsoctcp        p_machine       ifxsrv0pri
# ifx_cm group
```

```
ifx_cm              group       -              -              c=1
ifx_cm1             onsoctcp    cm1_machine    cm1_port1      g=ifx_cm
ifx_cm2             onsoctcp    cm2_machine    cm2_port2      g=ifx_cm
# ifx_proxy group
ifx_proxy           group       -              -              c=1
ifx_proxy1          onsoctcp    cm1_machine    cm1_proxy1     g=ifx_proxy
ifx_proxy2          onsoctcp    cm2_machine    cm2_proxy2     g=ifx_proxy
```

sqlhosts 文件中的条目指定以下行为：

- ifx_pri 是指定给主服务器的名称，这样发送到 ifx_pri 的客户机连接请求将定向到主服务器。
- ifx_cm 条目定义了由两个服务级别协议 ifx_cm1 和 ifx_cm2 构成的组。c=1 选项使客户机应用程序选择用于连接到服务器或服务级别协议列表的随机起点。 发送到 ifx_cm 的客户机连接请求将定向到 ifx_cm1 或 ifx_cm2。如果其中一个连接管理器实例脱机，那么客户机请求将自动路由到联机实例。
- ifx_proxy 条目还定义了由两个服务级别协议 ifx_proxy1 和 ifx_proxy2 构成的组。发送到 ifx_proxy 的客户机连接请求将定向到 ifx_proxy1 或 ifx_proxy2。如果其中一个连接管理器实例脱机，那么客户机请求将自动路由到联机实例。

如果由于 ifx_cm2 处于脱机状态或被随机选中而导致发送到 ifx_cm 的客户机连接请求定向到 ifx_cm1，那么 sqlhosts 文件会将该请求定向到正在 cm1_machine 上运行的连接管理器实例。与此类似，对 ifx_cm2 的请求也会定向到正在 cm2_machine 上运行的连接管理器实例。发送到 ifx_proxy 的连接请求将以同样方式发送到 cm1_machine 或 cm2_machine 上的连接管理器实例。

由于 ifx_cm1、ifx_cm2、ifx_proxy1 和 ifx_proxy2 是服务级别协议而不是数据库服务器，因此客户机请求将根据连接管理器配置文件中定义的服务级别协议重定向策略来定向。

下表中显示了第一个连接管理器实例 ifx_cm1 的配置文件：

```
NAME ifx_cm1
LOG          1
LOGFILE ${GBASEDBTDIR}/usr2/logs/cm1.log

CLUSTER my_cluster1
{
    GBASEDBTSERVER=ifx_pri,ifx_sds,ifx_rss,ifx_hdr
    SLA ifx_cm1 DBSERVERS=(SDS,HDR),RSS,primary
    SLA ifx_proxy1 DBSERVERS=HDR,RSS,primary \
                        MODE=PROXY
    FOC ORDER=ENABLED /
```

```
    TIMEOUT=5
}
```

此连接管理器配置文件指定了以下信息和行为：

- 连接管理器实例的名称为 ifx_cm1。
- 已启用日志记录，并且 cm1.log 日志文件位于 ${GBASEDBTDIR}/usr2/logs 目录中，其中 ${GBASEDBTDIR} 是 GBASEDBTDIR 环境变量的值。
- 连接单元是名为 my_cluster1 的高可用性集群。
- 初始化期间，连接管理器连接到 ifx_pri、ifx_sds、ifx_rss 和 ifx_hdr。
- CONNECT TO @ ifx_cm1 连接请求将定向到 SD 辅助服务器或 HDR 辅助服务器，具体取决于哪个数据库服务器的 CPU 使用率最低。如果 SD 辅助服务器和 HDR 辅助服务器均不可用，CONNECT TO @ ifx_cm1 连接请求将定向到 RS 辅助服务器。如果 RS 辅助服务器不可用，CONNECT TO @ ifx_cm1 连接请求将定向到主服务器。
- CONNECT TO @ ifx_proxy1 连接请求将连接管理器用作代理服务器。CONNECT TO @ ifx_proxy1 连接请求将定向到 HDR 辅助服务器。如果 HDR 辅助服务器不可用，CONNECT TO @ ifx_proxy1 连接请求将定向到 RS 辅助服务器。如果集群中包含多个 RS 辅助服务器，那么客户机请求将定向到工作负载最低的 RS 辅助服务器。如果 RS 辅助服务器不可用，客户机请求将路由到主服务器。
- 主服务器上 HA_FOC_ORDER 配置参数的值用于故障转移顺序配置。对于此示例，HA_FOC_ORDER 配置参数设置为其缺省值 SDS,HDR,RSS。
- 如果主服务器发生故障，SD 辅助服务器会成为主服务器。如果 SD 辅助服务器不可用，HDR 辅助服务器会成为主服务器。如果 HDR 辅助服务器不可用，RS 辅助服务器会成为主服务器。
- 如果主服务器发生故障，那么连接管理器必须在经过了 EVENT_TIMEOUT 和服务级别协议 TIMEOUT 值之和所表示的时间之后，才能将辅助服务器提升为主服务器。EVENT_TIMEOUT 参数的缺省值为 60 秒，因此必须在经过 65 秒都没有主服务器事件的情况下，才能将辅助服务器提升为主服务器。

下表中显示了第二个连接管理器实例 ifx_cm2 的配置文件：

```
NAME ifx_cm2
LOG          1
LOGFILE ${GBASEDBTDIR}/usr2/logs/cmsm2.log

CLUSTER my_cluster2
{
```

```
    GBASEDBTSERVER=ifx_pri,ifx_sds_ifx_rss,ifx_hdr
    SLA ifx_cm2 DBSERVERS=(SDS,HDR),RSS,primary
    SLA ifx_proxy2 DBSERVERS=HDR,RSS,primary \
                        MODE=PROXY
    FOC ORDER=ENABLED \
        TIMEOUT=5
}
```

配置多个连接管理器实例时，请确保每个连接管理器实例的服务级别协议 (SLA) 的定义完全相同。SLA 定义完全相同可确保无论使用哪个连接管理器实例，客户机都具有相同的重定向策略。除了 SLA 名称、集群名称和日志文件名之外，这两个配置文件完全相同。使用组名可确保即便其中一个连接管理器实例脱机，客户机也可连接到有效的连接管理器实例。sqlhosts 文件确保将发送到 ifx_cm 和 ifx_proxy 的请求定向到正确的连接管理器实例。如果连接管理器实例位于不同计算机上（如此示例中所示），那么不必严格要求使用不同的日志文件名。但是，如果连接管理器实例位于同一台计算机上，请使用不同的日志文件名或目录。

- 如果连接管理器配置文件位于 $GBASEDBTDIR/etc 目录中，并且名为 cm1_machine.cfg，请在 cm1_machine 上运行以下命令来启动连接管理器实例：

```
oncmsm -c $GBASEDBTDIR/etc/cm1_machine.cfg
```

- 如果连接管理器配置文件位于 $GBASEDBTDIR/etc 目录中，并且名为 cm2_machine.cfg，请在 cm2_machine 上运行以下命令来启动连接管理器实例：

```
oncmsm -c $GBASEDBTDIR/etc/cm2_machine.cfg
```

6.4.5 连接管理器网络监视和数据库服务器故障转移优先级

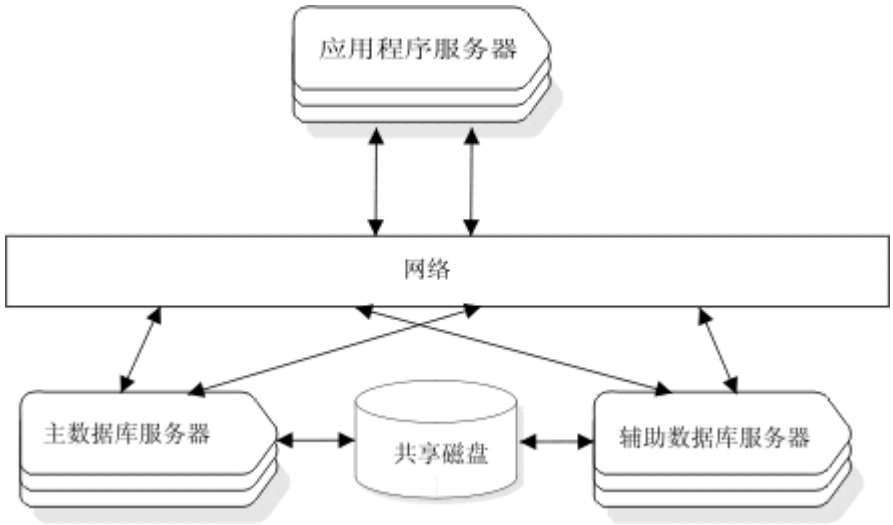
通过配置“连接管理器”以在发生网络故障时进行故障转移，可以确保应用程序服务器保持与数据库服务器的连接。

可以针对两种不同的情况配置数据库服务器故障转移：

- 当主服务器变得无法运行时
- 当应用程序服务器丢失与主服务器的网络连接时

以下示例是由一个应用程序服务器和带有一个 SD 辅助服务器的一个主服务器组成的网络。每个数据库服务器都配置了两块网络接口卡 (NIC)，因此每个数据库服务器都有两个 TCP/IP 地址。

图： 应用程序服务器和数据库服务器之间的网络连接



如果发生网络故障，并且应用程序服务器无法连接到主服务器，但可以连接到辅助服务器，那么在启用了网络监视的情况下，主服务器会故障转移到辅助服务器。

网络监视是通过设置 LOCAL_IP 参数和 FOC 参数的 PRIORITY 属性来启用的：

- LOCAL_IP 参数列出了运行连接管理器的计算机上 NIC 卡的可用 TCP/IP 地址。
- FOC 参数的 PRIORITY 属性用于配置在应用程序所在相同主机上或在高可用性集群中应用程序服务器上运行的连接管理器的故障转移优先级。

可为集群指定优先级，以防止因故障转移而丢失网络连接。如果连接管理器的 PRIORITY 属性设置为正整数，那么连接管理器将执行本地 IP 地址和其他集群计算机之间的网络监视。

PRIORITY 值在配置为管理特定集群的所有连接管理器之间必须是唯一的。如果 PRIORITY 值设置为正整数，那么 FOC 参数的 ORDER 属性必须设置为 ENABLED。

例如，在包含两个应用程序服务器的配置中，多个网络故障可能引起冲突，即故障转移到辅助数据库服务器会导致其中一个应用程序服务器丢失与数据库服务器的连接。由于故障转移会导致其中一个应用程序服务器丢失数据库连接，因此会产生冲突。

要解决这种情况，请在连接管理器配置的过程中设置每个应用程序服务器的优先级。如果为一个应用程序服务器配置的优先级高于另一个应用程序服务器，并且故障转移将导致丢失与优先级较低的应用程序服务器的连接，那么将允许数据库服务器的故障转移。如果故障转移导致丢失与优先级较高的应用程序服务器的连接，那么将不允许发生故障转移。

要监视连接管理器与数据库服务器之间的网络连接，连接管理器必须在与应用程序服务器相同的计算机上运行。

将连接管理器配置为在网络故障期间保持连接

要确保应用程序服务器保持与数据库服务器的连接，可以将连接管理器配置为在发生网络故障时启动故障转移。

- 连接管理器必须在应用程序服务器上运行。
- 必须至少有一个共享主服务器磁盘的 SD 辅助服务器。

要配置网络监视：

1. 如有必要，在每个应用程序服务器上至少安装两块网络接口卡 (NIC)。
2. 在每个应用程序服务器上执行以下步骤：
    a. 安装连接管理器软件。
    b. 在 sqlhosts 文件中创建数据库服务器和所有 SLA 的条目。
    c. 如有必要，使用 onpassword 实用程序创建密码文件并将其加密。
    d. 将 GBASEDBTDIR 环境变量设置为指向安装了连接管理器的目录。
    e. 在连接管理器配置文件的头中，将 LOCAL_IP 参数设置为计算机上 NIC 的可用 TCP/IP 地址。例如，如果应用程序服务器包含两个 NIC，其中 NIC1 的 TCP/IP 地址为为 9.25.151.14，NIC2 的 TCP/IP 地址为 9.25.151.15，那么参数条目为：

    ```
    LOCAL_IP 9.25.151.14,9.25.151.15
    ```

    f. 在连接管理器配置文件的主体中，配置服务级别协议 (SLA)。
    g. 在连接管理器配置文件的主体中，将 FOC 参数的 PRIORITY 属性设置为大于 0 的值。例如，要启用网络监视并设置最高优先级，请输入以下内容：

    ```
    PRIORITY=1
    ```

    h. 在连接管理器配置文件的主体中，将 FOC 参数的 ORDER 属性设置为 ENABLED。

    ```
    ORDER=ENABLED
    ```

3. 在每个数据库服务器上执行以下步骤：
    a. 在 sqlhosts 文件中，创建应用程序服务器和所有 SLA 的条目。
    b. 启动每个数据库服务器实例。
4. 启动每个连接管理器实例。

现在，如果发生网络故障，连接管理器将立即启动故障转移。连接管理器将确保在发生网络故障时，应用程序服务器可保持与数据库服务器的连接。

要在网络故障转移后返回到原始网络配置，请参阅集群故障转移、重定向和复原。

6.4.6 监视连接管理器

有工具可用于帮助诊断使用连接管理器引起的问题。

连接管理器日志文件中包含有关服务级别协议的信息、故障转移配置的信息和状态信息。启动连接管理器时，将显示日志文件的位置。

除了检查日志文件之外，还可以使用服务器随附的工具。可以通过以下方式监视连接管理器的状态：

- 使用 gstat 命令显示可用于对问题进行故障诊断的统计信息。
- 使用 GBase OpenAdmin Tool (OAT) 。

确定连接管理器的状态

可以使用 gstat 实用程序来显示有关活动连接管理器实例的信息。

使用 gstat -g cmsm 命令可显示连接到服务器实例的连接管理器守护程序，还可显示守护程序已处理的连接数。请参阅《GBase 8s 管理员参考》 以获取有关示例的更多信息。

以下代码是对单个活动连接管理器实例使用 gstat -g cmsm 命令生成的示例输出。

```
Connection Manager Name: argo
Hostname: argo

SLA          Connections   Service/Protocol   Rule
oltp         9             9593/onsoctcp      primary
oltp_ssl     4             9596/onsocssl      primary MODE=PROXY
report       0             9594/onsoctcp      SDS,HDR,RSS

Failover Configuration:

Connection Manager name   Rule            Timeout   State
argo                      SDS,HDR,RSS     10        Active Arbitrator, Primary is
up
```

以下代码是对两个活动连接管理器实例使用 gstat -g cmsm 命令生成的示例输出。

```
Connection Manager Name: reynolds
Hostname: argo

SLA          Connections   Service/Protocol   Rule
oltp         9             9593/onsoctcp      primary
oltp_ssl     4             9596/onsocssl      primary mode=proxy
report       0             9594/onsoctcp      SDS,HDR,RSS

Connection Manager Name: stimpson
Hostname: argo
```

```
SLA         Connections   Service/Protocol   Rule
oltp2       9             19903/onsoctcp     primary
report2     0             19904/onsoctcp     SDS,HDR,RSS

Failover Configuration:
Connection Manager name   Rule            Timeout   State
argo
                          SDS,HDR,RSS     10        Active Arbitrator, Primary is
up
argo2
                          SDS,HDR,RSS     0         Primary is up
```

6.4.7 停止连接管理器

不再希望执行连接重定向时，可停止连接管理器实例。

先决条件：

仅限 UNIX： 只有用户 gbasedbt 才能运行 oncmsm 命令。如果为用户 root 或 DBSA 组的成员授予了连接到 sysadmin 数据库的特权，那么用户 root 或 DBSA 组的该成员也可运行 oncmsm。

要停止连接管理器实例，请执行以下操作：

1. 登录到运行连接管理器实例的计算机。
2. 使用带 -k 选项的 oncmsm 实用程序。

```
oncmsm -k connection_manager_name
```

6.4.8 动态重新配置连接管理器

可以使用连接管理器的重新装入选项来添加或更改服务级别协议、故障转移参数、日志文件名、调试或其他选项。

先决条件：

仅限 UNIX： 只有用户 gbasedbt 才能运行 oncmsm 命令。如果为用户 root 或 DBSA 组的成员授予了连接到 sysadmin 数据库的特权，那么用户 root 或 DBSA 组的该成员也可运行 oncmsm。

可以编辑现有配置文件，或将原始文件替换为其中包含新参数的文件。新选项将立即生效。

要重新装入配置选项，请使用 -r 参数：

```
oncmsm -r connection_manager_name
```

connection_manager_name 是要重新装入的连接管理器实例的名称。因为同时可以有多个连接管理器实例处于活动状态，所以有必要指定连接管理器实例的名称。

提示： 使用 gstat 实用程序可确定连接管理器实例的名称：gstat -g cmsm

例如，要重新装入名称为 cm_1 的连接管理器实例的配置文件，请执行以下操作：

1. 使用新服务级别协议、故障转移参数、日志文件名或调试选项修改现有连接管理器配置文件。
2. 在运行连接管理器的计算机上，运行：oncmsm -r cm_1

   连接管理器将装入新参数。

6.4.9 将较旧格式的连接管理器配置文件转换为最新格式

GBase 8s Client Software Development Kit (Client SDK) V3.70.xC3 之前各版本中的连接管理器配置文件与当前版本的连接管理器不兼容。必须使用命令行选项转换 3.70.xC3 之前版本的配置文件。

先决条件：

- 在 UNIX™ 上，必须以用户 gbasedbt 或 root 身份登录才能转换配置文件。
- 必须具有对要转换的配置文件的读许可权。
- 必须具有对要创建的配置文件的写许可权。
- 如果转换多个配置文件，那么必须将每个文件中的 SLA 定义合并为一个配置文件。

即使连接管理器当前正在运行，也可以转换连接管理器的配置文件。

要将连接管理器文件转换为当前格式，请执行以下操作：

登录到运行连接管理器实例的计算机。

运行带 -n 选项的 oncmsm 实用程序，并指定新旧配置文件的路径和文件名。

如果连接管理器已在运行，那么必须重新启动连接管理器以使用新配置文件。

6.5 集群故障转移、重定向和复原

要保持可用性，必须计划在故障后对主服务器进行故障转移，从不可用的服务器重定向客户机连接，以及将集群复原到其原始配置。

6.5.1 故障转移配置

高可用性集群中发生故障意味着其中一个服务器不再可用。由于高可用性集群的目的是要保持可用性，因此必须计划在主服务器不可用时，集群中的某个辅助服务器成为主服务器。控制故障转移的最佳方式是配置连接管理器来执行到指定服务器的故障转移。

设置 DRAUTO 配置参数以指定如何执行故障转移，然后在现有主服务器发生故障时使用以下某种方法来创建新的主服务器：

- 连接管理器
- ISV 集群管理软件
- 手动转换

利用 ISV 集群管理软件进行故障转移

可以使用独立软件供应商 (ISV) 集群管理软件，而不是连接管理器来管理高可用性集群环境中的故障转移处理。

如果高可用性集群中的主服务器遇到问题，需要辅助服务器充当主服务器角色，那么在执行实际的故障转移之前，在发生故障的主服务器上禁止磁盘 I/O，而在新的主服务器上允许磁盘 I/O 非常重要。 另外，必须阻止对发生故障的主服务器的网络访问。 特别是对 SD 辅助服务器，如果未正确完成这些步骤，有可能损坏磁盘。

从高可用性集群环境中的服务器启用磁盘 I/O 操作的机制称为 I/O 防护。I/O 防护通过回调脚本进行配置。 如果主服务器发生故障，在辅助服务器充当主服务器角色之前，故障转移进程会在辅助服务器上执行回调脚本。 该脚本调用特定于 I/O 的任何命令，以便启用或禁用磁盘存取。该脚本启用对要充当主服务器的服务器上共享磁盘的写访问权，并禁用对发生故障的服务器上共享磁盘的写访问权。

可以通过 FAILOVER_CALLBACK 配置参数指定当数据库服务器从辅助服务器转换为主服务器，或从辅助服务器转换为标准服务器时，要运行的脚本名称。 $GBASEDBTDIR/etc 目录中提供了一个临时脚本，名称为 ifx_failover_callback.sh (UNIX™) 。 配置后，在辅助服务器转换为主服务器或标准服务器之前，将执行由 FAILOVER_CALLBACK 指定的脚本。

可根据高可用性集群的类型执行以下操作之一来测试故障转移脚本：

- 将 SD 辅助服务器转换为主服务器。
- 如果 DRAUTO 配置参数设置为 0，那么关闭主服务器并将 HDR 辅助服务器转换为标准方式。
- 如果 DRAUTO 配置参数设置为 1，那么关闭 HDR 对中的主服务器。
- 关闭远程独立集群中的主服务器，并将 RS 辅助服务器转换为标准方式。

online.log 中包含 Invoking Failover Callback 消息，运行故障转移脚本后，其中会列出该故障转移脚本的路径和文件名。

请参阅《GBase 8s 管理员参考》中有关 FAILOVER_CALLBACK 配置参数的信息。

如果 FAILOVER_CALLBACK 指定的脚本发生故障（即，如果它返回一个非零退出码），那么从辅助服务器到主服务器（或标准服务器）的故障转移也会失败。 这样的话，DBA 必须手动执行故障转移。

共享文件系统的 I/O 防护

可以配置 I/O 防护以在高可用性集群环境中保护共享资源。

软件或硬件故障可能导致未完成的操作写入到共享存储设备中。使用 I/O 防护可以隔离服务器，以防服务器访问共享存储。如果正在对高可用性集群中的服务器执行维护或测试，那么必须使用 I/O 防护。如果在服务器上或应用程序内检测到问题，集群管理器可检测问题，并阻止该服务器连接到共享数据。

可以配置一个脚本，以在主服务器出现故障时运行。 防护命令通过设置 FAILOVER_CALLBACK 配置参数来调用，启动故障转移时，该参数将运行脚本。

尽管无需 I/O 防护即可使用 GBase 8s 数据库软件，但仍然必须使用 I/O 防护来保护共享磁盘系统，以免遭受意外损失。

I/O 防护的类型

可使用若干类型的 I/O 防护，包括：

- 电源®防护 - 检测到问题时关闭节点电源。
- 光纤通道开关防护（需要 SCSI-3 持久性组保留）- 通过除去问题节点的保留来阻塞光纤通道设备上的端口。
- 实施 I/O 防护最常用的方法可能是使用光纤通道防护。光纤通道开关支持业界标准 SCSI-3 持久性组保留 (PR) 技术。 PR 技术允许一组系统临时注册到磁盘，并与包含数据的磁盘协调写互斥保留。

在大多数情况下，必须安装集群管理器软件。集群管理器软件提供将命令发出到光纤通道交换机所需的驱动程序和实用程序。例如，Linux™ Cluster Suite 提供名为 fence_scsi 的脚本。Sun Cluster 提供名为 scdidadm 的命令。

也可根据所用集群管理器软件和不同硬件能力来使用其他防护方法。

实施 I/O 防护

可在若干平台上配置 I/O 防护，包括：

- Linux
- Solaris
- AIX®

有关配置 I/O 防护的特定信息，请参阅设备制造商提供的文档。

集群故障

高可用性集群故障是指集群中数据库服务器之间的连接丢失，这可能是多种不同的情况造成的。

以下任一情况均可能导致集群故障：

- 一个数据库服务器的站点上发生灾难性故障（如火灾或大地震）
- 连接数据库服务器的联网电缆被破坏
- 一个数据库服务器上的处理中延迟过长
- 辅助数据库服务器上发生无法由镜像块解决的磁盘故障

提示： 集群故障不一定表示某个数据库服务器发生了故障，而只是表示数据库服务器之间的连接丢失。

数据库服务器会将以下任一情况解释为集群故障：

- 超过了指定的超时值。

  在正常的集群运行期间，数据库服务器预期集群中的其他数据库服务器进行通信确认。集群中的每个数据库服务器都有一个用于指定秒数的 ONCONFIG 参数 DRTIMEOUT。如果来自集群中另一数据库服务器的确认没有在 DRTIMEOUT 指定的秒数内返回，那么数据库服务器会假定发生了集群故障。
- 集群中的数据库服务器未响应通过网络进行的定期消息传递 (ping) 尝试。

  无论主数据库服务器是否向辅助数据库服务器发送任何记录，这两个数据库服务器均会互相执行 ping 操作。如果一个数据库服务器没有响应四次连续 ping 尝试，那么另一数据库服务器会假定发生了集群故障。

  集群中每个数据库服务器会在该数据库服务器上 DRTIMEOUT 参数指定的秒数过去之后，向集群中的其他数据库服务器发送 ping。

数据库服务器检测到集群故障后，它会向其消息日志写入一条消息（例如，DR: receive error）并关闭数据复制。如果发生了集群故障，那么两个数据库服务器之间的连接将断开，并且辅助数据库服务器将保持只读方式。

如果辅助数据库服务器在集群故障后仍保持联机状态，并且 DRAUTO 配置参数设置为 1 (RETAIN_TYPE)，那么该数据库服务器的类型将自动更改为标准。如果 DRAUTO 设置为 0 (off)，那么辅助数据库服务器将顶事尝试重新建立与主数据库服务器的通信。如果 DRAUTO 设置为 2 (REVERSE_TYPE)，那么一旦因原始主服务器发生故障（而非因原始主服务器重新启动）导致连接结束，辅助数据库服务器将立即成为主数据库服务器。

如果连接管理器已启用，那么将 DRAUTO 设置为 3 可防止高可用性集群中存在多个主服务器的可能性。 如果尝试使服务器联机以作为主服务器并且 DRAUTO=3，那么连接管理器将验证集群中没有其他活动的主服务器。如果另一台主服务器处于活动状态，那么连接管理器将拒绝该请求。

如果主数据库服务器发生故障，辅助数据库服务器可按以下方法运行：

- 辅助数据库服务器可保持处于逻辑恢复方式。在这种情况下，不采取任何操作。如

果您期望 HDR 的连接很快复原，那么这种情况是可取的。

- 辅助数据库服务器可以自动成为标准数据库服务器。此操作称为自动转换。
- 如果您使用手动切换来将数据库服务器方式更改为标准，那么辅助数据库服务器可以成为标准数据库服务器。

自动转换

自动转换意味着集群中的主服务器发生故障后，辅助数据库服务器将自动成为标准数据库服务器（如果 DRAUTO 配置参数设置为 1）或主数据库服务器（如果 DRAUTO 配置参数设置为 2）。

自动转换首先回滚所有打开的事务，然后作为主数据库服务器进入联机方式。仅当辅助数据库服务器的 onconfig 文件中的参数 DRAUTO 设置为 1 (RETAIN_TYPE) 或 2 (REVERSE_TYPE) 时，才会执行自动转换。

由于辅助数据库服务器成为标准或主数据库服务器，因此必须确保以下情况之一：

- 辅助数据库服务器有足够的逻辑日志磁盘空间来使处理什么继续进行而无需备份逻辑日志文件。
- 备份逻辑日志文件。

自动转换只会更改数据库服务器的类型。它不会将客户机应用程序重定向到辅助数据库服务器。

自动转换与手动转换相比有以下优势：

- 从主数据库服务器重定向到辅助数据库服务器的客户机可以继续写入数据和更新数据。
- 转换不会根据监视消息日志的操作程序来查看何时会发生高可用性数据复制故障，然后手动将辅助数据库服务器转换为标准数据库服务器。

自动转换需要非常稳定的网络才能正常运行。

在您成功使原始主数据库服务器恢复为联机状态时，将自动建立集群连接。

- 如果 DRAUTO 设置为 RETAIN_TYPE，那么辅助转为标准的数据库服务器将进行平稳关闭（以确保所有可能写入数据库服务器的客户机没有连接），然后切换回辅助数据库服务器。
- 如果 DRAUTO 设置为 REVERSE_TYPE，那么辅助转为主数据库服务器将直接切换为主类型。不会发生关闭。所有与此数据库服务器连接的应用程序可以保持连接状态。 原始主数据库服务器切换为辅助数据库服务器。

没有可靠网络时的自动转换

尽管自动转换可能是最佳解决方案，但是它不适合所有的环境。

如果主数据库服务器实际上并未发生故障，但是辅助数据库服务器向该主服务器注册失败，请考虑可能发生的情况。例如，如果当辅助数据库服务器由于网速慢或网络不稳定而向主数据库服务器发送信号时，辅助数据库服务器没有收到响应，那么辅助数据库服务器将假设主数据库服务器发生的故障并且自动切换为标准类型。如果当主数据库服务器向辅助数据库服务器发送信号时，主数据库服务器也没有收到响应，那么它将假设辅助数据库服务器发生故障并将关闭数据复制但是仍将保持联机方式。现在主数据库服务器和辅助数据库服务器（已切换为标准类型）都处于联机方式。

如果客户机可以独立地在两个数据库服务器上更新数据，那么对中的数据库服务器将处于以下状态：即每个数据库服务器具有另一数据库服务器所需的逻辑日志记录。 在此情况下，您必须重新启动并通过一整个数据库服务器的 0 级备份来执行初始数据复制，如首次启动 HDR 所述。 因此，如果网络不是十分稳定的话，那么您可能不会希望使用自动转换。无法在以前的辅助服务器上恢复 HDR，而不带任何丢失事务的风险。

手动转换

手动转换意味着辅助数据库服务器的管理员将辅助数据库服务器的类型更改为标准类型。

辅助数据库服务器回滚任何打开的事务，然后进入联机方式（如同标准数据库服务器一样），这样它就可以接受来自客户机应用程序的更新。

将脱机服务器连接到新的主服务器

故障转移之后，必须将所有脱机的辅助服务器重新连接到新的主服务器。

如果高可用性集群中的主服务器发生故障，系统将通知所有联机辅助服务器发生了该故障。辅助服务器将连接到新主服务器，并继续运行。 但是，故障转移时未联机的 RS 辅助服务器和 HDR 辅助服务器不会收到故障转移通知，因此这些服务器恢复联机时会尝试连接到原始（发生故障的）主服务器。在这种情况下，必须在这些辅助服务器上手动重置主服务器。

发生故障转移时在脱机的辅助服务器上运行以下命令。

- 对于 HDR 辅助服务器：

```
oninit -PHY
gadmin -d secondary new_primary
```

new_primary 指示当前主服务器的名称。

- 对于 RS 辅助服务器：

```
oninit -PHY
gadmin -d RSS new_primary
```

new_primary 指示当前主服务器的名称。

6.5.2 数据复制客户机的重定向和连接

当高可用性集群中的任何服务器变为不可用时，与该服务器的任何客户机连接都应重定向到可用的服务器。处理连接重定向的最佳方法是使用连接管理器将客户机连接自动重新连接到在服务级别协议 (SLA) 中指定的服务器。或者，可以使用环境变量或连接信息来控制连接重定向。

如果不使用连接管理器，那么可以通过将应用程序配置为连接到服务器所属的服务器组，自动将客户机重定向到集群中其他数据库服务器。 创建与服务器组的连接时，缺省情况下将建立与组中当前主服务器的连接。如果因为某个服务器发生故障而导致复制失败，那么将建立与联机服务器的连接（以标准方式或不带辅助服务器的主服务器方式）。您还可以从应用程序内自动执行此操作。 在 GBase 8s Client Software Development Kit (Client SDK) 中包含的某些客户机连接驱动程序有特定机制以用于使重定向自动化。有关详细信息，请参阅《GBase 8s Client Software Development Kit (Client SDK)》文档。

当您设计客户机应用程序时，您必须对重定向策略作出一些决策。您尤其必须决定是否在应用程序内处理重定向以及要使用哪种重定向机制。三种不同的重定向机制如下所示：

- 用 DBPATH 环境变量进行自动重定向
- 管理员用连接信息控制重定向
- 用户用 GBASEDBTSERVER环境变量控制重定向

您所用的机制决定您可在应用程序中使用的 CONNECT 语法。

使用 DBPATH 环境变量自动重定向客户机

可以在应用程序中使用 DBPATH 环境变量来重定向连接。

管理员必须执行的操作

管理员不会执行任何操作来重定向客户机，但他们可能需要照管数据库服务器的类型。

用户的任务

如果您的应用程序包含测试连接是否已失败的代码，并发出重新连接语句（如有必要），那么重定向得以自动处理。用户没有任何职责。

如果您的应用程序不包含这样的代码，那么正在运行客户机的用户必须退出并重新启动所有应用程序。

DBPATH 重定向方法的工作原理

当应用程序没有在 CONNECT 语句中明确指定数据库服务器，以及 GBASEDBTSERVER 环境变量指定的数据库服务器不可用时，客户机将使用 DBPATH 环境变量来定位数据库（和数据库服务器）。

如果复制对中的一台数据库服务器不可用，那么使用该数据库服务器的应用程序将其 DBPATH 环境变量设置为复制对中的另一数据库服务器时，这些应用程序无需重新设置

其 GBASEDBTSERVER 环境变量。它们的 GBASEDBTSERVER 环境变量必须始终包含其经常使用的数据库服务器的名称，并且它们的 DBPATH 环境变量必须始终包含对中备用数据库服务器的名称。

例如，如果应用程序通常使用名为 cliff_ol 的数据库服务器，并且在复制对中与 cliff_ol 成对的数据库服务器名为 beach_ol，那么那些应用程序的环境变量将如下所示：

```
GBASEDBTSERVER   cliff_ol
DBPATH           //beach_ol
```

因为当应用程序发出 CONNECT 语句时 DBPATH 环境变量是只读变量（如果需要），所以应用程序必须重新启动才能执行重定向。

应用程序可以包含测试连接是否已失败的代码，如果包含，那么尝试重新连接。如果应用程序有该代码，就无需重新启动。

您可以对该重定向方法使用 CONNECT TO 数据库语句。为使该方法可起作用，您不能使用任一以下语句：

- CONNECT TO DEFAULT
- CONNECT TO database@dbserver
- CONNECT TO @dbserver

该限制的原因是如果 CONNECT 语句指定了数据库服务器，应用程序就不使用 DBPATH。有关 DBPATH 的更多信息，请参阅《GBase 8s SQL 指南：参考》。

使用连接信息重定向客户机

通过使用 sqlhosts 信息，可以将客户机连接重定向到新的主服务器。

连接信息重定向方法依赖于这样的事实，即当应用程序连接到数据库服务器，它就使用连接信息找到数据库服务器。

如果复制对中的一个数据库服务器不可用，管理员可在连接信息中更改不可用数据库服务器的定义。按更改客户机连接信息中所述，不可用数据库服务器的字段（dbservername 字段除外）均更改为指向复制对中余下的数据库服务器。

因为当发出 CONNECT 语句时读取了连接信息，所以应用程序可能需要重新启动才能执行重定向。应用程序可以包含用于测试连接是否已失败的代码，也可以包含用于发出重新连接语句的代码（如有必要）。如果连接已失败，将自动进行重定向，并且您无需重新启动应用程序即可进行重定向。

应用程序可使用以下连接语句来支持该重定向方法：

- CONNECT TO database@dbserver
- CONNECT TO @dbserver

如果 GBASEDBTSERVER 环境变量始终保持设置为同一数据库服务器名称并且 DBPATH 环境变量未设置，那么应用程序也可使用以下连接语句：

- CONNECT TO DEFAULT

- CONNECT TO database

在 UNIX™ 上，GBASEDBTSQLHOSTS 环境变量在 $GBASEDBTDIR/etc/sqlhosts 中指定连接信息的完整路径名和文件名。有关 GBASEDBTSQLHOSTS 的更多信息，请参阅《GBase 8s SQL 指南：参考》。

更改客户机连接信息

要使用连接信息来重定向客户机，您必须为客户机更改连接信息并更改其他连接文件（如有必要）。

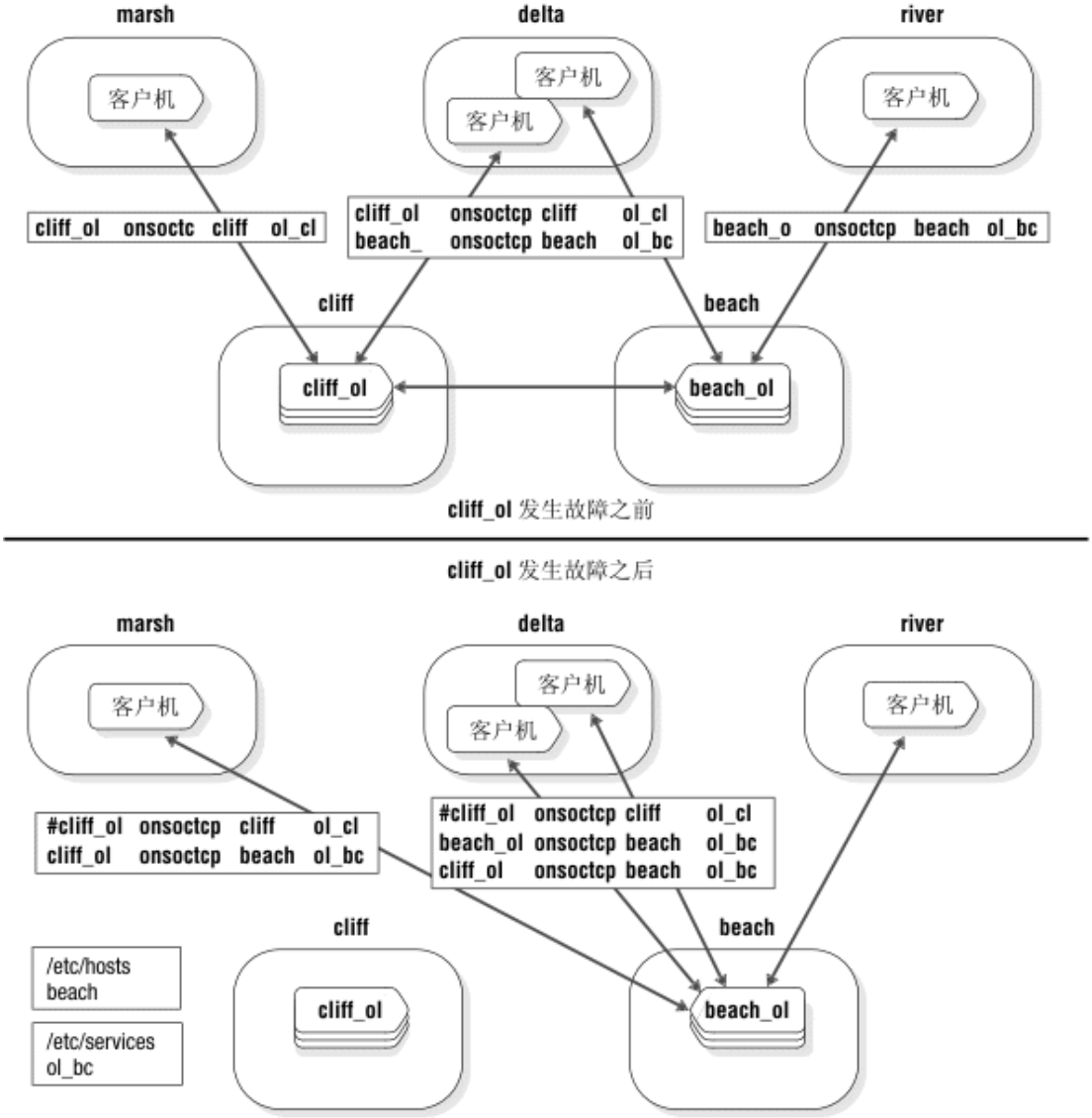
要更改有关客户机的连接信息，请执行以下操作：

1. 在 sqlhosts 文件或注册表中，注释掉发生故障的数据库服务器的条目。
2. 在 servername 字段中添加指定故障数据库服务器的 dbservername 的条目，并在 nettype、hostname 和 servicename 字段中添加客户机要重定向的目标数据库服务器的信息。
3. 如果发生故障，那么在 sqlhosts 文件或注册表中使用以下选项将应用程序重定向至其他数据库服务器：
   a. 连接重定向选项
   b. 组末尾选项
   c. 组选项
4. 根据需要，在 /etc/hosts 文件（在 UNIX™ 上）中添加 hostname 条目，以指定正在运行作为客户机重定向目标的数据库服务器的计算机。
5. 根据需要，在 /etc/services 文件（在 UNIX 上）添加 servicename 条目，以指定正在运行作为客户机重定向目标的数据库服务器的计算机。

下图显示可以如何修改连接值以重定向客户机。

无需在正运行数据库服务器的任一计算机上更改连接信息中的条目。

图： cliff_ol 数据库服务器发生故障前后的连接值



连接到数据库服务器

在管理员更改连接信息和其他连接文件（如果需要）后，客户机在发出其下一个 CONNECT 语句时会连接至管理员要将客户机重定向到的目标数据库服务器。

如果您的应用程序包含测试连接是否已失败的代码，并发出重新连接语句（如有必要），那么重定向得以自动处理。用户没有任何职责。如果您的应用程序不包含这样的代码，那么正在运行客户机的用户必须退出并重新启动所有应用程序。

服务器组的自动重定向

可以使用 sqlhosts 文件中的组选项来指定应用程序连接到的服务器组而不是单个数据库服务器。要使连接重定向自动化，请将主服务器和辅助服务器的数据库服务器定义均添加到服务器组定义。缺省情况下，建立了到 HDR 服务器组的连接请求时，会将该连接路由到主服务器。如果主服务器不可用，那么连接请求将路由到故障转移处理之后提升为主服务器的辅助服务器。

例如，以下 sqlhosts 条目提供了 HDR 服务器组 g_hdr，以及主服务器定义 hdr_prim 和辅助服务器定义 hdr_sec。

```
#dbservername      nettype        hostname        servicename         options
g_hdr              group           -               -                   i=1
hdr_prim           ontlitcp        machine1pri     port1               g=g_hdr
hdr_sec            ontlitcp        machine1sec     port1               g=g_hdr
```

应用程序可使用以下连接语句来支持该重定向方法：

- CONNECT TO database@dbserver_group
- CONNECT TO @dbserver_group

如果您的应用程序包含测试连接是否已失败的代码，并发出重新连接语句（如有必要），那么重定向得以自动处理。用户没有任何职责。如果您的应用程序不包含这样的代码，那么正在运行客户机的用户必须退出并重新启动所有应用程序。

使用 GBASEDBTSERVER 环境变量重定向客户机

应用程序未在 CONNECT 语句中显式指定数据库服务器时，可以使用 GBASEDBTSERVER 环境变量重定向方法，以便客户机连接至 GBASEDBTSERVER 环境变量指定的数据库服务器。

如果集群中的某个数据库服务器不可用，那么使用该数据库服务器的应用程序可将其 GBASEDBTSERVER 环境变量重置为集群中的另一数据库服务器以访问相同数据。

应用程序仅当它们启动时才读取 GBASEDBTSERVER 环境变量的值。因此，必须重新启动应用程序以识别环境变量中的更改。

要支持该重定向方法，您可以使用以下连接语句：

- CONNECT TO DEFAULT
- CONNECT TO database

您不能为该方法使用 CONNECT TO database@dbserver 或 CONNECT TO @dbserver 语句。当数据库服务器已明确命名，CONNECT 语句不使用 GBASEDBTSERVER 环境变量来查找数据库服务器。

管理员不会执行任何操作来重定向客户机，但他们可能需要更改数据库服务器的类型。

正在运行客户机应用程序的用户在决定用 GBASEDBTSERVER 环境变量重定向客户机时必须执行以下三个步骤。

要使用 GBASEDBTSERVER 环境变量重定向客户机，请执行以下操作：

1. 退出应用程序。
2. 将 GBASEDBTSERVER 环境变量更改为保存复制对中另一数据库服务器的名称。
3. 重新启动应用程序。

使用应用程序代码重定向客户机

如果使用 DBPATH 环境变量或连接信息来重定向连接，那么可以在客户机中包含一个例程，用于在客户机遇到集群故障时处理错误。该例程可以调用包含循环（该循环重复尝试连接到集群中的另一个数据库服务器）的其他函数。该例程重定向客户机但不需要用户退出应用程序并重新启动程序。

以下示例显示的是在使用 DBPATH 重定向机制的客户机应用程序中，在尝试重新连接时会循环的函数。在它建立了连接后，它还会测试数据库服务器的类型以确保服务器不是辅助数据库服务器。如果数据库服务器仍为辅助类型，那么它调用另一函数来提醒用户（或数据库服务器管理员）数据库服务器无法接受更新。

```
/* The routine assumes that the GBASEDBTSERVER environment
 * variable is set to the database server that the client
 * normally uses and that the DBPATH environment variable
 * is set to the other database server in the pair.
 */

#define SLEEPTIME 15
#define MAXTRIES 10

main()
{
    int connected = 0;
    int tries;
    for (tries = 0;tries < MAXTRIES && connected == 0;tries++)
    {
        EXEC SQL CONNECT TO "superstores";
        if (strcmp(SQLSTATE,"00000"))
        {
            if (sqlca.sqlwarn.sqlwarn6 != 'W')
            {
                notify_admin();
                if (tries < MAXTRIES - 1)
                    sleep(SLEEPTIME);
                }
            else connected =1;
            }
        }
        return ((tries == MAXTRIES)? -1:0);
    }
```

本示例假设 DBPATH 重定向机制并使用支持 DBPATH 重定向方法的 CONNECT 语句格式。 如果您使用连接信息来重定向，您可以有不同的连接语句，如下所示：

```
EXEC SQL CONNECT TO "superstores@cliff_ol";
```

在本示例中，superstores@cliff_ol 是客户机计算机识别的数据库服务器上的数据库。 要进行重定向，管理员必须更改连接信息，以使该名称指向不同的数据库服务器。 可能需要调整客户机在尝试连接前等待的时间量或函数定义的尝试次数。为数据库服务器上的管理操作提供足够时间（以更改连接信息或将辅助数据库服务器的类型更改为标准类型）。

比较重定向方法

不同的重定向方法具有不同的需求。

以下各表总结了重定向机制之间的不同之处。

表 1. DBPATH 连接的重定向方法

DBPATH	自动重定向	用户重定向
何时重定向客户机？	当客户机下次尝试连接指定数据库时	当客户机下次尝试连接指定数据库时
客户机是否需要重新启动以进行重定向？	否	是
重定向的范围是哪些？	重定向个别客户机	重定向个别客户机
需要对环境变量进行更改吗？	否	否

表 2. 连接信息的重定向方法

连接信息	自动重定向	用户重定向
何时重定向客户机？	在管理员更改连接信息后，当客户机下次尝试建立与数据库服务器的连接时	在管理员更改连接信息后，当客户机下次尝试建立与数据库服务器的连接时
客户机是否需要重新启动以进行重定向？	否	是
重定向的范围是哪些？	重定向所有使用给定数据库服务器的客户机	重定向个别客户机
需要对环境变量进	否	否

行更改吗？		

表 3. GBASEDBTDIR 连接的重定向方法

GBASEDBTDIR	用户重定向
何时重定向客户机？	当客户机重新启动并读取 GBASEDBTSERVER 环境变量的新值时
客户机是否需要重新启动以进行重定向？	是
重定向的范围是哪些？	重定向个别客户机
需要对环境变量进行更改吗？	是

表 4. “连接管理器”连接的重定向方法

连接管理器	用户重定向
何时重定向客户机？	当获得已配置的服务级别协议时。
客户机是否需要重新启动以进行重定向？	否
重定向的范围是哪些？	重定向个别客户机
需要对环境变量进行更改吗？	否

6.5.3 故障后恢复 HDR 和 RS 集群

将 HDR 或 RS 集群复原回原始配置时，可能需要复原关键介质，以及重新启动和配置集群中的服务器。

磁盘故障的结果取决于磁盘故障发生在主数据库服务器上还是辅助数据库服务器上、磁盘上的块是否包含关键介质（根数据库空间、逻辑日志文件或物理日志）以及块是否已镜像。

如果块已镜像，您可以执行恢复，如同您为使用镜像的标准数据库服务器所做的一样。

如果块未镜像，那么复原主数据库服务器的过程取决于发生故障的磁盘是否包含关键介质：

- 如果磁盘中包含关键介质，那么主数据库服务器将发生故障。必须使用主数据库空间备份（或如果辅助数据库服务器切换为标准方式并重定向了活动，那么使用辅助数据库空间备份）执行完全复原。
- 如果磁盘不包含关键介质，您可以用热复原来分别复原受影响的数据库空间。热复原由两个部分组成：首先从备份对发生故障的数据库空间进行复原，随后对自该数

据库空间备份以来所有写入的逻辑日志记录进行逻辑复原。您必须在执行热复原之前备份所有逻辑日志文件。

在块未镜像的情况下，如果磁盘包含关键介质，辅助数据库服务器将发生故障；但如果磁盘不包含关键介质，辅助数据库服务器将保持联机状态。在这两种情况下，都必须在主数据库服务器上使用数据库空间备份执行完全复原。在第二种情况下，您无法从辅助数据库空间备份复原所选的数据库空间，因为此时它们可能是从主数据库服务器上的相应数据库空间派生而来的。您必须执行完整的复原。

关键数据损坏后恢复集群

如果 HDR 或 RS 集群中的某个数据库服务器遇到损坏根数据库空间（包含逻辑日志文件或物理日志的数据库空间）的故障，那么必须将发生故障的数据库服务器视为在磁盘上没有任何数据，并且假定您是首次启动 HDR 或 RS 集群。 将正在运行的带有完整磁盘的数据库服务器作为带有数据的数据库服务器来使用。

主服务器故障

对于以下步骤，假设配置中包含名为 srv_A 的主服务器和名为 srv_B 的 HDR 辅助服务器。重新启动 RS 集群的步骤与此类似。

要在严重介质故障后重新启动 HDR，请执行以下操作：

1. srv_B 上的 DRAUTO 配置参数会影响您下一步的操作
   - 如果此参数设置为 0，那么必须通过运行 gadmin -d make primary 命令将服务器转换为主服务器。
   - 如果此参数设置为 1 (RETAIN_TYPE)，那么通过运行 gadmin -d make primary 命令将服务器转换为主服务器。
   - 如果设置为 2 (REVERSE_TYPE)，那么一旦因旧的主服务器发生故障导致连接结束，辅助数据库服务器将立即成为主数据库服务器。
2. 从上次数据库空间备份复原 srv_A（主数据库服务器）。
3. 使用 gadmin -d 命令将 srv_A 设置为 HDR 辅助数据库服务器并启动 HDR。

   gadmin -d 命令可在 srv_B 上从逻辑日志文件启动逻辑恢复。如果由于已在 srv_B上备份并释放了逻辑日志文件而无法完成逻辑恢复，那么 HDR 要到执行下一步时才会启动。
4. 应用来自 srv_B 的逻辑日志文件（新的主数据库服务器），这些文件已备份到磁带。HDR 对现在可运行；但是，将交换 srv_A 和 srv_B 的角色。要将 srv_A 和 srv_B 交换回其原始角色，请遵循指示信息：在辅助服务器成为主服务器后恢复 HDR 集群。

表 1. 主数据库服务器上发生严重介质故障后恢复 HDR 的步骤

步骤	在主数据库服务器（svr_A）上	在辅助数据库服务器（svr_B）上

1.		gadmin 命令 gadmin -d make primary srv_A
2.	gtape 命令 gtape -p ON-Bar 命令 gbackuprestore -r -p	
3.	gadmin 命令 gadmin -d secondary srv_B	
4.	gtape 命令 gtape -l ON-Bar 命令 gbackuprestore -r -l	

辅助服务器故障

如果辅助数据库服务器遇到严重介质故障，请遵循首次启动集群的步骤来恢复集群。

主服务器和辅助服务器故障

如果正在运行复制对中数据库服务器的两台计算机不幸同时遇到损坏根数据库空间（包含逻辑日志文件或物理日志的数据库空间）的故障，那么必须重新启动集群。

要在两个数据库服务器上都发生严重介质故障后重新启动 HDR 或 RS 集群：

1. 从存储空间和逻辑日志备份复原主数据库服务器。
2. 复原主数据库服务器后，请将另一台发生故障的数据库服务器视为在磁盘上没有任何数据，并且假定您是首次启动 HDR 或 RS 集群。

网络故障后重新启动 HDR 或 RS 集群

发生网络故障之后，HDR 或 RS 集群可能需要重新启动。在网络故障后，主数据库服务器处于联机方式，辅助数据库服务器处于只读方式。复制在两个数据库服务器上都会关闭 (state = off)。

可能不需要重新启动集群，因为主数据库服务器会尝试每 10 秒重新连接一次，并且每 2 分钟显示一次有关无法连接的消息。

如果集群未自动重新启动，那么重新建立连接时，可以通过在辅助服务器上运行以下命令来重新启动集群：

- gadmin -d secondary primary_name，以使该辅助服务器成为主服务器。

辅助服务器发生故障时重新启动 HDR 或 RS 集群

如果在辅助服务器发生故障后必须重新启动 HDR 或 RS 集群，那么除了启动该辅助服务器外，还可能需要在该辅助服务器上执行逻辑日志复原。

这些步骤假设您自辅助数据库服务器发生故障后一直按需要备份主数据库服务器上的逻辑日志文件。

表 1. 辅助数据库服务器上发生故障后的重新启动步骤

步骤	在主服务器上	在辅助服务器上
1.	主数据库服务器必须处于联机方式。	oninit 如果您在消息日志中接收到以下消息，请继续步骤 2 ： DR: Start Failure recovery from tape
2.		gtape 命令 gtape -l ON-Bar 命令 gbackuprestore -r -l

在辅助服务器成为主服务器后恢复 HDR 集群

如果在原始主服务器发生故障后，HDR 集群中的辅助服务器成为主服务器，那么可以使用脚本来重新建立原始主服务器，然后将当前主服务器转换回辅助服务器。

假设主服务器 srv_pri 遇到了错误，以至于要故障转移至辅助服务器 srv_hdr_sec。此时，主服务器是 srv_hdr_sec，并且集群中的所有其他辅助服务器现在都指向 srv_hdr_sec。

要将集群复原至 srv_pri 执行故障转移之前的状态，请遵循以下步骤：

1. 通过运行适当命令将 srv_pri 初始化为 HDR 辅助服务器：

   UNIX™ 系统：

   ```
   $GBASEDBTDIR/bin/hdrmksec.sh srv_hdr_sec
   ```

2. 通过运行以下命令将 srv_pri 更改为主服务器：

   ```
   gadmin -d make primary srv_pri
   ```

   此命令将使 srv_pri 成为主服务器，并将集群中的其他任何辅助服务器重定向为指向新的主服务器。 该命令也会关闭原有 HDR 主服务器 (srv_hdr_sec)，这是因为高可用性环境中只能存在一个主服务器。

3. 通过运行以下命令将 srv_hdr_sec 初始化为 HDR 辅助服务器：

   在 UNIX 系统上：

   ```
   $GBASEDBTDIR/bin/hdrmksec.sh srv_pri
   ```

主服务器发生故障时重新启动

在主服务器发生故障后重新启动 HDR 或 RS 集群的过程取决于是否辅助服务器是否成为主服务器，以及该辅助服务器成为主服务器的方法。

辅助数据库服务器未更改为标准数据库服务器

如果主数据库服务器发生故障后必须重新启动 HDR 或 RS 集群，并且辅助数据库服务器没有更改为标准数据库服务器，请使用 oninit 命令启动主数据库服务器。

辅助数据库服务器已手动更改为标准数据库服务器

如果主数据库服务器发生故障后必须重新启动 HDR 或 RS 集群，并且您已将辅助数据库服务器手动更改为标准数据库服务器，请完成下表中的步骤。

表 1. 将辅助数据库服务器更改为标准服务器后的重新启动步骤

步骤	在主数据库服务器上	在辅助数据库服务器上
1.		gadmin -s 该步骤将辅助数据库服务器（现为标准服务器）变为静默方式。所有连接至该数据库服务器的客户机必须断开连接。执行更新的应用程序必须重定向到主服务器。
2.		gadmin -d

步骤	在主数据库服务器上	在辅助数据库服务器上
		secondary prim_name
3.	oninit 如果所有写入辅助数据库服务器的逻辑日志记录仍在辅助数据库服务器磁盘上，那么当您发出 oninit 命令时主数据库服务器将从该磁盘恢复这些记录。 如果您已备份并释放辅助数据库服务器上的逻辑日志文件，那么这些文件中的记录不再在磁盘上。在这种情况下，会提示您从磁带恢复这些逻辑日志文件（步骤 4）。 对于 gtape 用户： 如果您希望通过网络读取逻辑日志记录，请将逻辑日志磁带设备设置为正在运行辅助数据库服务器的计算机上的设备。	
4.	如果提示您从磁带恢复逻辑日志记录，请执行此步骤。 gtape 命令 gtape -l ON-Bar 命令 gbackuprestore -r -l	

辅助数据库服务器已自动更改为标准数据库服务器

如果主数据库服务器发生故障后必须重新启动 HDR 或 RS 集群，并且辅助数据库服务器已自动更改为标准数据库服务器，请完成下表中所示的步骤。

表 2. 将辅助数据库服务器更改为标准服务器后的自动重新启动步骤

步骤	在主数据库服务器上	在辅助数据库服务器上
1.	% oninit 如果 DRAUTO = 1，那么此数据库服务器的类型将设置为主数据库服务器。 如果 DRAUTO = 2，那么当此数据库服务器重新启动时，其类型将设置为辅助数据库服务器。 如果所有写入辅助数据库服务器的逻辑日志记录仍在辅助数据库服务器磁盘上，那么当您发出 oninit 命令时主数据库服务器将从该磁盘恢复这些记录。 如果您已备份和释放的逻辑日志文件位于辅助数据库服务器上，那么这些文件中的记录将不再位于磁盘之上。在这种情况下，会提示您从磁带恢复这些逻辑日志文件（步骤 2）。 对于 gtape 用户： • 将逻辑日志磁带设备设置为正在运行辅助数据库服务器的计算机上的设备。	如果 DRAUTO = 1 ，那么当您进行主备份时，辅助数据库服务器将自动平稳关闭。这可确保断开所有客户机的连接。然后，类型切换回辅助。执行更新的所有应用程序必须重定向到主数据库服务器。 如果 DRAUTO = 2，那么辅助数据库服务器将自动转换为主数据库服务器。在旧的主数据库服务器重新启动并与其他服务器连接，并确定它现在是主数据库服务器之后，它将成为辅助数据库服务器。
2.	如果提示您从磁带恢复逻辑日志记录，请执行此步骤。 gtape 命令 % gtape -l ON-Bar 命令 gbackuprestore -r -l	

6.5.4 数据损坏后恢复共享磁盘集群

如果共享磁盘集群发生故障，必须对受影响的数据库空间执行复原。需要执行的复原类型取决于是否损坏了关键数据。

关键数据已损坏

如果主服务器发生了故障，导致根数据库、包含逻辑日志文件的数据库空间或包含物理日志的数据库空间损坏，那么必须将失败的数据库服务器视为其磁盘上没有数据。 必须对主服务器执行完全复原。在这种情况下，主服务器和 SD 辅助服务器处于脱机状态。

要在介质出现严重故障之后恢复共享磁盘集群，请执行以下操作：

1. 对主服务器执行完全复原。根据备份是使用 ON-Bar 还是 gtape 实用程序执行的，运行以下某个命令：
   - gbackuprestore -r
   - gtape -r

   复原完成之后，主服务器将重新启动。
2. 重新启动 SD 辅助服务器。

此外，也可以对主服务器上的关键数据库空间执行冷复原，重新启动 SD 辅助服务器，然后对非关键数据库空间执行热复原。

关键数据未损坏

如果不包含关键介质的磁盘出现故障，可以使用热复原来复原受影响的数据库空间。在这种情况下，主服务器和 SD 辅助服务器处于联机状态。

要恢复共享磁盘集群中的非关键数据，请执行以下操作：

1. 关闭并重新启动 SD 辅助服务器。
2. 对受影响的数据库空间执行热复原。根据备份是使用 ON-Bar 还是 gtape 实用程序执行的，运行以下某个命令：
   - gbackuprestore -r 加上要复原的数据库空间的名称
   - gtape -r -D 加上要复原的数据库空间的名称

6.5.5 在辅助服务器成为主服务器后恢复 SD 集群

如果在原始主服务器发生故障后，SD 集群中的辅助服务器成为主服务器，那么可以使用脚本来重新建立原始主服务器，然后将当前主服务器转换回辅助服务器。

在该示例中，主服务器 srv_pri 故障转移至 SD 辅助服务器 srv_sds_sec。此时，主服务器是 srv_sds_sec，并且集群中的所有辅助服务器现在都指向 srv_sds_sec。要将集群复原至 srv_pri 执行故障转移之前的状态，请遵循以下步骤：

1. 如果必要，请在 srv_pri 的 onconfig 文件中设置以下参数：

```
SDS_ENABLE 1
```

```
SDS_PAGING <path 1>,<path 2>
SDS_TEMPDBS <dbsname>,<dbspath>,<pagesize>,<offset>,<size>
```

dbsname 值必须唯一。此外，dbsname 必须在所有现有的数据库空间、BLOB 空间和智能大对象空间空间中唯一，包括从主服务器继承的临时空间（可能已禁用）。如果有多个 SD 辅助服务器，dbsname 值对于每个服务器必须唯一，且不得与其他任何 SD 辅助服务器或主服务器共享。请参阅设置共享磁盘辅助服务器，以获取有关设置这些参数的信息。

2. 通过在 srv_pri 上运行 oninit 命令，将 srv_pri 初始化为 SD 辅助服务器。
3. 手动对 srv_pri 执行故障转移以便使其成为主服务器：

```
gadmin －d make primary srv_pri
```

以上命令从集群中除去 srv_sds_sec，并使 srv_pri 成为主服务器。

4. 通过在 srv_sds_sec 上运行 oninit 命令，将 srv_sds_sec 复原为 SD 辅助服务器。

7 分布式数据

7.1 多阶段落实协议

两阶段落实协议确保跨多个数据库服务器统一落实或回滚事务。您可以将 GBase 8s 数据库服务器与 GBase 8s Enterprise Gateway 产品或事务管理器一起使用，以便在非 GBase 8s 数据库中处理数据。跨多个 GBase 8s 数据库服务器的分布式查询支持两阶段落实。

异类落实协议确保对单一事务中的一个或多个 GBase 8s 数据库以及一个非 GBase 8s 数据库的更新可统一落实或回滚。

这些主题包含有关使用两阶段落实协议的信息。 有关从失败的两阶段落实事务手动恢复的信息，请参阅从失败的两阶段落实手动恢复。

这些主题还包含有关使用支持符合 XA 的外部数据源的事务的信息，这些数据源可参与两阶段落实事务。请参阅 GBase 8s 事务对符合 XA 的外部数据源的支持。

7.1.1 事务管理器

事务管理器支持两阶段落实和回滚。例如，如果数据库是 GBase 8s，记帐系统是 Oracle，而汇款系统是 Sybase，那么可以使用事务管理器在不同数据库之间通信。 您还可以使用事务管理器通过使用分布式事务而非 Enterprise Replication 或高可用性数据复制，确保 GBase 8s 或非 GBase 8s 数据库之间的数据一致性。

TP/XA 库（带事务管理器）

全局事务是一种在查询中涉及多个数据库服务器的分布式查询。 全局事务环境具有以下部件：

- 客户机应用程序
- 资源管理器（GBase 8s 数据库服务器）
- 事务管理器（供应商软件）

TP/XA 是一个函数库，使数据库服务器可充当 X/Open DTP 环境中的资源管理器。将 TP/XA 库作为 GBase 8s ESQL/C 的一部分安装可启用第三方事务管理器和数据库服务器之间的通信。X/Open 环境支持大型的高性能 OLTP 应用程序。

当您的数据库具有以下特征时，请使用 TP/XA：

- 数据分发在多供应商数据库中
- 事务包括 GBase 8s 和非 GBase 8s 数据

Microsoft Transaction Server (MTS/XA)

数据库服务器支持 Microsoft™ Transaction Server (MTS/XA) 在 XA 环境中充当事务管理器。 要使用 MTS/XA，请安装 GBase 8s Client Software Development Kit、最新版本的 GBase 8s ODBC Driver 以及 MTS/XA。 有关更多信息，请联系 GBase 8s 技术支持，并参阅 GBase 8s 客户机产品安装指南 和 MTS/XA 文档。

GBase 8s 事务对符合 XA 的外部数据源的支持

GBase 8s 事务管理器 GBase 8s 的主体部分，而不是单独模块，用于识别符合 XA 的外部数据源。这些数据源可参与两阶段落实事务。

对于参与特殊事务事件（例如，准备、落实或回滚）的分布式事务每个符合 XA 的外部数据源，事务管理器都会对其运行支持例程。该交互符合 X/Open 接口标准。

符合 XA 的外部数据源的事务支持（也称为资源管理器）使您能够执行以下操作：

- 创建符合 XA 的外部数据源类型及其实例。
- 创建或修改用户定义的例程 (UDR)、虚拟表接口或虚拟索引接口以启用符合 XA 的数据源来为来自符合 XA 的数据源的外部数据提供数据访问机制。
- 将符合 XA 的外部数据源注册到 GBase 8s。
- 注销符合 XA 的外部数据源。
- 在同一个全局事务中使用多个符合 XA 的外部数据源。

事务与符合 XA 的外部数据源的协调仅在以 GBase 8s 登录的数据库和符合 ANSI 标准的数据库中受支持，因为这些数据库都支持事务。事务与符合 XA 的外部数据源的协调在未登录的数据库中不受支持。

可使用以下 DDL 语句，它们是用于管理 XA 数据源类型和数据源的 SQL 语句的扩展：

语句	描述

语句	描述
CREATE XADATASOURCE TYPE	创建符合 XA 的外部数据源类型
CREATE XADATASOURCE	创建符合 XA 的外部数据源实例
DROP XADATASOURCE	删除符合 XA 的外部数据源实例
DROP XADATASOURCE TYPE	删除符合 XA 的外部数据源类型

有关这些语句的更多信息，请参阅《GBase 8s SQL 指南：语法》 。

GBase 8s 和符合 XA 的外部数据源之间的交互通过一组用户定义且支持 XA 的例程（例如，xa_open、xa_end、xa_commit 和 xa_prepare）执行。可在使用 CREATE XADATASOURCE TYPE 语句之前创建这些支持例程。

在创建了符合 XA 的外部数据源之后，可使用 mi_xa_register_xadatasource()（或 ax_reg()）和 mi_xa_unregister_xadatasource()（或ax_unreg()）函数将该数据源注册到当前事务和注销该数据源。在分布式环境中，必须在本地协调者服务器上注册数据源。注册是瞬态的，持续时间仅仅是该事务的持续时间。

使用以下 gstat 选项可显示有关包含符合 XA 的数据源的事务的信息：

gstat 选项	此命令显示的符合 XA 的数据源信息
gstat -x	显示有关事务中的 XA 参与者的信息。
gstat -G	显示有关全局事务中的 XA 参与者的信息。
gstat -g ses session id	显示会话信息，包括有关参与事务的 XA 数据源的信息。

高可用性集群中的 XA

X/Open 分布式事务处理 (DTP) 模型允许高可用性集群中的可更新辅助服务器在分布式事务中充当资源管理器。

任何 XA 全局事务中的参与者有三种：

- 应用程序 (AP)：定义事务界限，并指定构成事务分支的操作。
- 资源管理器 (RM)：提供对资源（如数据库）的访问。
- 事务管理器 (TM)：为事务分支指定标识 (XID)，监视事务进度，以及协调事务分支以完成操作和实现故障恢复。

在高可用性集群中，会话可以从集群中的任何服务器创建事务分支或附加到事务分支。例如，可从 server_2 附加已从 server_1 分离的事务分支。应用程序可在不跟踪启动事务的服务器的情况下，使用连接管理器连接到集群。事务管理器也可以使用连接管理器连接到

集群，以完成 XA 事务，方法是对松散和紧密耦合的事务均执行落实、回滚或忽视（请参阅松耦合与紧耦合方式）。

所有从辅助服务器启动的全局事务分支都将使用现有代理接口重定向到主服务器。 主服务器启动和维护所有事务分支，并执行与分支关联的所有请求工作。

在可更新辅助服务器上启动了 XA 事务时，主服务器上也将启动相应的 XA 事务。主服务器上的 XA 事务执行 XA 事务的整个生命周期（启动、结束、准备以及落实或回滚）。辅助服务器上的 XA 事务用于支持未重定向到主服务器的查询。发出对 xa_end() 函数的调用时，系统将释放 XA 事务，并将用户会话从 XA 事务分离。所有 XA 事务请求和 XA 事务内发出的所有写操作都将重定向到主服务器。

以下功能特定于GBase 8s XA 实施：

- 所有 XA 接口请求在可更新的辅助服务器上均可用（请参阅辅助服务器上的数据库更新）。
- 支持从可更新的辅助服务器启动、准备以及落实或回滚 XA 事务。
- 支持 xa_recover() 函数，该函数用于从资源管理器获取已就绪事务分支的列表。
- 支持在高可用性集群服务器中执行 XA 事务分支迁移。集群中的任何服务器均可附加到 XA 事务分支，而不考虑事务分支是否源自该服务器。
- XA 客户机和事务管理器可使用连接管理器连接到任何高可用性集群服务器（请参阅连接管理）。
- 支持将 XA 请求从辅助服务器重定向到主服务器。
- 支持对 XA 事务执行事务挽救（请参阅在集群故障转移期间完成事务）。
- 如果运行已重定向的 XA 事务的辅助服务器出现故障，将回滚事务。
- 支持在 XA 环境内运行，但位于 XA 事务外的 SQL 事务。

在辅助服务器上运行的 XA 事务有以下限制：

- 不支持从其他用户会话（在相同或不同辅助服务器上）恢复暂挂的全局事务分支。
- 用户会话不能附加到与另一个辅助服务器上的其他用户会话关联的全局事务分支。
- XA 事务与辅助服务器上的其他数据具有相同的限制。请参阅辅助服务器上的数据库更新。
- XA 事务不能在只读辅助服务器上启动。 如果应用程序尝试在只读辅助服务器上创建新的 XA 事务，将收到 XA 错误代码 XAER_RMERR。此外，在只读辅助服务器上运行 xa_prepare()、xa_commit() 或 xa_rollback() 将返回错误代码 XA_NOTA (-4)。
- 只读辅助服务器上支持以下 XA API：
    - xa_open()
    - xa_close()

重要： 如果要将 .NET Framework 用于 Microsoft™ Transaction Server 以管理高可用性集群中的 XA 事务，那么必须使用 TransactionScope 类而非 ServiceConfig 类。TransactionScope 类可用于 .NET Framework 3.5。

松耦合与紧耦合方式

数据库服务器支持松耦合及紧耦合方式的 XA 全局事务：

- 松耦合方式表示不同的数据库服务器可协调事务，但不可共享资源。 来自事务的所有分支的记录分别作为独立事务显示在逻辑日志中。
- 紧耦合方式表示不同的数据库服务器可协调事务并且共享诸如锁定和日志记录这些资源。 来自事务的所有分支的记录作为同一事务显示在逻辑日志中。

BEA Systems 提供的 Tuxedo 事务管理器支持松耦合方式。

7.1.2 两阶段落实协议

两阶段落实协议在执行事务期间发生系统或介质故障的情况下提供自动恢复机制。两阶段落实协议确保所有参与的数据库服务器接收并执行同一操作（落实或回滚事务），而不论是否有本地或网络故障。

如果有任何数据库服务器无法落实它这一部分的事务，那么一定会阻止参与该事务的所有数据库服务器落实各自的工作。

何时使用两阶段落实协议

数据库服务器对任何在多个数据库服务器上修改数据的事务自动使用两阶段落实协议。

例如，假设连接了三台名为 **australia**、**italy** 和 **france** 的数据库服务器，如下图所示。

图： 已连接的数据库服务器

如果运行以下示例中显示的命令，那么结果是在三台不同的数据库服务器上执行一次更新和两次插入。

```
CONNECT TO stores_demo@italy
BEGIN WORK
        UPDATE stores_demo:manufact SET manu_code = 'SHM' WHERE
manu_name = 'Shimara'
```

```
        INSERT INTO stores_demo@france:manufact VALUES ('SHM', 'Shimara', '30')
        INSERT INTO stores_demo@australia:manufact VALUES ('SHM', 'Shimara',
'30')
COMMIT WORK
```

两阶段落实概念

每个全局事务均有一个协调者和一个或多个参与者，定义如下：

- 协调者可指导全局事务的解决方案。它决定全局事务是必须落实还是必须停止。

  两阶段落实协议始终将协调者的角色分配至当前数据库服务器。在单个事务期间，协调者的角色无法更改。 在何时使用两阶段落实协议 中的样本事务中，协调者是 italy。如果您将该示例中的第一行更改为以下语句，那么两阶段落实协议将协调者的角色分配至 france：

  ```
  CONNECT TO stores_demo@france
  ```

  使用 gstat -x 选项显示分布式事务的协调者。有关更多信息，请参阅监视全局事务。
- 每个参与者指示一个事务分支的执行，事务分支是涉及单个本地数据库的那部分全局事务。全局事务在以下情况中包含几个事务分支：
  - 应用程序使用多个进程为全局事务工作
  - 多个远程应用程序为同一全局事务工作

  在何时使用两阶段落实协议 中，参与者是 france 和 australia。协调者数据库服务器 italy 也起着参与者的作用，因为它也在进行更新。

两阶段落实协议依赖两种通信，消息和逻辑日志记录：

- 消息在协调者和每个参与者之间传递。 来自协调者的消息包括事务标识号和指示信息（如 prepare to commit、commit 或 roll back）。来自每个参与者的消息包括事务状态和所采取操作的报告（如 can commit 或 cannot commit、committed 或 rolled back）。
- 事务的逻辑日志记录保留在磁盘或磁带上以确保即使在参与的数据库服务器（参与者或协调者）上发生故障时的数据完整性和一致性。

有关更多的详细信息，请参阅两阶段落实和逻辑日志记录。

两阶段落实协议的阶段

在两阶段落实事务中，协调者将所有数据修改指示信息（例如，插入）发送至所有参与者。然后，协调者启动两阶段落实协议。两阶段落实协议分两部分，预落实阶段和后决策阶段。

预落实阶段

在预落实阶段期间，协调者和参与者执行以下对话：

协调者

协调者指导每个参与者数据库服务器准备落实事务。

参与者

每个参与者通知协调者它是否可落实其事务分支。

协调者

协调者根据每个参与者的响应来决定落实还是回滚事务。它仅当所有参与者指示它们可以落实各自的事务分支时才决定落实。如果有任何参与者指示它尚未准备好落实其事务分支（或如果它未响应），那么协调者将决定结束全局事务。

后决策阶段

在后决策阶段期间，协调者和参与者执行以下对话：

协调者

协调者将落实记录或回滚记录写入协调者的逻辑日志，然后指示每个参与者数据库服务器落实或回滚事务。

参与者

如果协调者发出落实消息，那么参与者通过将落实记录写入逻辑日志并将消息发送至协调者（确认事务已落实）来落实事务。 如果协调者发出回滚消息，那么参与者回滚事务，但不向协调者发送确认。

协调者

如果协调者发出消息以落实事务，它将在结束全局事务前一直等待以接收来自各参与者的确认。如果协调者发出消息以回滚事务，它将不等待参与者的确认。

两阶段落实协议如何处理故障

两阶段落实协议设计为用可保留所有参与的数据库服务器上的数据完整性的方式来处理系统和介质故障。如果发生故障，两阶段落实协议执行自动恢复。

自动恢复处理的故障类型

以下事件可能导致协调线程或参与者线程终止或挂起，因此需要自动恢复：

- 协调者的系统故障
- 参与者的系统故障
- 网络故障
- 管理员终止协调线程
- 管理员终止参与者线程

管理员在自动恢复中的角色

自动恢复中管理员的唯一角色是在系统或网络故障后将协调者和/或参与者恢复联机。

重要： 慢速网络无法触发自动恢复。除非协调者系统发生故障、网络发生故障或管理员终止协调线程，否则此处描述的恢复机制均不会生效。

协调者故障的自动恢复机制

如果协调线程发生故障，各参与者数据库服务器必须决定在其落实或回滚事务之前还是在其回滚事务之后启动自动恢复。此职责是假定结束的优化的一部分。（请参阅假定结束的优化。）

参与者故障的自动恢复机制

无论何时参与者线程预落实了在两阶段落实协议可完成之前就终止的一项工作，就会发生参与者恢复。参与者恢复的目标是根据协调者作出的决定来完成两阶段落实协议。

根据协调者是决定落实还是回滚全局事务，参与者恢复可由协调者或者参与者驱动。

重要： 要在跨服务器事务打开的同时支持下级服务器关闭或重新启动之后的自动恢复，sqlhosts 文件必须为可能启动分布式操作的每个数据库服务器包含一个条目。在自动恢复期间，协调者的名称从逻辑日志恢复，且下级服务器与协调者重新连接以完成该事务。由于协调者总是使用自己的 onconfig 文件的 DBSERVERNAME 配置参数中的名称来向各参与者标识它自己，因此协调者的 DBSERVERNAME 设置必须是参与者都已知的因特网协议连接名称，但是也可使用正确的连接协议，为协调者和下级服务器之间的连接至少定义一个 DBSERVERALIASES 设置。下级服务器必须能够使用协调者的 DBSERVERNAME 设置或 DBSERVERALIASES 设置连接到该协调者。

假定结束的优化

假定结束的优化是描述两阶段落实协议如何处理事务回滚的术语。

回滚是按以下方式处理的。当协调者确定事务必须回滚时，它发送消息给所有的参与者以回滚它们的工作片段。 协调者不会等待该消息的确认，而是继续进行以关闭事务并将其从共享内存中除去。如果参与者尝试确定该事务的状态，即查明事务已落实还是已回滚（例如：在参与者恢复期间）— 它将在共享内存中找不到任何事务状态。参与者必定将此解释为表示事务已回滚。

7.1.3 独立操作

两阶段落实环境中的独立操作是一种独立于两阶段落实协议而发生的操作。独立操作可能与两阶段落实协议指定的操作相对立，也可能不对立。如果操作是与两阶段落实协议相对立，那么操作会导致错误或启发式决策。启发式决策可能导致不一致的数据库，并且需要手动的两阶段落实恢复。手动恢复是非常复杂的管理过程，必须尽量避免。（有关手动恢复过程的说明，请参阅从失败的两阶段落实手动恢复。）

启动独立操作的情境

两阶段落实协议期间的独立操作很少见，但它可能会发生在以下情况中：

- 参与者的工作片段发展成长事务错误并回滚。

- 管理员在协议的后决策阶段使用 gadmin -z 停止参与者线程。
- 管理员在协议的后决策阶段使用 gadmin -Z 结束参与者事务（工作片段）。
- 在协调者发出落实决策并知道参与者故障之后，管理员使用 gadmin -z 或 gadmin -Z 结束了协调者数据库服务器上的全局事务。该操作始终会导致错误，尤其是错误 -716。

独立操作的可能结果

如前所述，不是所有的独立操作与两阶段落实协议相对立。独立操作可能会产生以下三种可能的结果：

- 成功完成两阶段落实协议
- 错误状况
- 启发式决策

如果操作不是与两阶段落实协议相对立，那么事务将正常落实或回滚。如果操作过早结束全局事务，会导致错误状况。在协调者上结束全局事务不会被视为启发式决策。如果操作与两阶段落实协议相对立，那么会导致启发式决策。 所有这些情况均在随后各节中进行说明。

允许事务成功完成的独立操作

独立操作不一定需要与两阶段落实协议相对立。例如，如果参与者数据库服务器上的工作片段因为发展成长事务而回滚，并且协调者发出回滚全局事务的决策，那么数据库将保持一致。

导致错误条件的独立操作

如果您（作为协调者数据库服务器管理员）在协调者发出其最终的落实决策后运行 gadmin -z（停止协调者线程）或 gadmin -Z（停止全局事务），那么将从协调者数据库服务器上的共享内存中除去所有事务的信息。

该操作不会被视为启发式决策，因为它不干扰两阶段协议；它可能是可接受的，或可能干扰了参与者恢复并导致错误。

在所有参与者可以毫无困难地落实事务的任何时候，该操作均是可接受的。在这种情况下，强制结束事务的操作是多余的。仅当协调者准备终止事务时，才会收到您运行了 gadmin -Z 的指示。

但实际上，仅当您要尝试加快结束保持打开时间异常长的全局事务时，才可能会考虑在协调者数据库服务器上运行 gadmin -z 或 gadmin -Z。在这种情况中，问题的来源可能是某些参与者数据库服务器上发生的故障。协调者尚未接收到参与者落实其工作片段的确认，而协调者正在尝试建立与参与者的通信以进行调查。

如果您在协调者正主动尝试重新建立通信的同时运行 gadmin -z 或 gadmin -Z，那么协调线程会遵循您的指令而终止，但终止之前它会将错误 -716 写入数据库服务器消息日志。该

操作被视为错误，这是因为两阶段落实协议被强制中断，使协调者不能确定数据库是否一致。

在协调者数据库服务器上停止全局事务不会被视为启发式决策，但它可能导致不一致的数据库。例如，如果参与者最终恢复联机但在协调者共享内存中找不到全局事务，它将回滚其工作片段，从而导致数据库不一致。

导致启发式决策的独立操作

当以下两个条件都成立时，有些独立操作会发展成启发式决策：

- 参与者数据库服务器已经将 can commit 消息发送到协调者并随后回滚。
- 协调者的决定是落实事务。

当两个条件都成立时，最终结果就是未一致实现的全局事务（由一个或多个数据库服务器落实但由另一数据库服务器回滚）。数据库变得不一致。

以下是两个可能的启发式决策：

- 启发式回滚（在启发式回滚场景中有描述）
- 启发式结束事务（在启发式结束事务场景中有描述）

在发生启发式回滚或结束事务之后，可能必须执行手动恢复，这是一个复杂耗时的过程。必须完全了解启发式决策，以便避免这些问题。在两阶段落实的上下文中运行 gadmin -z 或 gadmin -Z 务必谨慎。

启发式回滚场景

在启发式回滚中，数据库服务器或管理员回滚已经发送了 can commit 消息的工作片段。

导致启发式回滚的条件

以下两个条件可能导致启发式回滚：

- 逻辑日志填充至 LTXEHWM 配置参数定义的点。（请参阅《GBase 8s 管理员参考》 中有关配置参数的主题。）长事务状况的来源是正代表全局事务执行的工作片段。
- 管理员执行 gadmin -z session_id 以停止数据库服务器线程，该线程正在执行代表全局事务而执行的工作片段。

在任一情况中，如果该工作片段已经向其协调者发送了 can commit 消息，那么该操作被视为启发式决策。

条件 1：逻辑日志填充至高水位标志

在两阶段落实中，将阻拦正在等待协调者指令的参与者数据库服务器完成其事务。因为事务保持打开，所以包含与该事务相关联的记录的逻辑日志文件无法释放。结果是逻辑日志由于并发用户的活动而继续填充。

如果当参与者正在等待时，逻辑日志填充至长事务高水位标志的值 (LTXHWM)，那么数据库服务器将指导所有拥有长事务的数据库服务器线程开始回滚这些长事务。如果预落实的工作片段就是该违规的长事务，那么数据库服务器启动启发式回滚。也就是说，该数据库服务器在没有协调者的指令或协调者不知道的情况下正在回滚预落实的工作片段。

在两阶段落实中，包含与该工作片段相关联的记录的逻辑日志文件视为打开，直到写入 ENDTRANS 逻辑日志记录。该类型的事务与涉及单个数据库服务器的事务（其中回滚实际关闭事务）不同。

逻辑日志可能继续填充直至达到互斥高水位标志 (LTXEHWM)。如果发生这种情况，所有用户线程暂挂，但当前回滚或当前落实的那些线程除外。在两阶段落实应用场合中，打开的事务使您不能备份逻辑日志文件以及释放逻辑日志中的空间。在这些特殊情况下，逻辑日志可以完全填充。如果发生这种情况，那么参与者数据库服务器关闭，且您必须执行数据复原。

条件 2：系统管理员执行 gadmin -z

您（作为管理员）可以决定通过运行 gadmin -z 来启动预落实工作片段的启发式回滚。 您可能因为希望释放该工作片段拥有的资源而作出该决策。（如果通过运行 gadmin -z 停止参与者线程，那么即便您没有结束事务，也会释放参与者线程拥有的所有锁定和共享内存资源。）

启发式回滚的结果

这些主题描述了在发生启发式回滚时协调者和参与者上所发生的事件，以及此过程如何会导致不一致的数据库：

在发生回滚的参与者数据库服务器上，记录放置在数据库服务器逻辑日志（HEURTX 类型）中。事务拥有的锁和资源得以释放。参与者线程将以下消息写入数据库服务器消息日志，指示发生了长事务状况和回滚：

```
Transaction Completed Abnormally (rollback):
tx=address flags=0xnn
```

协调者发出后决策阶段指令以落实事务。

发生启发式回滚的数据库服务器上的参与者线程将错误消息 -699 返回至协调者，如下所示：

```
-699 Transaction heuristically rolled back.
```

此时该错误消息不返回至应用程序；它对协调者来说是条内部通知。协调者等待至所有参与者响应该落实指令。 直至所有参与者报告后协调者才能确定数据库一致性。

接下去的步骤取决于发生在其他参与者上的操作。可能有两种情境。

情境 1：协调者发出落实并且所有参与者都报告启发式回滚

协调者收集所有来自参与者的响应。如果每个参与者报告启发式回滚，那么后果是会发生以下事件：

1. 协调者将以下消息写入其自己的数据库服务器消息日志：

   ```
   Transaction heuristically rolled back.
   ```

2. 协调者向所有的参与者发送消息以结束事务。
3. 每个参与者都将 ENDTRANS 记录写入其逻辑日志缓冲区中。 （从事务表中除去该事务条目。）
4. 协调者将错误 -699 返回至应用程序，如下所示：

   ```
   -699 Transaction heuristically rolled back.
   ```

5. 在这种情况下，所有数据库保持一致。

情境 2：协调者已发出落实；有一个参与者落实并有一个参与者报告启发式回滚

协调者收集所有来自参与者的响应。如果至少一个参与者报告启发式回滚并且至少一个参与者报告确认落实，那么该结果称为混合事务结果。后果是发生以下事件：

1. 协调者将以下消息写入其本身的数据库服务器消息日志：

   ```
   Mixed transaction result. (pid=nn user=userid)
   ```

pid 值是协调者过程的用户过程标识号。user 值是与协调者进程相关联的用户标识。与该消息相关联的是附加消息，这些附加消息列出每个报告了启发式回滚的参与者数据库服务器。附加消息采用以下形式：

```
Participant database server dbservername heuristically rolled back.
```

2. 协调者向每个启发式回滚其工作片段的参与者发送消息，指导各参与者结束事务。
3. 每个参与者都将 ENDTRANS 消息写入其逻辑日志缓冲区中。 （从事务表中除去该事务条目。）
4. 协调者将 ENDTRANS 消息写入其逻辑日志缓冲区中。 （从共享内存事务表中除去该事务条目。）
5. 协调者将错误 -698 返回至应用程序，如下所示：

   ```
   -698 Inconsistent transaction. Number and names of servers rolled back.
   ```

6. 与该错误消息相关联的是报告了启发式回滚的参与者数据库服务器的列表。如果大量数据库服务器回滚该事务，此列表可能会被截断。完整的列表始终包含在协调者数据库服务器的消息日志中。

在这种情况中，检查每个参与者数据库服务器站点上的逻辑日志并确定您的数据库系统是否一致。（请参阅确定事务是否不一致地实现。）

启发式结束事务场景

启发式结束事务是管理员采取的独立操作，以便回滚工作片段并从事务表中除去所有有关事务的信息。当管理员执行 gadmin -Z address 命令时会启动启发式结束事务进程。

每当通过运行 gadmin -Z 来启动启发式结束事务，都会除去数据库服务器支持两阶段落实协议及其自动恢复功能所需的关键信息。如果运行 gadmin -Z，那么您应负责确定您的联网数据库系统是否一致。

何时执行启发式结束事务

仅在一种很少见的情况下，才必须运行 gadmin -Z 选项来启动启发式结束事务。 当已启发式回滚的工作片段保持打开时会发生这种情况，它会使您的逻辑日志文件不能成为可用的文件。结果，逻辑日志将十分接近填满，这很危险。

通常，协调者会在合理时间段内发出其落实或回滚决策。 但是，如果协调者发生故障，不返回联机状态以结束在参与者数据库服务器上启发式回滚的事务，您可能会面临严重的问题。

问题应用场合以如下方式开始：

1. 代表全局事务执行工作片段的参与者线程已向协调者发送 can commit 响应。
2. 工作片段等待来自协调者的指令。
3. 当工作片段正在等待时，逻辑日志填充超过长事务高水位标志。
4. 正在等待指令的工作片段是长事务的来源。参与者数据库服务器指导执行线程回滚该工作片段。该操作就是启发式回滚。
5. 参与者继续等待协调者指导其结束事务。事务保持打开。逻辑日志继续填充。

如果协调者联系参与者并指导其在合理的时间段内结束事务，那么不会产生任何问题。 如果在参与者数据库服务器上发生启发式回滚并且随后协调者发生故障，致使协调者不能指导参与者结束事务，那么会发生严重问题。

结果，事务保持打开。打开的事务使您不能备份逻辑日志文件以及释放逻辑日志中的空间。当逻辑日志继续填充时，它可能会达到互斥存取长事务高水位标志 (LTXEHWM) 指定的点。如果到达该点，正常的处理将暂挂。在到达高水位标志后的某个时刻，您必须判定打开的事务是否在危及您的逻辑日志。危险就是如果逻辑日志完全填满，那么数据库服务器关闭，并且您必须执行数据复原。

您必须决定是结束事务并保护您的系统不会有填满逻辑日志的可能性（而不考虑所有与运行 gadmin -Z 相关联的问题），还是等着查看是否能及时重新建立与协调者的通信，以便在逻辑日志填满前结束事务。

如何使用 gadmin -Z

只有在协调者和参与者之间的通信中断时，才会考虑使用 gadmin -Z address 命令。为了确保该通信真正中断，在正执行工作片段的线程由于超过 TXTIMEOUT 指定的时间量而终止之前，gadmin -Z 命令不会运行。 有关此选项的更多信息，请参阅《GBase 8s 管理员参考》。

address 参数可从 gstat -x 输出中获取。有关 gstat -x 选项的更多信息，请参阅《GBase 8s 管理员参考》。

启发式结束事务时执行的操作

运行 gadmin -Z 时，您指示 gadmin 实用程序从事务表中除去了位于指定地址的参与者事务条目。

逻辑日志中写入了两条记录以记载该操作。 记录类型为 ROLLBACK 和 ENDTRANS，或如果事务已经启发式回滚，那么仅有类型 ENDTRANS。以下消息写入参与者数据库服务器消息日志：

```
(time_stamp) Transaction Completed Abnormally (endtx): tx=address flags:0xnn
user username tty ttyid
```

协调者接收来自发生了 gadmin -Z 的参与者的错误消息以作为其对 COMMIT 指令的响应。协调者查询参与者数据库服务器，该服务器不再有有关事务的消息。参与者数据库服务器上缺少事务表条目就表明事务已落实。 协调者假设参与者已发送确认消息，但由于某种原因未接收到该消息。因为协调者不知道此参与者的工作片段未落实，所以未生成指示全局事务不一致实现的消息。只有运行 gadmin -Z 命令的管理员才会知道实现不一致。

监视全局事务

使用 gstat -x 命令跟踪开放事务并确定它们是否已经启发式回滚。

例如，在输出中，**flags** 字段中的 H 标志标识启发式回滚，G 标志标识全局事务，L 标志指示松耦合方式，而 T 标志指示紧耦合方式。

curlog 和 **logposit** 字段提供逻辑日志记录的准确位置。 如果事务未在回滚，那么 **curlog** 和 **logposit** 描述最近写入的日志记录。当事务在回滚时，这些字段描述最近“撤销”的日志记录。在事务回滚时，**curlog** 和 **logposit** 值会减少。在长事务中，**logposit** 和 **beginlg** 的值汇聚的速率可帮助您估计回滚将需要的更多时间量。

有关 gstat -x 输出的更多信息及其示例，请参阅《GBase 8s 管理员参考》。

还可以使用 gstat -u 和 gstat -k 命令跟踪事务以及它们保存的锁。有关 gstat -x 显示的字段的描述，请参阅《GBase 8s 管理员参考》。

在辅助服务器上，当启用了故障转移后完成事务（通过设置 FAILOVER_TX_TIMEOUT 配置参数）时，两个全局事务可能有相同的全局事务标识：一个是本地临时全局事务，另一个是属于恢复线程的全局事务。快速区分真正的全局事务与临时事务的方法是，如果真正的事务执行了任何操作，将有一个 B 标志。也可通过使用 gstat -g ath 命令来检查事务的所有者。调用了 xa_end() 函数之后，将删除辅助服务器上的临时全局事务。

以下 gstat 实用程序的示例输出说明了高可用性集群环境中主服务器和辅助服务器上对 XA 事务的支持。gstat -x、gstat -G 和 gstat -ath 命令会单独记录，但是组合的 gstat -xG 命令的输出专门针对全局事务。示例显示了重定向事务的每种状态。

在示例中，所显示的在辅助服务器上运行的全局事务是临时事务。临时事务用于支持在辅助服务器上执行的 SQL 语句（而非重定向到主服务器的事务）。仅当用户线程主动与全局事务分支关联时，才会显示临时事务。

以下示例显示运行 xa_start() 函数之后在辅助服务器上运行的 gstat -xG 命令生成的输出：

```
Transactions

est.
address           flags userthread        locks   begin_logpos       current logpos
isol     rb_time   retrys coord
7000000104d4190   AT--G 7000000104a7b68   0       -                  -
LC       -         0
7000000104d8bd0   ALB-G 7000000104a5aa8   1       180:0x0
180:0x4eb018      DIRTY    0:00       0

Global Transaction Identifiers
address           flags   isol     timeout   fID          gtl   bql   data
7000000104d4190   AT--G   COMMIT   0         5067085      15    4
00010203040506070809 0A0B0C0D0E0F000000
7000000104d8bd0   ALB-G   DIRTY    0         5067085      15    4
00010203040506070809 0A0B0C0D0E0F000000
```

在辅助服务器上运行的 gstat -g ath 生成的输出：

```
Threads:
 tid       tcb       rstcb     prty      status                  vp-class     name
 317       7000001500902c8   7000000104a7b68   1      cond wait   netnorm
1cpu            sqlexec
 84        7000001403a7dc0   7000000104a5aa8   3      sleeping secs: 1
5cpu            xchg_2.0
```

在主服务器上运行的 gstat -xG 生成的输出：

```
Transactions

est.
address           flags userthread        locks   begin_logpos       current logpos
isol     rb_time   retrys coord
7000000104d9e60   ATB-M 7000000104a8bc8   2       180:0x4ea018
180:0x4eb018      COMMIT   0:00       0
```

```
Global Transaction Identifiers
address              flags   isol     timeout   fID          gtl   bql   data
7000000104d9e60    AT--M   COMMIT   0         5067085      15    4
000102030405060708090A0B0C0D0E0F000000
```

以上示例中的 M 标志指示全局事务是从辅助服务器启动的。在主服务器上启动的全局事务会显示 G 标志。M 标志仅在主服务器上显示。

gstat -g ath|grep 7000000104a8bc8 命令生成的输出：

```
196       70000013012d3a8   7000000104a8bc8   1      sleeping secs: 1
4cpu              proxyTh
```

以下示例显示运行 xa_end() 函数之后在辅助服务器上运行 gstat -xG 命令生成的输出：

```
Transactions

est.
address              flags userthread          locks   begin_logpos        current logpos
isol      rb_time   retrys coord
7000000104d8bd0   ALB-G 7000000104a5aa8   1         180:0x0
180:0x4ee018         DIRTY     0:00        0

Global Transaction Identifiers
address              flags   isol     timeout   fID          gtl   bql   data
7000000104d8bd0    ALB-G   DIRTY    0         5067085      15    4
000102030405060708090A0B0C0D0E0F000000
```

在主服务器上运行 gstat -xG 命令生成的输出：

```
Transactions

est.
address              flags userthread          locks   begin_logpos        current logpos
isol      rb_time   retrys coord
7000000104d9e60   -TB-M 0                   2         180:0x4ea018
180:0x4ee018         COMMIT    0:00        0


Global Transaction Identifiers
address              flags   isol     timeout   fID          gtl   bql   data
7000000104d9e60    -T--M   COMMIT   -1        5067085      15    4
000102030405060708090A0B0C0D0E0F000000
```

运行 xa_prepare() 函数之后在辅助服务器上运行 gstat -xG 命令生成的输出：

```
Transactions

est.
address             flags userthread          locks   begin_logpos        current logpos
isol      rb_time   retrys coord
7000000104d8bd0   ALX-G 7000000104a5aa8   1         180:0x0
180:0x4ef018        DIRTY     0:00        0

Global Transaction Identifiers
address             flags   isol      timeout   fID           gtl   bql   data
7000000104d8bd0     ALX-G   DIRTY     0         5067085       15    4
000102030405060708090A0B0C0D0E0F000000
```

在主服务器上运行 gstat -xG 命令生成的输出：

```
Transactions

est.
address             flags userthread          locks   begin_logpos        current logpos
isol      rb_time   retrys coord
7000000104d9e60   -TX-M 0                   2         180:0x4ea018
180:0x4ef018        COMMIT    0:00        0

Global Transaction Identifiers
address             flags   isol      timeout   fID           gtl   bql   data
7000000104d9e60     -TX-M   COMMIT    -1        5067085       15    4     0
```

7.1.4 两阶段落实协议错误

以下两阶段落实协议错误需要管理员特别加以注意。

错误号　　描述

-698

如果您接收到错误 -698，那么说明已发生启发式回滚，并已导致未一致实现的事务。启发式回滚的结果中描述了导致该事件的环境。有关不一致事务如何发展的说明并了解您可用的选项，请参阅这些信息。

-699

如果您接收到错误 -699，那么说明已发生启发式回滚。启发式回滚的结果中描述了导致该事件的环境。有关不一致事务如何发展的说明，请参阅这些信息。

-716

如果您接收到错误 -716，那么说明协调线程已在其发出最后决定后被管理员终止。在导致错误条件的独立操作中描述了该应用场合。

7.1.5 两阶段落实和逻辑日志记录

数据库服务器使用逻辑日志记录实现两阶段落实协议。您可以使用这些逻辑日志记录来检测启发式决策并（如有必要）帮助您执行手动恢复。（请参阅从失败的两阶段落实手动恢复。）

分布式事务中包括以下逻辑日志记录：

- BEGPREP
- PREPARE
- TABLOCKS
- HEURTX
- ENDTRANS

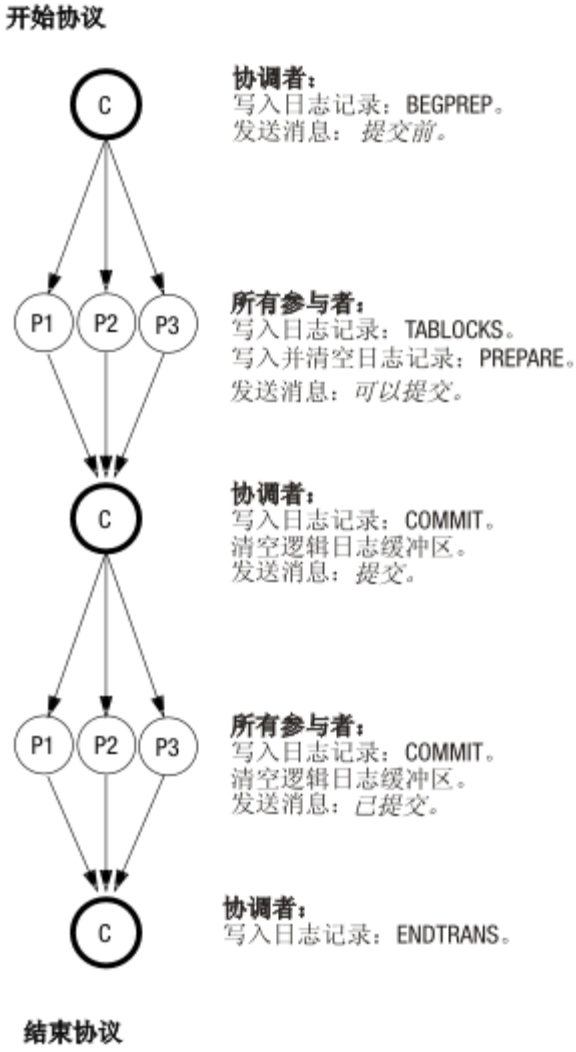
有关这些逻辑日志记录的信息，请参阅《GBase 8s 管理员参考》中有关解释逻辑日志记录的章节。

本节将检查在以下数据库服务器应用场合期间写入的逻辑日志记录的顺序：

事务落实时的逻辑日志记录

下图说明了在生成已落实事务的成功两阶段落实协议期间逻辑日志记录的写入顺序。

图： 已落实事务期间写入的逻辑日志记录



有些逻辑日志记录必须立即从逻辑日志缓冲区清空；对于另外一些记录，清空却不是关键的。

协调者的落实工作记录（COMMIT 记录）包含启动两阶段落实协议所需的所有信息。它还在协调者的主机上发生故障的情况下充当自动恢复的起点。因为该记录对于恢复很关键，所以不允许它保留在逻辑日志缓冲区中。协调者必须立即清空 COMMIT 逻辑日志记录。

上图中的参与者必须立即清空 PREPARE 和 COMMIT 这两条逻辑日志记录。清空 PREPARE 记录可确保快速恢复能在参与者的主机发生故障时，确定此参与者是否为全局事务的一部分。 作为恢复的一部分，参与者可以查询协调者以了解该事务的最终布置。

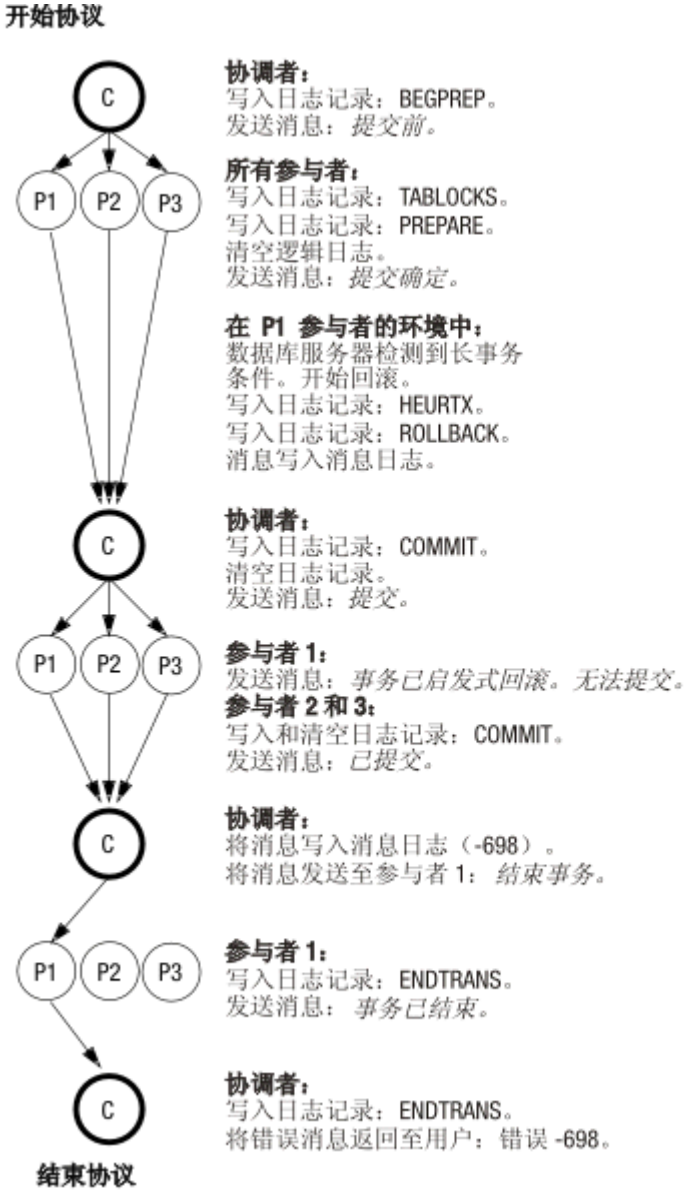
清空参与者的 COMMIT 记录可确保在参与者的主机发生故障时，参与者具有关于事务的所执行操作的记录。要理解该信息为何至关重要，请考虑在写入 PREPARE 记录之后但在清空 COMMIT 记录之前参与者崩溃的情况。在快速恢复后，PREPARE 记录得以复原，但 COMMIT 记录丢失（因为在发生故障时它位于逻辑日志缓冲区中）。PREPARE 记录的存在将启动对协调者的有关事务的查询。但是，协调者会对该事务一无所知，因为协调者在接收到参与者的确认（已执行落实）后就结束了事务。在这种情况中，参与者将把缺

少信息解释为回滚事务的最后指示。两阶段落实协议需要立即清空参与者的 COMMIT 记录，以防止出现此类误解。

启发式回滚期间写入的逻辑日志记录

下图说明了数据库服务器在启发式回滚期间写入逻辑日志记录的顺序。 因为只有在参与者发送了消息说明其可以落实，并且协调者发送了消息去落实之后，才会执行启发式回滚，所以此协议的第一个阶段与图 1 中所示相同。当执行启发式回滚时，会假设回滚是由于参与者 1 (P1) 数据库服务器上发生长事务状况而导致的。最终结果是不一致实现的事务。请参阅启发式回滚场景。

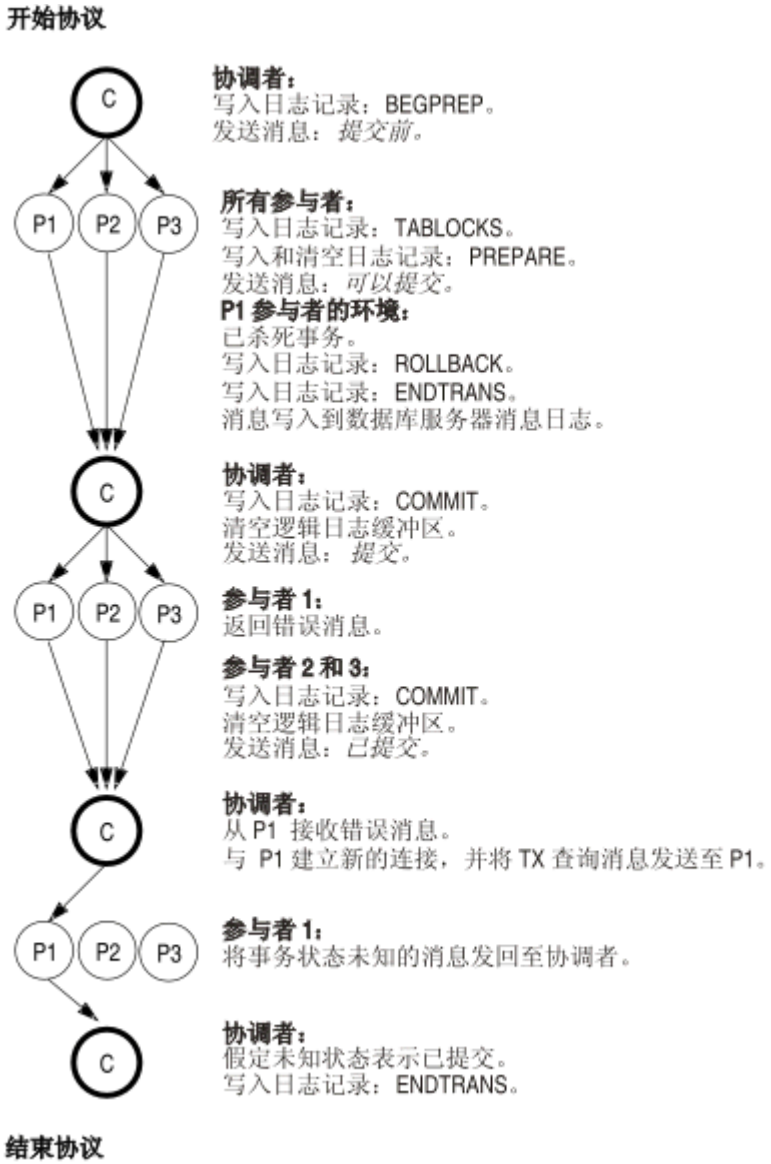
图： 启发式回滚期间写入的逻辑日志记录



启发式结束事务后写入的逻辑日志记录

下图说明了启发式结束事务期间逻辑日志记录的写入顺序。事件始终是在参与者发送了 can commit 消息后，数据库服务器管理员在参与者数据库服务器上结束事务（请参阅《GBase 8s 管理员参考》中有关 gadmin 实用程序的信息）的结果。在下图中，假设已在参与者 1 (P1) 数据库服务器上执行启发式结束事务。结果是未一致实现的事务。 请参阅启发式结束事务场景。

图： 启发式结束事务期间写入的逻辑日志记录



7.1.6 两阶段落实中使用的配置参数

以下两个配置文件参数是特定于分布式环境的：

- DEADLOCK_TIMEOUT
- TXTIMEOUT

虽然两个参数都指定了超时周期，但它们是互相独立的。有关这些配置参数的更多信息，请参阅《GBase 8s 管理员参考》。

DEADLOCK_TIMEOUT 参数的功能

如果强制分布式事务等待共享内存资源的时间超过 DEADLOCK_TIMEOUT 指定的秒数，那么拥有事务的线程会假设存在多服务器死锁。会返回以下错误消息：

```
-154 ISAM error: deadlock timeout expired - Possible deadlock.
```

DEADLOCK_TIMEOUT 的缺省值为 60 秒。调整该值时要小心。如果把该值设置得过低，各个数据库服务器会结束那些非死锁的事务。如果把该值设置得过高，那么多服务器死锁可能会减少并发性。

TXTIMEOUT 参数的功能

TXTIMEOUT 配置参数特定于两阶段落实协议。仅当事务协调者和参与者之间的通信被中断，并且必须重新建立时，才使用该参数。

TXTIMEOUT 参数指定参与者数据库服务器在分布式事务期间等待接收来自协调者数据库服务器的 commit 指令的时间段。如果过了 TXTIMEOUT 指定的时间段，那么参与者数据库服务器会检查事务的状态以确定参与者是否必须启动自动参与者恢复。

TXTIMEOUT 以秒为单位指定。缺省值为 300（5 分钟）。该参数的最佳值将根据您的特定环境和应用程序而变化。修改此参数之前，请阅读两阶段落实协议如何处理故障中的说明。

7.1.7 异类落实协议

术语“异类环境”在用于 GBase 8s 数据库服务器环境中时，是指其中至少有一个数据库服务器不是 GBase 8s 数据库服务器的数据库服务器组。异类落实可确保异类环境中分布式事务的全有或全无的基础。

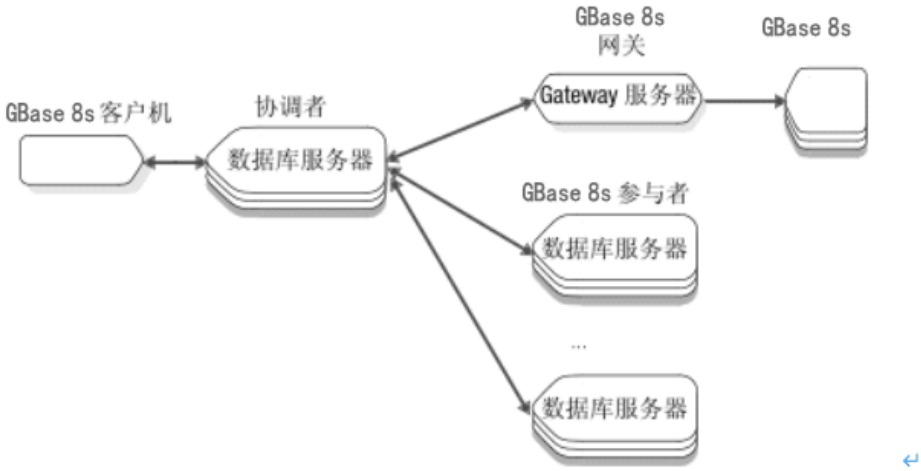
与两阶段落实协议不同，异类落实协议支持非 GBase 8s 参与者的参与。非 GBase 8s 参与者（称为网关参与者）必须通过 GBase 8s 网关与协调者通信。

当满足了以下条件时，数据库服务器将使用异类落实协议：

- 异类落实已启用。（即，HETERO_COMMIT 配置参数设置为 1。）
- 落实的协调者是 V8.5 版本的 GBase 8s。
- 非 GBase 8s 参与者通过 GBase 8s 网关与 GBase 8s 数据库服务器通信。
- 在单个事务中至多有一个非 GBase 8s 参与者执行更新。

下图说明了此场景。

图： 需要对分布式事务执行异类落实的配置



可参与异类落实事务的网关

网关充当 GBase 8s 应用程序（在本例中是数据库服务器）和非 GBase 8s 数据库服务器之间的桥梁。可使用网关将 GBase 8s 应用程序用于访问和修改存储非 GBase 8s 数据库中存储的数据。

下表列出了可参与某事务（数据库服务器在该事务中使用异类落实协议）的网关和相应的数据库服务器。

表 1. 网关和相应的数据库服务器/异类落实事务

网关	数据库服务器
GBase 8s Enterprise Gateway for EDA/SQL	EDA/SQL
GBase 8s Enterprise Gateway Manager	任何具有 ODBC 连接的数据库服务器

启用和禁用异类落实

使用文本编辑器来更改启用或禁用异类落实的 HETERO_COMMIT 配置参数：此更改在关闭并重新启动数据库服务器后生效。

使用文本编辑器或 ISA 来更改启用或禁用异类落实的 HETERO_COMMIT 配置参数：更改的关闭并重新启动数据库服务器后生效。

如果将 HETERO_COMMIT 设置为 1，事务协调者会检查是否有需要使用异类落实的分布式事务。当协调者检测到这类事务，它会自动执行异类落实协议。

如果将 HETERO_COMMIT 设置为 0 或任何非 1 的数字，那么事务协调者将禁用异类落实协议。 下表总结事务协调者为确保分布式事务的完整性而使用哪一协议（异类落实还是两阶段落实）。

HETERO_COMMIT 设置	网关参与者是否已更新	数据库服务器协议
已禁用	否	两阶段落实
已禁用	是	两阶段落实
已启用	否	两阶段落实
已启用	是	异类落实

异类落实的工作原理

异类落实协议是标准两阶段落实协议的修改版本。异类落实协议中的后决策阶段与两阶段落实协议的后决策阶段相同。预落实阶段则包含一处轻微修改，并且向异类落实协议添加了称为网关落实阶段的新阶段。

以下主题解释了对预落实阶段和网关落实阶段的修改。有关后决策阶段的详细说明，请参阅后决策阶段。

预落实阶段

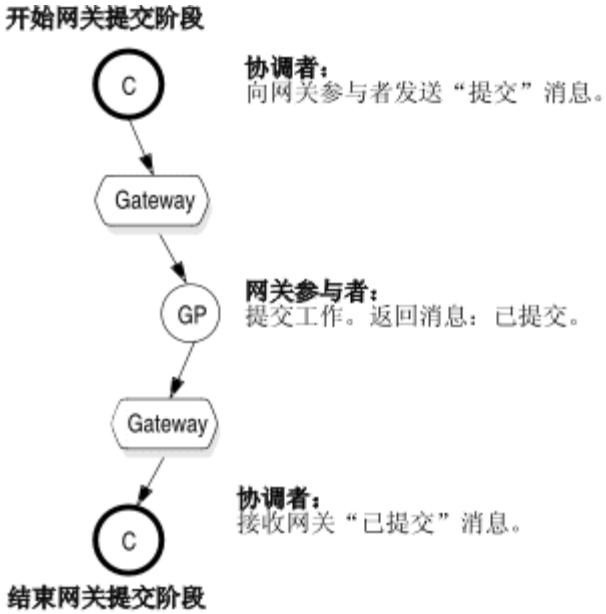
协调者指导每个更新参与者（网关参与者除外）准备落实事务。

如果更新满足了所有延迟的约束，那么所有参与者（网关参与者除外）会将指示其可以提交各自的工作片段的消息返回至协调者。

网关落实阶段

如果所有参与者均成功返回了指示其已准备好落实的消息，那么协调者将落实消息发送至网关。接着，网关会将指示网关是否已落实它这部分事务的响应发送至协调者。如果网关落实了事务，协调者会决定落实整个事务。下图说明了此过程。

图： 生成已落实事务的异类落实阶段



如上图所示，如果网关未能落实事务，那么协调者将回滚整个事务。

异类落实优化

当唯一接收更新的参与者是非 GBase 8s 数据库时，数据库服务器将优化异类落实协议。在这种情况下，协调者将在不调用异类落实协议的情况下向所有参与者发送一条落实消息。

失败异类落实的含意

在分布式事务期间，任何时候数据库服务器使用异类落实进行处理，协调者或任意数目的参与者都可能发生故障。数据库服务器用与两阶段落实协议中所使用方法相同的方法来处理这些故障，但某些情况除外。以下主题详细分析了这些特殊情况。

数据库服务器协调者故障

协调者发生故障后数据的一致性取决于异类落实过程中协调者发生故障的时刻。如果协调者在向网关发送落实消息之前发生故障，那么一旦恢复后就停止整个事务，这与两阶段落实的情况相同。

如果协调者在写入落实日志记录之后发生故障，那么一旦恢复后就成功落实整个事务，这与两阶段落实的情况相同。

如果协调者在向网关发送落实消息之后但在写入落实日志记录之前发生故障，那么一旦恢复，事务中的远程 GBase 8s 数据库服务器站点将停止。如果网关接收到落实消息并落实了事务，那么这种情况可能会导致不一致。

下表总结了这些应用场合。

数据库服务器协调者发生故障的时间点	预期结果
在协调者写入 PREPARE 日志记录之后，但在网关落实阶段之前	维持了数据一致性。

数据库服务器协调者发生故障的时间点	预期结果
在协调者向网关发送落实消息之后，但在它接收到答复之前	数据有可能不一致。协调者没有指示有可能数据不一致。
在网关落实阶段之后，但在协调者向逻辑日志写入 COMMIT 记录之前	失去数据一致性。协调者没有指示数据不一致。

参与者故障

无论何时使用异类协议的分布式事务中的参与者发生故障，协调者都将发送以下错误消息：

```
-441    Possible inconsistent data at the target DBMS name due to an aborted commit.
```

此外，数据库服务器还向消息日志发送以下消息：

```
Data source accessed using gateway name might be in an inconsistent state.
```

参与者故障并不限于数据库服务器或网关的故障。此外，协调者和网关之间的通信链接的故障被认为是网关故障。如果发生链接故障，那么网关将终止。网关必定会终止，因为它没有保留事务日志，因此无法重新建立与协调者的连接并重新开始事务。由于这种限制，所以存在一些应用场合，在这些应用场合中网关故障可能使数据处于不一致状态。 下表总结了这些应用场合。

参与者发生故障的时间点	预期结果
在参与者接收到来自协调者的落实事务消息之后，但在参与者执行落实之前	维持了数据一致性。
在参与者接收到来自协调者的落实事务消息并落实了事务之后，但在参与者回复协调者之前	数据不一致。
在参与者落实事务并向协调者发送回复之后	如果在协调者接收到回复之前通信链接发生故障，那么数据不一致。如果协调者接收到回复，那么数据是一致的（前提是协调者在写入 COMMIT 记录之前没有发生故障）。

当参与者发生故障时数据库服务器所遵循的恢复过程与两阶段落实中所遵循的过程相同。有关此过程的更多信息，请参阅两阶段落实协议如何处理故障。

异类落实错误消息的解释

当数据库服务器未能使用异类落实来处理分布式事务时，它将返回以下主题中说明的两个错误消息之一。

应用程序尝试更新多个网关参与者

当 HETERO_COMMIT 设置为 1 时，如果您的客户机应用程序尝试在多个网关参与者上更新数据，那么协调者将返回以下错误消息：

```
-440 Cannot update more than one non-GBase8s DBMS within a transaction.
```

如果您接收到该错误消息，可重写该违规的应用程序，以使它在单个分布式事务中至多更新一个网关参与者。

尝试使用异类落实来落实分布式事务失败

数据库服务器有可能因为以下一个或多个原因而导致在使用异类协议时未能落实分布式事务：

- 通信错误
- 站点故障
- 网关故障
- 其他未知错误

当发生这类故障时，协调者将返回以下消息：

```
-441 Possible inconsistent data at the target DBMS name due to an aborted commit.
```

在数据库服务器发送该消息后，它会回滚所有正在参与事务的更新站点，但网关参与者站点上完成的工作有可能出现异常。 如果故障发生在网关参与者处理落实消息之后，那么网关参与者可能已落实其更新。如果网关参与者落实了更新，您必须手动回滚这些更新。

7.2 从失败的两阶段落实手动恢复

分布式事务遵循两阶段落实协议。某些操作的发生独立于两阶段落实协议，就导致事务不一致地实现。（请参阅独立操作。）在这些情况中，可能需要从事务进行手动恢复。

7.2.1 确定是否需要手动恢复

以下主题概述了确定数据库一致性并更正相关情况（如果需要）的过程中的步骤。

以下主题中描述了其中每个步骤。

确定事务是否不一致地实现

首要任务是确定事务是否由于独立操作而不一致地实现。

全局事务过早结束

如果运行了 gadmin -z 命令来结束协调者上的全局事务，那么事务可能会不一致地实现。（有关该情况会如何发生的说明，请参阅导致错误条件的独立操作。）您可以通过首先检查数据库服务器消息日志来检查协调者是否有不一致的事务。查找以下错误消息：

```
-716 Possible inconsistent transaction.
Unknown servers are server-name-list.
```

该消息列出所有充当参与者的数据库服务器。 检查每个参与者的逻辑日志。如果至少一个参与者执行了落实且一个参与者执行了回滚，那么事务不一致地实现。

启发式结束事务

如果运行了 gadmin -Z address 命令来结束某个参与者执行的工作片段，并且协调者决定落实事务，那么事务将不一致地实现。（有关该应用场合的描述，请参阅启发式结束事务场景。）检查每个参与者的逻辑日志。如果至少一个参与者执行了落实且一个参与者执行了回滚，那么事务不一致地实现。

启发式回滚

您可以用以下方法来确定启发式决策所影响的特定数据库服务器参与者，从而回滚事务：

检查应用程序中 COMMIT WORK 语句的返回码。

- 以下消息指示有一个参与者执行了启发式回滚：

```
-698 Inconsistent transaction. Number and names of servers rolled back.
```

- 检查数据库服务器消息日志文件的消息。

  如果由于参与数据库服务器上的启发式决策导致有可能出现数据库不一致，那么协调者的数据库服务器消息日志文件中会出现以下消息：

```
Mixed transaction result. (pid=nn user=user_id)
```

  无论何时返回错误 -698，均会写入该消息。与该消息相关联的是事务回滚所涉及的参与者数据库服务器的列表。这是完整的列表。 如果大量参与者回滚了该事务，那么与 -698 错误消息一起创建的列表可能会被截断。

- 检查每个参与者的逻辑日志。

  如果至少一个参与者回滚了其工作片段并且一个参与者落实了其工作片段，那么事务将不正确地实现。

确定分布式数据库是否包含不一致的数据

如果您确定事务是不一致地实现，您必须确定这种情况对您的分布式数据库系统意味着什么。您尤其必须确定数据完整性是否受到影响。

无论何时当一个参与者回滚的工作片段与另一参与者更新的工作片段相关时，不一致实现的事务就造成了问题。不能使用 SQL 定义这些依赖性，因为分布式事务不支持引用多个

数据库服务器上数据的约束。仅当数据已在两个独立事务中更新时，该工作片段才是独立的（不存在相关性）。否则，就认为该工作片段是有相关性的。

在您继续之前，请考虑导致该错误的事务。更新的数据段和回滚的数据段是否互相相关？单个事务可能会由于其他原因（而不是维护数据完整性）而包含多次更新。 例如，以下是三种可能的原因：

- 减少的事务开销
- 简化的编码
- 程序员喜好

并验证每个假设已落实事务的数据库服务器实际修改了数据。只读数据库服务器可能会作为已落实事务的参与者而列出。

如果不一致的事务没有导致数据完整性的违例，此时您就可以退出该过程。

获取逻辑日志记录中的信息

要确定数据完整性是否受到不一致实现的全局事务的影响，必须重建全局事务，并确定事务的哪些部分已落实，哪些已回滚。使用 glogdump 实用程序获取必要信息。过程如下：

1. 在包含 HEURTX 记录的参与者上重建事务。

   a. 参与者数据库服务器逻辑日志是您进行信息搜索的开始点。日志中的每条记录均有本地事务标识号 (xid)。获取 HEURTX 记录的 xid。

   b. 使用本地 xid 定位所有相关联的日志记录，这些记录作为该工作片段的一部分而回滚。

2. 确定哪一数据库服务器是充当全局事务的协调者。

   a. 查找参与者上包含相同本地 xid 的 PREPARE 记录。PREPARE 记录为该参与者标记了两阶段落实协议的起点。

   b. 使用 glogdump -l 选项获取 PREPARE 记录的详细输出。

      该记录包含全局事务标识 (GTRID) 和协调数据库服务器的名称。有关 GTRID 的信息，请参阅获取全局事务标识。

3. 从协调者日志获取其他参与者的列表。

   a. 检查协调者数据库服务器上的日志记录。 找到 BEGPREP 记录。

   b. 检查 BEGPREP 记录的详细输出。

      如果该记录中 GTRID 的前 32 字节与参与者的 GTRID 相匹配，那么 BEGPREP 记录是同一全局事务的一部分。请注意，BEGPREP 详细输出的 ASCII 部分中显示的参与者。

4. 重建每个参与者上的事务。

a. 在每个参与者数据库服务器上，读取逻辑日志以找到包含与该事务关联的 GTRID 的 PREPARE 记录，并获取该参与者执行的工作片段的本地 xid。

b. 在每个参与者数据库服务器上，使用本地 xid 定位所有与该事务（已落实或已回滚）相关联的逻辑日志记录。

在您遵循该过程之后，您将知道事务的所有参与者是哪些、分配给每个参与者哪些工作片段以及每个工作片段是已回滚还是已落实。根据该信息，您可以确定独立操作是否影响了数据完整性。

获取全局事务标识

当全局事务开始时，它会接收到称为全局事务标识 (GTRID) 的唯一标识号。 GTRID 包含协调者的名称。GTRID 已写入协调者的 BEGPREP 逻辑日志记录以及每个参与者的 PREPARE 逻辑日志记录。

要查看 GTRID，请使用 glogdump -l 选项。GTRID 在记录的数据部分中偏移了 20 字节，长为 144 字节。以下示例显示 BEGPREP 记录的 glogdump -l 输出。协调者为 **chrisw**。

```
4a064      188   BEGPREP   4           0   4a038          0      1
           000000bc 00000043 00000004 0004a038 .......C .......8
                    00087ef0 00000002 63687269 73770000 ..~..... chrisw..
                    00000000 00000000 00000000 00087eeb ........ ......~.
           00006b16 00000000 00000000 00000000 ..k..... ........
           00000000 00000000 00000000 00000000
           00000000 00000000 00000000 00000000
           00000000 00000000 00000000 00000000
           00000000 00000000 00000000 00000000
           00000000 00000000 00000000 00000000
           00000000 00000000 00000000 00000000
                    00000000 00000001 6a756469 74685f73 ........ judith_s
           6f630000 736f6374 63700000                  oc..soct cp..
```

协调者上的 BEGPREP 记录与参与者（这些参与者是同一全局事务的一部分）上的 PREPARE 记录的 GTRID 的前 32 字节相同。例如，将以下示例中 PREPARE 记录的 GTRID 与上一个示例中 BEGPREP 记录的 GTRID 相比较。

```
c7064      184   PREPARE   4           0   c7038          chrisw
           000000b8 00000044 00000004 000c7038 .......D ......p8
                    00005cd6 00000002 63687269 73770000 ...... chrisw..
                    00000000 00000000 00000069 00087eeb ........ ...i..~.
           00006b16 00000000 00000010 00ba5a10 ..k..... ......Z.
                    00000002 00ba3a0c 00000006 00000000 ......:. ........
           00ba5a10 00ba5a1c 00000000 00000000 ..Z...Z. ........
```

```
                00ba3a0e 00254554 00ba2090 00000001 ..:..%ET .. .....
00000000 00ab8148 0005fd70 00ab8148 .......H ...p...H
0005fe34 0000003c 00000000 00000000 ...4...< ........
                00000000 00ab80cc 00000000 00ab80c4 ........ ........
00ba002f 63687269 73770000 00120018 .../chrisw......
00120018 00ba0000                   ........
```

确定是否需要执行操作来更正情况

如果不一致的事务创建了不一致的数据库，那么您可以有以下三个选项：

- 让联网数据库保持处于不一致状态。
- 除去落实事务之处受到的事务影响，从而回滚整个事务。
- 在回滚事务之处重新应用事务的影响，从而落实事务。

如果事务未严重影响数据库数据，那么您可以让数据库保持处于不一致状态。如果正在执行事务的应用程序可以继续按原状执行，并且您得出的决定是通过除去影响或重新应用事务来将数据库返回至一致状态的成本（时间与工作量方面的成本）过高，您就可能会遇到这种情况。

无需立即作出此决定。您可以使用以下段落中描述的方法来确定事务在更新哪些数据以及哪些记录受到影响。

当您作决定时，请考虑到没有自动过程或实用程序可以执行已落实事务的回滚或可以落实已回滚事务的一部分。以下段落描述如何浏览数据库服务器消息日志和逻辑日志以找到受影响的记录。如果不具备详细的应用程序知识，那么仅凭消息不足以确定所发生的事件。根据您的应用程序和系统的知识，您必须确定是回滚还是落实事务。您还必须对执行回滚或落实的补偿事务进行编程。

7.2.2 手动恢复的示例

本实例说明了手动恢复所涉及的工作。 以下 SQL 语句由用户 **nhowe** 执行。 返回了错误 -698。

```
dbaccess
CREATE DATABASE tmp WITH LOG;
CREATE TABLE t (a int);
CLOSE DATABASE;
CREATE DATABASE tmp@apex WITH LOG;
CREATE TABLE t (a int);
CLOSE DATABASE;
DATABASE tmp;
BEGIN WORK;
```

```
INSERT INTO t VALUES (2);
INSERT INTO tmp@apex:t VALUES (2);
COMMIT WORK;
### return code -698
```

以下摘要引用自当前数据库服务器上的逻辑日志：

```
addr     len   type      xid       id    link
.....
17018    16        CKPOINT   0         0       13018     0

18018    20        BEGIN     2         1       0         08/27/91 10:56:57
   3482      nhowe



1802c    32        HINSERT   2         0       18018     1000018   102
   4
1804c    40        CKPOINT   0         0       17018     1

   begin     xid       id addr      user

   1         2           1  1802c     nhowe

19018    72        BEGPREP   2         0       1802c     6d69      1

19060    16        COMMIT    2         0       19018     08/27/91 11:01:38

1a018    16        ENDTRANS  2         0       19060     580543
```

以下摘要引用自数据库服务器 apex 上的逻辑日志：

```
addr     len   type      xid       id    link
.....
16018    20        BEGIN     2             1   0         08/27/91
   10:57:07 3483     pault



1602c    32        HINSERT   2         0       16018     1000018   102
   4
```

```
1604c     68          PREPARE   2         0       1602c       eh

17018     16          HEURTX    2         0       1604c           1

17028     12          CLR       2         0       1602c

17034     16          ROLLBACK  2         0       17018     08/27/91 11:01:22

17044     40          CKPOINT   0         0       15018     1

        begin     xid       id addr       user
        1         2         1  17034      --------

18018     16          ENDTRANS  2         0       17034     8806c3
   ....
```

首先您要尝试将当前数据库服务器日志中的事务与 apex 数据库服务器日志中的事务相匹配。BEGPREP 和 PREPARE 日志记录均包含 GTRID。您可以通过使用 glogdump -l 以及查看 BEGPREP 和 PREPARE 日志记录的数据部分来抽取 GTRID。GTRID 在数据部分中偏移了 22 字节，长为 68 字节。更为简单但是准确度较低的方法是查看 COMMIT 或 ROLLBACK 记录的时间。 虽然由于将落实（或回滚）消息从协调者传输至参与者所用时间而导致稍有延迟，但这两个时间必须很接近。（第二个方法缺乏准确度，因为虽然来自同一协调者的并发事务很可能不在同一时间落实，但并发事务是可以在同一时间落实的。）

更正该样本情况

1. 查找所有已更新的记录。
2. 使用 glogdump 和记录类型表来识别记录类型（插入、删除、更新）。
3. 使用每个记录的 glogdump -l 输出以获取本地 xid、表空间数和行标识。
4. 通过将表空间数与 **systables** 系统目录表的 **partnum** 列中的值相比较，从而将表空间数映射到表名。
5. 运用您的应用程序知识来确定需要哪一操作来更正该情况。

在本示例中，不同日志中 COMMIT 和 ROLLBACK 记录上的时间戳记很接近。没有其他活动事务造成另一并发的落实或回滚的可能性。在这种情况下，当前数据库服务器上落实了已分配行标识 102（十六进制）或 258（十进制）的插入 (HINSERT)。 因此，补偿事务如下：

```
DELETE FROM t WHERE rowid = 258
```

8 自动监视和更正操作概述

可以使用 SQL 管理 API、调度程序和向下钻取查询来管理自动维护、监视和管理任务。

GBase 8s 的这些组件简化了复杂系统中的信息收集和服务器维护。

SQL 管理 API

SQL 管理 API 用于通过 SQL 函数执行远程管理。因为 SQL 管理 API 操作完全在 SQL 中执行，所以可在客户机工具中使用这些函数来管理数据库服务器。

调度程序

调度程序是一组任务，用于在预定义时间或按照服务器内部确定的时间来执行 SQL 语句。SQL 语句可以收集信息或监视和调整服务器。

向下钻取查询

向下钻取查询提供了有关最近执行的 SQL 语句的统计信息，以便跟踪各个 SQL 语句的性能并分析语句历史记录。

可以在服务器 HDR 对的主服务器上使用 SQL 管理 API 和调度程序。

这些工具中的每一个工具都需要额外的磁盘空间以用于存储信息。

还可以使用基于 PHP 的 Web 浏览器管理工具 OpenAdmin Tool (OAT) (OAT) 从单一位置管理多个数据库服务器实例。可以使用 OAT 执行的一些任务包括：

- 通过 SQL 管理 API 和调度程序定义和管理自动执行任务
- 为 SQL 语句的分析和调整创建并显示性能柱状图

8.1 调度程序

可以使用调度程序创建作业，以用于在可预测时间运行管理任务或收集信息。调度程序使用 SQL 语句，而不是使用操作系统的作业调度工具。

调度程序由 **sysadmin** 数据库中的一组表进行控制。

调度程序有四种不同的作业类型可供选择：

任务

在指定时间按照频率运行操作。

传感器

在特定时间按特定频率运行操作，以收集数据、创建结果表、将数据存储在结果表中，以及在指定时间之后清除旧数据。

启动任务

仅当服务器从静默方式切换为联机方式时才运行的任务。

启动传感器

仅当服务器从静默方式切换为联机方式时才运行的传感器。

任务或传感器的操作可以是一个或多个 SQL 语句、用户定义的例程或存储过程。

除了为任务或传感器定义操作之外，还可以使用调度程序执行以下操作：

- 将任务和传感器关联到功能组
- 每次运行任务或传感器时跟踪执行时间和返回值
- 使用各种严重性定义警报
- 定义阈值以控制何时运行任务或传感器

调度程序中包含可自动运行的内置任务和传感器。 可以修改内置任务和传感器，并定义自己的任务和传感器。

磁盘空间需求

调度程序表和传感器结果表可能使用大量磁盘空间。

您可以使用以下公式来估计一个传感器的磁盘使用情况：

所收集的行数 * 所收集的行大小 * 每日的数据收集频率 * 保留期

对所有传感器重复该估计，然后您可以确定所需空间的接近估计值。

可以通过降低数据收集的频率来减小存储的数据量，或者通过更新 **ph_task** 表来缩短保留期。

可以使用 SQL 管理 API 将 **sysadmin** 数据库移至其他数据库空间，但是该数据库中的所有现有数据都将丢失。

有关 sysadmin 数据库的更多信息，请参阅《GBase 8s 管理员参考》。

8.1.1 调度程序表

调度程序表位于 **sysadmin** 数据库中，包含有关任务和传感器的信息。

sysadmin 数据库包含下表中列出的调度程序表。ph_task 表与其他各表具有直接关系。

表 1. 调度程序表

表	描述
ph_alert	包含与必须监视的任务关联的错误、警告或参考消息的列表。ph_alert 表包含数据库服务器自动使用的内置警报。 可以添加您自己的警报。

ph_group	包含组名称的列表。每个任务和传感器都是组的成员。ph_group 表包含数据库服务器使用的内置组。 可以添加您自己的组。
ph_run	包含有关每个任务和传感器的运行方式和运行时间的信息。
ph_task	列出任务和传感器，并包含有关数据库服务器运行这些任务和传感器的方式和时间的信息。ph_task 表包含数据库服务器自动使用的内置任务和传感器。 可以添加您自己的任务和传感器。
ph_threshold	包含与任务或传感器关联的阈值的列表。如果达到某个阈值，关联的任务可执行某个操作，例如在 ph_alert 表中插入警报。 ph_threshold 表包含数据库服务器使用的内置阈值。 可以添加您自己的阈值。
results	包含传感器收集的历史数据的多个表。这些表的结构由 ph_task 表的传感器定义中的 CREATE TABLE 语句确定。

有关这些表的详细信息，请参阅《GBase 8s 管理员参考》。

8.1.2 内置任务和传感器

调度程序中包含可自动运行的内置任务和传感器。

下表显示内置调度程序任务和传感器。传感器具有结果表，其中存储了所收集信息，以及用于确定信息存储时间长度的保留期。可通过更新 **ph_task** 表来更改任务和传感器属性，如频率。某些任务由阈值触发。可通过更新 **ph_threshold** 表更改阈值。 可通过将 **ph_task** 表中 **tk_enable** 列的值更改为 f 来禁用任务或传感器。

通过查询 **ph_task** 表中的 **run_duration** 列，可以确定任务所用的时间。

表 1. 内置任务和传感器

任务或传感器	描述	结果表	频率	保留时间
add_storage	此任务在配置自动空间管理时自动添加更多存储空间。		根据需要	
清除警报	该任务从 ph_alert 表中除去存在时间超过了阈值（15 天）的所有警报条目。该阈值在 ph_threshold 表中的名称为 ALERT HISTORY RETENTION。		每天一次	

任务或传感器	描述	结果表	频率	保留时间
auto_compress	此任务压缩配置为自动压缩的表。			
auto_crsd	该任务对表和分段执行压缩、收缩、重新打包和取消分段操作。 缺省情况下，已禁用该任务。必须通过更新 ph_task 表启用该任务。 每个操作在 ph_threshold 表中有两行：一行控制是否启用了该操作，一行控制该操作的阈值。 有关更多信息，请参阅自动优化数据存储。		每周一次	
autoreg exe	该任务注册首次使用的数据库扩展。		根据需要	
autoreg migrate-console	内部。此任务使用日志或缓冲日志的日志记录选项来检查每个数据库，并根据需要将所有内置数据库扩展迁移到对于数据库服务器而言正确的版本。此任务根据需要为各个数据库创建子任务。		服务器启动时	
autoreg vp	该任务在必要时为数据库扩展创建专用虚拟处理器。		根据需要	
auto_tune_cpu_vps	如果分配的 VP 数小于计算机上的 CPU 处理器数的一半，该任务将自动添加 CPU 虚拟处理器。		服务器启动时	

任务或传感器	描述	结果表	频率	保留时间
Auto Update Statistics 评估	该任务根据当前 Auto Update Statistics (AUS) 策略分析记录的所有数据库中的所有表，标识必须更新其分发的表，并为这些表生成 UPDATE STATISTICS 语句。AUS 策略由 **ph_threshold** 表中的阈值设置： • AUS_AGE：统计信息在 30 天之后更新。 • AUS_CHANGE：统计信息在更改数据量达到 10% 时更新。 • AUS_AUTO_RULES：按照准则更新统计信息。 • AUS_SMALL_TABLES：包含的行数少于 100 的表始终自行更新其统计信息。		每天一次	
Auto Update Statistics 刷新	该任务运行“Auto Update Statistics 评估”任务生成的 UPDATE STATISTICS 语句。更新统计信息的 PDQ 优先级由 **ph_threshold** 表中的阈值 AUS_PDQ 设置为 10。		每周六和周日的凌晨 1 点到 5 点之间	
bad_index_alert	该任务检查已损坏的索引。如果找到了任何已损坏索引，将向 **ph_alert** 表添加警告警		每天一次	

任务或传感器	描述	结果表	频率	保留时间
	报。 有关更多信息，请参阅验证索引。			
bar_act_log_rotate	该任务对 BAR_ACT_LOG 配置参数中指定的 ON-Bar 活动日志文件进行循环交替。 ON-Bar 活动日志循环交替时，服务器将切换到新联机消息日志文件，并将之前的日志文件的标识号加 1。达到日志文件最大数量之后，会删除具有最高标识的日志文件。 要循环交替的最大日志的阈值在 ph_threshold 表中指定。		每 30 天的凌晨 3 点（最大日志文件数为 12）	
bar_debug_log_rotate	该任务对 BAR_DEBUG_LOG 配置参数中指定的 ON-Bar 调试日志文件进行循环交替。 ON-Bar 调试日志循环交替时，服务器将切换到新联机消息日志文件，并将之前的日志文件的标识号加 1。达到日志文件最大数量之后，会删除具有最高标识的日志文件。 要循环交替的最大日志的阈值		每 30 天的凌晨 3 点（最大日志文件数为 12）	

任务或传感器	描述	结果表	频率	保留时间
	在 ph_threshold 表中指定。			
check_backup	该任务检查以确保 ph_threshold 表中的阈值指定的时间段以来已运行了备份。 • REQUIRED LEVEL BACKUP：任何级别备份之间最大为 2 天 • REQUIRED LEVEL 0 BACKUP: 0 级备份之间最大为 2 天 如果未进行备份，将向 ph_alert 表添加警告警报。		每天一次	
check_for_ipa	对于每一个具有一个或多个未完成定点变更操作的表，该任务为其在 ph_alert 表中添加一个条目。		每周一次	
idle_user_timeout	该任务终止空闲时间超过了 60 分钟的用户会话。 缺省情况下,已禁用该任务。必须通过更新 ph_task 表启用该任务。 有关更多信息,请参阅自动终止		每 2 小时	

任务或传感器	描述	结果表	频率	保留时间
	空闲连接。			
ifx_ha_monitor_log_replay_task	该任务监视高可用性集群的重放位置。		未设置	
ifx_TrickleFeed_load_ID	此任务持续刷新数据集市中的数据。数据集市和加速器的名称在任务描述中列出。在为数据集市启用缓慢更新后，此任务在调度程序中显示。每个已启用缓慢更新的数据集市都有一个单独的任务。任务名称中的 ID 是唯一的。		启用缓慢更新时指定的每个秒数	
作业运行程序	该任务使用专用 dbWorker 线程在后台运行 OpenAdmin Tool (OAT) 的服务器任务。仅供 OpenAdmin Tool (OAT) 内部使用。		根据需要	
清除作业结果	该任务除去存在时间超过了阈值（30 天）的 OpenAdmin Tool (OAT) 作业结果条目。仅供 OpenAdmin Tool (OAT) 内部使用。		每天一次	
mon_checkpoint	该传感器保存有关检查点的信息。	mon_checkpoint	每小时	7 天
mon_chunk	此传感器保存有关块使用情况和 I/O 块性能的常规信息。	mon_chunk	每小时	30 天
mon_command_history	该任务从 command_history 表中删		每天一次	

任务或传感器	描述	结果表	频率	保留时间
	除存在时间超过了阈值（30 天）的行。该阈值在 ph_threshold 表中的名称为 COMMAND HISTORY RETENTION。			
mon_compression_estimates	此传感器保存有关在压缩数据时可节省的空间量的信息。	mon_compression_estimates	每周一次	30 天
mon_config	该传感器保存 onconfig 文件中每个配置参数的最新值。	mon_config	每天一次	
mon_config_startup	该传感器保存服务器启动时 onconfig 文件中每个配置参数的值。	mon_config	服务器启动时	99 天
mon_iohistory	此传感器保存有关块 I/O 的性能信息。您可以在 ph_threshold 表中更改 IO_SAMPLES_PER_HOUR 参数以更频繁地收集信息。		每小时	30 天
mon_low_storage	该任务扫描数据库空间列表，以查找低于 SP_THRESHOLD 配置参数指定的阈值的空间。然后，该任务将通过使用存储池中的条目来扩展块或添加块，以扩充空间。 有关更多信息，请参阅自动空间管理。	mon_low_storage	每小时	7 天
mon_memory_system	该传感器收集有关服务器所用内存量的信息。	mon_memory_system	每小时	7 天
mon_page_usage	此传感器保存有关存储空间中已	mon_page_usa	每天一次	7 天

任务或传感器	描述	结果表	频率	保留时间
	用和可用的页面的信息。	ge		
mon_profile	该传感器保存服务器概要文件信息。	mon_prof	每 4 小时	30 天
mon_sysenv	该启动传感器保存数据库服务器启动时有关环境的信息。	mon_sysenv	服务器启动时	60 天
mon_table_names	该传感器保存表名称及其创建时间。	mon_table_names	每天一次	30 天
mon_table_profile	该传感器保存表概要文件信息，其中包括该表上发生的更新、插入和删除操作的总数。	mon_table_profile	每天一次	7 天
mon_users	该传感器保存有关每个用户的概要文件信息。	mon_users	每 4 小时	7 天
mon_vps	该传感器收集虚拟处理器信息。	mon_vps	每 4 小时	15 天
online_log_rotate	该任务对 MSGPATH 配置参数中指定的联机消息日志文件进行循环交替。 联机消息日志循环交替时，服务器将切换到新联机消息日志文件，并将之前的日志文件的标识号加 1。达到日志文件最大数量之后，会删除具有最高标识的日志文件。 要循环交替的最大日志的阈值在 ph_threshold 表中指定。		每 30 天的凌晨 3 点（最大日志文件数为 12）	
post_alarm_mess	该任务发布警报。		每小时	

任务或传感器	描述	结果表	频率	保留时间
age				
purge_tables	此任务标识已超过其清除策略的滚动窗口表。它根据每个清除策略放弃或分离符合条件的分段，直至满足该策略或者直至无法再除去任何分段。		每天 00:45	
保存 SQL 跟踪	用户在 OpenAdmin Tool (OAT) 中启用该传感器后，该传感器将当前 SQL 历史记录跟踪缓冲区的内容保存到表中。仅供 OpenAdmin Tool (OAT) 内部使用。	sql_savesnap	每 15 分钟	1 天
SET tk_enable	此任务启用对消息日志文件进行循环交替的任务。		每 30 天的凌晨 3 点	

8.1.3 创建任务

可以创建调度程序任务，以用于在特定时间执行特定操作。

必须以用户 **gbasedbt** 或其他授权用户身份连接 **sysadmin** 数据库。

要创建任务，请使用 INSERT 语句在 ph_task 表中添加一行：

1. 包含以下列的值：

   a. **tk_name**：为任务提供唯一名称。

   b. **tk_type**：将作业类型更改为 TASK 或 STARTUP TASK。

   c. **tk_description**：添加任务所执行的操作的描述。

   d. **tk_execute**：添加任务所执行的操作。

      该操作可以是用户定义的函数、单个 SQL 语句或使用 PREPARE SQL 创建的多语句预编译对象，该多语句预编译对象用于在运行时组合使用一个或多个 SQL 语句。

      命令长度限制为 2048 个字节。

2. **可选**：更改以下列的缺省值：
   - **tk_start_time**：缺省启动时间为 8:00:00。对于启动任务，请将启动时间设置为 NULL。
   - **tk_stop_time**：缺省停止时间为 19:00:00。对于启动任务，请将停止时间设置为 NULL。
   - **tk_frequency**：缺省频率为一天一次。对于启动任务，请将频率设置为 NULL。
   - **tk_group**：缺省组为 MISC。
   - **tk_monday** 到 **tk_sunday**：缺省值为每天运行。

任务会在指定的开始时间以及随后通过频率计算的时间运行。

示例

以下任务使用 SQL 管理 API 在周一、周三和周五早上 8 点到晚上 7 点之间，每隔两分钟执行一个检查点。

```
INSERT INTO ph_task
( tk_name,
tk_description,
tk_type,
tk_group,
tk_execute,
tk_start_time,
tk_stop_time,
tk_frequency,
tk_Monday,
tk_Tuesday,
tk_Wednesday,
tk_Thursday,
tk_Friday,
tk_Saturday,
tk_Sunday)
VALUES
( "Example Checkpoint",
"Example to do a checkpoint every 2 minutes.",
"TASK",
"EXAMPLES",
"EXECUTE FUNCTION admin('checkpoint')",
DATETIME(08:00:00) HOUR TO SECOND,
DATETIME(19:00:00) HOUR TO SECOND,
```

```
INTERVAL ( 2 ) MINUTE TO MINUTE,
't',
  ‘f',
't',
  ‘f',
't',
  ‘f',
  ‘f');
```

以下示例显示了在一天的凌晨 2:00 运行一次的任务代码，以确保 **command_history** 表仅包含最近的数据。在该示例中，最近数据的定义存储在 **ph_threshold** 表的 **Command History Interval** 列中。

```
INSERT INTO ph_task
(
tk_name,
tk_group,
tk_description,
tk_type,
tk_execute,
tk_start_time,
tk_frequency
)
VALUES
(
"mon_command_history",
"TABLES",
"Monitor how much data is kept in the command history table",
"TASK",
"delete from command_history where cmd_exec_time < (
            select current - value::INTERVAL DAY to SECOND
            from ph_threshold
            where name = 'COMMAND HISTORY INTERVAL' ) ",
"2:00:00",
"1 0:00:00"
);
```

8.1.4 创建传感器

可创建调度程序传感器来收集和存储有关数据库服务器的数据。

必须以用户 gbasedbt 或其他授权用户身份连接 sysadmin 数据库。

要创建传感器，请使用 INSERT 语句在 ph_task 表中添加行：

1. 包含以下列的值：
   - **tk_name**：为任务提供唯一名称。
   - **tk_description**：添加任务所执行的操作的描述。
   - **tk_result_table**：添加用于保存传感器所收集数据的表的名称。
   - **tk_create**：添加 CREATE 语句以创建结果表。结果表必须具有名为 ID 的 INTEGER 列，用于容纳传感器标识。可向该表添加其他列。
   - **tk_execute**：添加传感器执行的操作。该操作可以是用户定义的函数、单个 SQL 语句或使用 PREPARE SQL 创建的多语句预编译对象，该多语句预编译对象用于在运行时组合使用一个或多个 SQL 语句。
2. 可以选择更改以下列的缺省值：
   - **tk_type**：缺省值为 SENSOR。对于启动传感器，请将该值更改为 STARTUP SENSOR。
   - **tk_delete**：删除传感器数据之前的缺省时间间隔为一天。
   - **tk_start_time**：缺省启动时间为 8:00:00。对于启动传感器，请将启动时间设置为 NULL。
   - **tk_stop_time**：缺省停止时间为 19:00:00。对于启动传感器，请将停止时间设置为 NULL。
   - **tk_frequency**：缺省频率为一天一次。对于启动传感器，请将频率设置为 NULL。
   - **tk_group**：缺省组为 MISC。
   - **tk_monday** 到 **tk_sunday**：缺省值为每天运行。

传感器会在指定的开始时间以及随后通过频率计算的时间运行。

示例

以下示例显示了跟踪数据库服务器启动环境的传感器的代码。传感器当前执行的是 $DATA_SEQ_ID 变量。

```
INSERT INTO ph_task
(
tk_name,
tk_type,
tk_group,
```

```
tk_description,
tk_result_table,
tk_create,
tk_execute,
tk_stop_time,
tk_start_time,
tk_frequency,
tk_delete
)
VALUES
(
"mon_sysenv",
"STARTUP SENSOR",
"SERVER",
"Tracks the database servers startup environment.",
"mon_sysenv",
"create table mon_sysenv (ID integer, name varchar(250), value lvarchar(1024))",
"insert into mon_sysenv select $DATA_SEQ_ID, env_name, env_value
FROM sysmaster:sysenv",
NULL,
NULL,
NULL,
"60 0:00:00"
);
```

以下示例显示收集有关正在使用的内存量的信息并在 **mon_memory_system** 表中存储信息的传感器的代码。如果表不存在，那么任务将创建表。该任务（每 30 分钟运行一次）将删除 **mon_memory_system** 表中存在时间超过 30 天的任何数据。

```
INSERT INTO ph_task
(
tk_name,
tk_group,
tk_description,
tk_result_table,
tk_create,
tk_execute,
tk_stop_time,
tk_start_time,
```

```
tk_frequency,
tk_delete
)
VALUES
("mon_memory_system",
"MEMORY",
"Server memory consumption",
"mon_memory_system",
"create table mon_memory_system (ID integer, class smallint, size int8,
        used int8, free int8 )",
"insert into mon_memory_system select $DATA_SEQ_ID, seg_class, seg_size,
        seg_blkused, seg_blkfree FROM sysmaster:sysseglst",
NULL,
NULL,
INTERVAL ( 30 ) MINUTE TO MINUTE,
INTERVAL ( 30 ) DAY TO DAY);
```

8.1.5 任务和传感器的操作

任务或传感器的操作是用于执行一个或多个操作的 SQL 语句或例程。

如果操作中仅包含单个操作，SQL 语句很有用。 如果操作中包含多个操作，使用 C 或 Java™ 编写的存储过程或用户定义的例程很有用。操作存储在 **ph_task** 表的 **tk_execute** 列中。

创建操作时具有很高的灵活性。 操作类型可包括：

- 执行 DML 操作。可使用传感器在表中插入或更新数据。可使用任务从表中删除较旧的数据。
- 执行管理操作。可使用任务来运行 SQL 管理 API 函数，以管理数据库服务器。 例如，可创建一个任务来每两分钟执行一次检查点。
- 基于阈值执行操作。可使用 ph_threshold 表中的阈值确定是否必须运行某个任务操作。例如，可创建一个任务，该任务在可用共享内存量低于阈值时添加共享内存段。
- 创建警报以报告操作或警告存在潜在问题。例如，可创建一个任务，用于在用户会话终止时终止在 ph_alert 表中插入行的空闲用户。也可创建任务来监视备份，并在备份未执行时在 ph_alert 表中插入警告。

请在任务或传感器操作中使用以下变量：

- $DATA_TASK_ID：用于指示当前任务或传感器。该变量对应于 ph_task 表中 tk_id 字段的值。
- $DATA_SEQ_ID：用于指示任务或传感器的当前执行情况。该变量对应于 ph_task 表

中的 tk_sequence 字段和 ph_run 表中的 run_task_sequence 字段的值。

示例

以下操作是使用的 SQL 语句，供内置 **mon_command_history** 任务用于从 **command_history** 表中除去较旧的行。

```
DELETE FROM command_history
WHERE cmd_exec_time < (
SELECT CURRENT - value::INTERVAL DAY to SECOND
FROM ph_threshold
WHERE name = 'COMMAND HISTORY RETENTION' )
```

以下示例描述的是 SQL 语句，供内置 mon_vps 传感器用于向 mon_vps 结果表添加数据：

```
INSERT INTO mon_vps
SELECT $DATA_SEQ_ID, vpid, num_ready,
class, usecs_user, usecs_sys
FROM sysmaster:sysvplst
```

以下示例描述的是存储过程，用于终止空闲时间超过了阈值设置值的用户会话，并且向 ph_alert 表添加警报。

```
/*
 ************************************************************
 *    Create a function that will find all users that have
 *     been idle for the specified time. Call the SQL admin API to
 *     terminate those users.    Create an alert to track which
 *     users have been terminated.
 ************************************************************
 */
CREATE FUNCTION idle_timeout( task_id INT, task_seq INT)
RETURNING INTEGER

DEFINE time_allowed INTEGER;
DEFINE sys_hostname CHAR(16);
DEFINE sys_username CHAR(257);
DEFINE sys_sid          INTEGER;
DEFINE rc               INTEGER;

    {*** Get the maximum amount of time to be idle ***}
    SELECT value::integer
            INTO time_allowed
```

```
            FROM ph_threshold
            WHERE name = "IDLE TIMEOUT";

    {*** Find all users who are idle longer than the threshold ***}
    FOREACH SELECT admin("gadmin","z",A.sid), A.username, A.sid, hostname
            INTO rc, sys_username, sys_sid, sys_hostname
            FROM sysmaster:sysrstcb A , sysmaster:systcblst B,
                 sysmaster:sysscblst C
            WHERE A.tid = B.tid
            AND C.sid = A.sid
            AND lower(name) in    ("sqlexec")
            AND CURRENT - DBINFO("utc_to_datetime",last_run_time) >
time_allowed UNITS MINUTE
            AND lower(A.username) NOT IN( "gbasedbt", "root")

            {*** If a user is successfully terminated, log ***}
            {*** the information into the alert table.       ***}
            IF rc > 0 THEN
                INSERT INTO ph_alert
                    (
                      ID, alert_task_id,alert_task_seq,
                      alert_type, alert_color,
                      alert_state,
                      alert_object_type, alert_object_name,
                      alert_message,
                      alert_action
                    ) VALUES (
                      0,task_id, task_seq,
                      "INFO", "GREEN",
                      "ADDRESSED",
                      "USER","TIMEOUT",
                       "User "||TRIM(sys_username)||"@"||TRIM(sys_hostname)||
                                  " sid("||sys_sid||")"||
                                " terminated due to idle timeout.",
                      NULL
                    );
            END IF
```

```
    END FOREACH;

    RETURN 0;

END FUNCTION;
```

8.1.6 创建组

可创建组来组织调度程序任务和传感器。

必须以用户 **gbasedbt** 或其他授权用户身份连接 **sysadmin** 数据库。

创建任务或传感器时，可在 **ph_task** 表的 **tk_group** 列中指定 **ph_group** 表中的组名。

要创建组，请执行以下操作：

使用 INSERT 语句将行添加到 **sysadmin** 数据库中的 **ph_group** 表中。

必须在 **group_name** 列中包含组的名称，并且在 **group_description** 列中包含组的描述。数据库服务器在 **group_id** 列中为组生成标识。

示例

以下示例添加名为 TABLES 的组：

```
INSERT INTO ph_group
(
group_name,
group_description
)
VALUES
(
"TABLES",
"Tasks that trim history and results tables."
);
```

8.1.7 创建阈值

可创建阈值来确定运行调度程序任务或传感器的条件。

必须以用户 **gbasedbt** 或其他授权用户身份连接 **sysadmin** 数据库。

阈值指定一个值，该值可用于与当前值进行比较，以便确定是否必须运行任务或传感器。

要创建阈值，请执行以下操作：

1. 使用 INSERT 语句添加 **ph_threshold** 表中以下列的值：

- **name**：阈值的名称
- **task_name**：ph_task 表中的任务的名称
- **value**：阈值的值
- **value_type**：阈值的数据类型（STRING 或 NUMERIC）
- **description**：阈值所执行操作的描述

2. 编写任务或传感器操作以使用阈值。

示例

以下示例为任务 Idle_timeout 添加阈值 IDLE TIMEOUT：

```
INSERT INTO ph_threshold
(
name,
task_name,
value,
value_type,
description)
VALUES
(
"IDLE TIMEOUT",
"Idle_timeout",
"60",
"NUMERIC",
"Maximum amount of time in minutes for non-gbasedbt users to be idle."
);
```

任务操作从当前时间中减去上一个用户操作的时间，并将该值与 ph_threshold 表中的值列进行比较。

8.1.8 创建警报

执行调度程序任务或传感器的操作时，可创建警报。

必须以用户 **gbasedbt** 或其他授权用户身份连接 **sysadmin** 数据库。

要创建警报，请执行以下操作：

使用 INSERT 语句在 **ph_alert** 表中添加行。 包含以下列的值：

- **ID**：生成的系统；请为该值使用 0。
- **alert_task_id**：必须引用 **ph_task** 表中的作业标识。

- **alert_task_seq**：必须引用 **ph_task** 表中的作业序号。
- **alert_type**：选择 INFO、WARNING 或 ERROR。
- **alert_color**：选择 GREEN、YELLOW 或 RED。
- **alert_state**：选择 NEW、IGNORED、ACKNOWLEDGED 或 ADDRESSED。
- **alert_object_type**：警报描述的对象类型，如 SERVER。
- **alert_object_name**：对象的名称。
- **alert_message**：描述警报的消息。
- **alert_action**：用于执行更正操作 SQL 语句或函数，或者为 NULL。

示例

以下示例添加一个警报，用于警告尚未执行备份。此代码片段是将 task_id 和 task_seq 用作自变量的存储过程的一部分。

```
INSERT INTO ph_alert
(
ID,
alert_task_id,
alert_task_seq,
alert_type,
alert_color,
alert_state,
alert_object_type,
alert_object_name,
alert_message,
alert_action
)
VALUES
(
0,
task_id,
task_seq,
"WARNING",
"RED",
"NEW",
"SERVER",
"dbspace_name",
"Dbspace ["||trim(dbspace_name)|| "] has never had a level-0 backup.
     Recommend taking a level-0 backup immediately.",
```

```
NULL
);
```

8.1.9 监视调度程序

可使用 gstat -g dbc 命令监视正在运行的调度程序线程。可在 ph_run 表中查看有关已完成的任务和传感器的信息。

调度程序在运行时使用以下两种线程：

- dbWorker：这种线程运行已调度的任务和传感器。
- dbScheduler：该线程准备已安排要运行的下一个任务或传感器。

要查看有关当前正在运行的任务和传感器以及将运行的下一个任务或传感器的信息，请使用 gstat -g dbc 命令。

要查看有关已完成的任务和传感器的信息，请查询 **sysadmin** 数据库中的 **ph_run** 表。 必须以用户 **gbasedbt** 或其他授权用户身份连接 **sysadmin** 数据库。

示例

gstat -g dbc 命令的以下输出显示两个 dbWorker 线程以及 dbScheduler 线程：

```
Worker Thread(0)      46fa6f10
====================================
Task:                  47430c18
Task Name:             mon_config_startup
Task ID:               3
Task Type:             STARTUP SENSOR
Last Error
     Number                -310
     Message               Table (gbasedbt.mon_onconfig)
                            already exists in database.
     Time                  09/11/2007 11:41
     Task Name              mon_config_startup

Task Execution:        onconfig_save_diffs

WORKER PROFILE
     Total Jobs Executed            10
     Sensors Executed                8
     Tasks Executed                  2
     Purge Requests                  8
```

```
    Rows Purged                    0

Worker Thread(1)    46fa6f80
====================================
Task:                4729fc18
Task Name:           mon_sysenv
Task ID:             4
Task Type:           STARTUP SENSOR
Task Execution:      insert into mon_sysenv select 1, env_name,
                     env_value FROM sysmaster:sysenv

WORKER PROFILE
    Total Jobs Executed            3
    Sensors Executed               2
    Tasks Executed                 1
    Purge Requests                 2
    Rows Purged                    0


Scheduler Thread    46fa6f80
====================================
Run Queue
    Empty
Run Queue Size          0
Next Task               7
Next Task Waittime      57
```

以下输出显示 **ph_run** 表中四个调度程序作业的历史记录：

```
SELECT * FROM ph_run;

RUN_ID              1
RUN_TASK_ID         2
RUN_TASK_SEQ        1
RUN_RETCODE         0
RUN_TIME            2009-07-20 13:04:59
RUN_DURATION        0.131850300007433
RUN_ZTIME           1248109468
RUN_BTIME           1248109468
```

```
RUN_MTIME           1248109499

RUN_ID              2
RUN_TASK_ID         3
RUN_TASK_SEQ        1
RUN_RETCODE         0
RUN_TIME            2009-07-20 13:04:59
RUN_DURATION        0.120845244247991
RUN_ZTIME           1248109468
RUN_BTIME           1248109468
RUN_MTIME           1248109499

RUN_ID              3
RUN_TASK_ID         4
RUN_TASK_SEQ        1
RUN_RETCODE         0
RUN_TIME            2009-07-20 13:04:59
RUN_DURATION        0.00254887164461759
RUN_ZTIME           1248109468
RUN_BTIME           1248109468
RUN_MTIME           1248109499

RUN_ID              2087
RUN_TASK_ID         7
RUN_TASK_SEQ        742
RUN_RETCODE         0
RUN_TIME            2009-09-09 11:09:51
RUN_DURATION        0.00489335523104662
RUN_ZTIME           1248109468
RUN_BTIME           1248109468
RUN_MTIME           1252508991
```

8.1.10 修改调度程序

可以修改调度程序任务、传感器、警报、阈值或组的属性。既可修改内置属性，也可修改您添加的属性。

必须以用户 **gbasedbt** 或其他授权用户身份连接 **sysadmin** 数据库。

要修改调度程序属性，请执行以下操作：

将 UPDATE 语句用于 **sysadmin** 数据库中的相应调度程序表。

示例

以下示例停止运行名为 task1 的任务：

```
UPDATE ph_task
   SET tk_enable = "F"
      WHERE tk_name = "task1";
```

以下示例将内置传感器 mon_profile 收集数据的时间量更改为 99 天：

```
UPDATE ph_task
SET tk_delete = "INTERVAL ( 99 ) DAY TO DAY"
WHERE tk_name = "mon_profile";
```

以下示例将名为 COMMAND HISTORY RETENTION 的阈值更改为 20，这样 **command_history** 表会将有关 SQL 管理 API 命令的信息保留 20 天：

```
UPDATE ph_threshold SET value = "20 0:00:00"
WHERE name = "COMMAND HISTORY RETENTION";
```

8.2 自动监视和更正操作概述

可以使用 SQL 管理 API、调度程序和向下钻取查询来管理自动维护、监视和管理任务。

GBase 8s 的这些组件简化了复杂系统中的信息收集和服务器维护。

SQL 管理 API

SQL 管理 API 用于通过 SQL 函数执行远程管理。因为 SQL 管理 API 操作完全在 SQL 中执行，所以可在客户机工具中使用这些函数来管理数据库服务器。

调度程序

调度程序是一组任务，用于在预定义时间或按照服务器内部确定的时间来执行 SQL 语句。SQL 语句可以收集信息或监视和调整服务器。

向下钻取查询

向下钻取查询提供了有关最近执行的 SQL 语句的统计信息，以便跟踪各个 SQL 语句的性能并分析语句历史记录。

可以在服务器 HDR 对的主服务器上使用 SQL 管理 API 和调度程序。

这些工具中的每一个工具都需要额外的磁盘空间以用于存储信息。

还可以使用基于 PHP 的 Web 浏览器管理工具 OpenAdmin Tool (OAT) (OAT) 从单一位置管理多个数据库服务器实例。可以使用 OAT 执行的一些任务包括：

- 通过 SQL 管理 API 和调度程序定义和管理自动执行任务
- 为 SQL 语句的分析和调整创建并显示性能柱状图

8.2.1 使用 SQL 管理 API 执行远程管理

您可以使用 SQL 管理 API 来利用 SQL 语句执行远程管理任务。

SQL 管理 API 函数采用一个或多个自变量来定义任务。许多任务也可使用命令行实用程序来完成。使用 SQL 管理 API 函数的优点是可从其他数据库服务器远程运行这些函数。而运行命令行实用程序命令时，必须直接连接到数据库服务器。

使用 SQL 管理 API 可执行以下类型的管理任务：

- 控制数据压缩
- 更新配置参数
- 检查数据、分区和扩展数据块一致性，控制 B 型树扫描程序，以及强制执行检查点
- 设置和管理 Enterprise Replication
- 设置和管理高可用性集群
- 控制日志记录和逻辑日志
- 控制共享内存和添加缓冲池
- 控制镜像过程
- 控制决策支持查询
- 更改服务器方式
- 添加、删除和配置存储空间
- 控制 SQL 语句高速缓存
- 控制和配置 SQL 跟踪
- 动态启动和停止侦听控制线程
- 执行其他任务，例如移动 sysadmin 数据库、终止会话或添加虚拟处理器

有关 SQL 管理 API 的更多信息，请参阅《GBase 8s 管理员参考》。

SQL 管理 API 的 admin() 和 task() 函数

SQL 管理 API 中包含两个在 **sysadmin** 数据库中定义的函数：admin() 和 task()。

这两个函数执行相同任务，但是返回不同格式的结果。task() 函数返回描述命令结果的字符串。admin() 函数返回整数。

缺省情况下，只有 **gbasedbt** 用户可连接到 **sysadmin** 数据库。如果 **root** 用户或 DBSA 组的成员被授予连接到 **sysadmin** 数据库的特权，那么该 **root** 用户或 DBSA 组的成员还可以运行 SQL 管理 API 的 task() 和 admin() 函数。

可以使用 EXECUTE FUNCTION 语句执行 admin() 和 task() 函数。 例如，等同于 gcheck -ce 命令的以下 SQL 语句可指示数据库服务器检查扩展数据块：

```
EXECUTE FUNCTION admin("check extents");
```

可在调度程序任务操作中使用 SQL 管理 API 函数。例如，可以通过在任务操作中使用以下语句来定义用于创建数据库空间的任务：

```
EXECUTE FUNCTION admin("create dbspace","dbspace2","/work/dbspace2","20
MB");
```

有关使用 admin() 和 task() 函数以及示例的信息，请参阅《GBase 8s 管理员参考》。

查看 SQL 管理 API 历史记录

可以查看 **sysadmin** 数据库中 **command_history** 表内前 30 天运行的所有 SQL 管理 API 函数的历史记录。

必须以 **gbasedbt** 用户或其他授权用户身份连接 **sysadmin** 数据库。

command_history 表显示了管理任务是通过 admin() 还是 task() 函数执行的，并显示了有关运行命令的用户、运行命令的时间、命令以及数据库服务器完成运行命令时返回的消息的信息。

要显示命令历史记录，请执行以下操作：

使用 SELECT 语句从 **command_history** 表返回数据。

以下示例显示过去 30 天的所有命令历史记录：

```
SELECT * FROM command_history;
```

下表显示了采样命令和采样 command_history 表中的相关联结果。有关 command_history 表中所有信息的描述，请参阅《GBase 8s 管理员参考》。

表 1. command_history 表中某些信息的示例

命令	样本返回消息
设置 SQL 跟踪为打开	SQL 跟踪打开，并带有 1000 个大小为 2024 字节的缓冲区。
创建数据库空间	空间“space12”已添加。
检查点	检查点已完成。
添加日志	已将 3 个逻辑日志添加到数据库空间日志数据库。

控制 command_history 表的大小

可以缩短 **command_history** 表的保留期或从该表中除去行，从而防止该表变得过大。

必须以 **gbasedbt** 用户或其他授权用户身份连接 **sysadmin** 数据库。

缺省情况下，**command_history** 表中的行在 30 天之后会自动除去。保留期由 ph_threshold 表中的 COMMAND HISTORY RETENTION 行进行控制。

要缩短保留期，请执行以下操作：

使用 UPDATE 语句修改 ph_threshold 表中 COMMAND HISTORY RETENTION 行的值。

以下示例将保留期设置为 25 天：

```
UPDATE ph_threshold
SET value = "25"
WHERE name = "COMMAND HISTORY RETENTION";
```

可以使用 DELETE 或 TRUNCATE TABLE 之类的 SQL 命令手动从此表中除去数据。也可在 ph_task 表中创建任务以从 **command_history** 表清除数据。

以下示例显示了监视 **command_history** 表中的数据量并当表太旧时清除数据的任务。

```
INSERT INTO ph_task
( tk_name, tk_type, tk_group, tk_description, tk_execute,
tk_start_time, tk_stop_time, tk_frequency )
VALUES
("mon_command_history",
"TASK",
"TABLES",
"Monitor how much data is kept in the command history table",
"delete from command_history where cmd_exec_time < (
        select current - value::INTERVAL DAY to SECOND
        from ph_threshold
        where name = 'COMMAND HISTORY RETENTION' ) ",
DATETIME(02:00:00) HOUR TO SECOND,
NULL,
INTERVAL ( 1 ) DAY TO DAY);
```

8.2.2 向下钻取查询

可以使用向下钻取查询（也称为 SQL 跟踪）来收集有关运行的每个 SQL 语句的统计信息以及分析语句历史记录。

SQL 跟踪可帮助您回答如下问题：

- SQL 语句耗费多少时间？
- 单个语句使用多少资源？
- 语句执行耗费多少时间？

- 等待每个资源耗费多少时间？

统计信息存储在循环缓冲区（内存中名为 syssqltrace 的伪表）中，即存储在 sysmaster 数据库中。您可以动态地调整循环缓冲区的大小。

缺省情况下，SQL 跟踪已关闭，但是您可以对所有用户或一组特定用户打开此功能。在启用具有缺省配置的 SQL 跟踪时，数据库服务器会跟踪运行的上 1000 条 SQL 语句，以及这些语句的概要统计信息。还可以全局禁用 SQL 跟踪或禁用对特定用户的 SQL 跟踪。

如果您计划保存大量历史信息，那么 SQL 跟踪所需的内存相当大。SQL 跟踪所需的缺省空间量为 2 MB。可以根据需求增加或减少存储量。

显示的信息包括：

- 运行命令的用户的用户标识
- 数据库会话标识
- 数据库的名称
- SQL 语句的类型
- SQL 语句执行的持续时间
- 该语句完成的时间
- 带有语句类型的 SQL 语句文本或函数调用列表（也称为堆栈跟踪），例如：

```
procedure1() calls procedure2() calls procedure3()
```

- 统计信息包括：
  - 缓冲区读取和写入的数目
  - 页面读取和写入的数目
  - 排序和磁盘排序的数目
  - 锁请求和等待的数目
  - 逻辑日志记录的数目
  - 索引缓冲区读取的数目
  - 估计的行数
  - 优化器估计成本
  - 返回的行数
- 数据库隔离级别。

也可指定跟踪中要包含的信息的升级级别，如下所示：

- 低级别跟踪，缺省情况下已启用，用于捕获以下示例中显示的信息。该信息包含语句统计信息、语句文本和语句迭代器。
- 中等级别跟踪，除了用于捕获低级别跟踪中包含的所有信息外，还捕获表名、数据

库名称和存储过程堆栈的列表。

- 高级别跟踪，除了用于捕获中等级别跟踪中包含的所有信息外，还捕获主变量。

所跟踪的信息量影响该历史数据所需的内存量。

您可以在任何时候启用和禁用跟踪，并可在数据库服务器运行时更改跟踪缓冲区的数目和大小。如果调整跟踪缓冲区的大小，那么数据库服务器将尝试维护缓冲区的内容。如果增大这些参数，数据将不会被截断。但是，如果缓冲区的数目或大小减小，那么跟踪缓冲区中的数据将被截断或丢失。

缓冲区的数目确定了所跟踪的 SQL 语句数。每个缓冲区包含单个 SQL 语句的信息。缺省情况下，各个跟踪缓冲区的大小是固定的。如果缓冲区中存储的文本信息超过跟踪缓冲区的大小，那么数据被截断。

以下示例显示 SQL 跟踪信息：

```
select * from syssqltrace where sql_id = 5678;

sql_id              5678
sql_address         4489052648
sql_sid             55
sql_uid             2053
sql_stmttype        6
sql_stmtname        INSERT
sql_finishtime      1140477805
sql_begintxtime     1140477774
sql_runtime         30.86596333400
sql_pgreads         1285
sql_bfreads         19444
sql_rdcache         93.39127751491
sql_bfidxreads      5359
sql_pgwrites        810
sql_bfwrites        17046
sql_wrcache         95.24815205913
sql_lockreq         10603
sql_lockwaits       0
sql_lockwttime      0.00
sql_logspace        60400
sql_sorttotal       0
sql_sortdisk        0
sql_sortmem         0
```

```
sql_executions      1
sql_totaltime       30.86596333400
sql_avgtime         30.86596333400
sql_maxtime         30.86596333400
sql_numiowaits      2080
sql_avgiowaits      0.014054286131
sql_totaliowaits    29.23291515300
sql_rowspersec      169.8958799132
sql_estcost         102
sql_estrows         1376
sql_actualrows      5244
sql_sqlerror        0
sql_isamerror       0
sql_isollevel       2
sql_sqlmemory       32608
sql_numiterators    4
sql_database        db3
sql_numtables       3
sql_tablelist       t1
sql_statement       insert into t1 select {+ AVOID_FULL(sysindices) } 0, tabname
```

有关所有表行的解释，请参阅《GBase 8s 管理员参考》的 sysmaster 数据库部分中有关 syssqltrace 表的信息。

使用 SQLTRACE 配置参数指定启动 SQL 跟踪信息

使用 SQLTRACE 配置参数可控制数据库服务器启动时的缺省跟踪行为。缺省情况下，不设置该参数。所设置的信息包括要跟踪的 SQL 语句数目和跟踪方式。

可以修改 onconfig 文件的任何用户均可修改 SQLTRACE 配置参数的值，并可影响启动配置。但是，只有用户 **gbasedbt**、**root** 或被授予 **sysadmin** 数据库连接特权的 DBSA 才可以使用 SQL 管理 API 命令来修改 SQL 跟踪的运行时状态。

要在数据库服务器启动时指定 SQL 跟踪信息，请执行以下操作：

1. 设置 onconfig 文件中的 SQLTRACE 配置参数。
2. 重新启动数据库服务器。

示例

onconfig 文件中的以下设置指定数据库服务器将收集有关系统上所有用户执行过的低级别信息，最多收集 2000 条，并分配大约 4 MB 内存 (2000 * 2 KB)。

```
SQLTRACE level=LOW,ntraces=2000,size=2,mode=global
```

如果仅使用部分已分配的缓冲区空间（例如，缓冲区空间的 42%），那么所分配的内存量仍然为 2 KB。

如果不想设置 SQLTRACE 配置参数并重新启动了服务器，那么可以运行以下 SQL 管理 API 命令，该命令提供的功能与为当前会话设置 SQLTRACE 的功能相同：

```
EXECUTE FUNCTION task("set sql tracing on", 100,"1k","med","user");
```

在以用户方式启用 SQL 跟踪系统后，就可以启用对每个用户的跟踪。请参阅启用 SQL 跟踪。

有关使用 task() 和 admin() 函数的更多信息，请参阅《GBase 8s 管理员参考》。

有关 SQLTRACE 配置参数的更多信息（包括某些字段的最小值和最大值），请参阅《GBase 8s 管理员参考》。

全局禁用 SQL 跟踪或禁用对某个会话的 SQL 跟踪

即使 SQLTRACE 配置参数中指定的方式为 global 或 user，但如果要完全关闭所有用户和全局跟踪，并取消分配给 SQL 跟踪当前正在使用的资源，也可以禁用 SQL 跟踪。缺省情况下，禁用对所有用户的 SQL 跟踪。

必须以用户 **gbasedbt** 或其他授权用户身份连接 **sysadmin** 数据库。

要禁用全局 SQL 跟踪，请运行使用 **set sql tracing off** 自变量的 SQL 管理 API task() 或 admin() 函数。

要禁用对特定会话的 SQL 跟踪，请运行 set sql tracing off 作为第一个自变量，会话标识号作为第二个自变量的 task() 或 admin() 函数。

示例

以下示例全局禁用 SQL 跟踪：

```
EXECUTE FUNCTION task('set sql tracing off');
(expression) SQL tracing off.


1 row(s) retrieved.
```

以下示例对标识为 47 的会话禁用 SQL 跟踪：

```
EXECUTE FUNCTION task(“set sql user tracing off”,47);
```

有关使用 task() 或 admin() 函数的更多信息，请参阅《GBase 8s SQL 指南：语法》。

启用对特定用户的 SQL 跟踪

在指定 user 作为 SQLTRACE 配置参数中的方式后，必须运行 SQL 管理 API task() 或 admin() 函数来打开对特定用户的 SQL 历史记录跟踪。

必须以用户 **gbasedbt** 或其他授权用户身份连接 **sysadmin** 数据库。

无需启用全局 SQL 跟踪，即可对特定用户进行 SQL 跟踪。

要启用对特定用户的 SQL 跟踪，请运行使用 set sql tracing on 作为第一个自变量，用户会话标识作为第二个自变量的 task() 或 admin() 函数。

要对除 **root** 或 **gbasedbt** 之外的所有用户启用用户 SQL 跟踪，可运行使用 **set sql tracing on** 自变量和定义这些用户的信息的 task() 或 admin() 函数。

示例

以下示例对会话标识为 74 的用户启用 SQL 跟踪：

```
EXECUTE FUNCTION task("set sql user tracing on", 74);
```

以下示例启用对当前连接到系统的用户（只要它们未以用户 root 或 gbasedbt 身份登录）的 SQL 语句跟踪。

```
dbaccess sysadmin -<<END
 execute function task("set sql tracing on", 1000, 1,"low","user");
  select task("set sql user tracing on", session_id)
          FROM sysmaster:syssessions
  WHERE username not in ("root","gbasedbt");
END
```

有关 task() 和 admin() 函数的更多信息，请参阅《GBase 8s 管理员参考》。

启用对某个会话的全局 SQL 跟踪

可以通过运行 SQL 管理 API task() 或 admin() 函数，对当前会话启用全局 SQL 跟踪。

必须以用户 **gbasedbt** 或其他授权用户身份连接 **sysadmin** 数据库。

缺省情况下，全局 SQL 跟踪未启用。可以设置 SQLTRACE 配置参数以永久启用全局跟踪。

要对当前数据库服务器会话启用全局用户 SQL 历史记录跟踪，请运行使用 **set sql tracing on** 自变量的 SQL 管理 API task() 或 admin() 函数。

示例

以下示例对所有用户启用全局低级别 SQL 跟踪：

```
EXECUTE FUNCTION task("set sql tracing on", 1000, 1,"low","global");
```

如果在语句运行后有新用户登录到系统，那么可以启用对该新用户的跟踪。请参阅启用 SQL 跟踪。

有关 task() 和 admin() 函数的更多信息，请参阅《GBase 8s SQL 指南：语法》。

GBASE®

南大通用数据技术股份有限公司
General Data Technology Co., Ltd.

微信二维码

技术支持热线：400-013-9696