Red Hat Enterprise Linux 6

全局文件系统 2

Red Hat 全局文件系统 2

版 7

Last Updated: 2017-10-12

Red Hat Enterprise Linux 6 全局文件系统 2

Red Hat 全局文件系统 2
版 7

法律通告

摘要

本书提供了有关 Red Hat Enterprise Linux 6 安装、配置和维护 Red Hat GFS2（Red Hat 全局文件系统 2）的信息。

目录

简介

本书提供有关配置和维护 Red Hat GFS2（Red Hat 全局文件系统 2）的信息，GFS2 包含在 Resilient Storage Add-On 中。

1. 读者

本书主要面向熟悉以下活动的 Linux 系统管理员：

- 包括内核配置在内的 Linux 系统管理流程
- 安装和配置共享存储网络，比如光纤 SAN

2. 相关文档

有关使用 Red Hat Enterprise Linux 的详细信息请参考以下资源：

- *《安装指南》*—记录有关安装 Red Hat Enterprise Linux 6 的信息。
- *《部署指南》*—记录有关部署、配置及管理 Red Hat Enterprise Linux 6 的信息。
- *《存储管理指南》*—提供如何有效管理 Red Hat Enterprise Linux 6 中的存储设备和文件系统的说明。

有关 Red Hat Enterprise Linux 6 中的高可用性附加组件和 Resilient Storage Add-On 详情请参考以下资源：

- *《高可用性附加组件概述》*—为您提供 Red Hat 高可用性附加组件高级概述。
- *《集群管理》*—提供有关安装、配置和管理高可用性附加组件的信息。
- *《LVM 管理员指南》*—提供逻辑卷管理器（LVM）的描述，其中包括在集群环境中运行 LVM 的信息。
- *《设备映射器多路径》*—提供有关使用 Red Hat Enterprise Linux 设备映射器多路径功能的信息。
- *《负载平衡器管理》*—提供使用负载平衡器附加组件配置高性能系统和服务的信息。负载平衡器附加组件是一组整合的软件组件，可提供在一组真实服务器之间平衡 IP 负载的 Linux 虚拟服务器（LVS）。
- *《发行注记》*—提供有关 Red Hat 产品当前发行本信息。

Red Hat Cluster Suite 文档及其他 Red Hat 文档在 Red Hat Enterprise Linux 文档 CD 中有 HTML、PDF 以及 RPM 版本，其在线地址为 https://access.redhat.com/site/documentation/。

3. 我们需要反馈意见！

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla: http://bugzilla.redhat.com/ against the product **Red Hat Enterprise Linux 6** and the component **doc-Global_File_System_2**. When

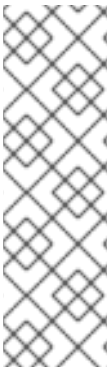
submitting a bug report, be sure to mention the manual's identifier:

```
rh-gfs2(EN)-6 (2014-10-8T15:15)
```

如果您有针对文档的建议，请尽量具体描述。如果您发现任何错误，请附带章节号以及上下文，以便我们查找。

第 1 章 GFS2 概述

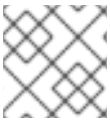
Red Hat GFS2 文件系统包含在 Resilient Storage Add-On 中。它是固有文件系统，直接与 Linux 内核文件系统界面（VFS 层）互动。当作为集群文件系统使用时，GFS2 采用分布式元数据和多个日志（multiple journal）。Red Hat 只支持将 GFS2 文件系统作为在 High Availability Add-On 中的部署使用。



注意

虽然 GFS2 文件系统既可以作为独立系统部署，也可以作为集群配置的一部分，但在 Red Hat Enterprise Linux 6 发行本中，Red Hat 不支持将 GFS2 作为单节点文件系统使用。Red Hat 支持很多高性能单节点文件系统，这些系统是为单节点优化的，因此一般比集群文件系统的消耗要小。Red Hat 建议在只需要将单一节点挂载到文件系统的情况下，这些文件系统应首先使用这些文件系统，而不是 GFS2。

Red Hat 将继续支持单一节点 GFS2 文件系统挂载集群文件系统快照（例如：用于备份）。



注意

Red Hat 不支持在部署超过 16 个节点的集群文件系统中使用 GFS2。

GFS2 基于 64 位构架，理论上可提供 8 EB 文件系统。但是，目前支持的 64 位硬件的最大 GFS2 文件系统为 100 TB，为 32 位硬件支持的最大 GFS2 文件系统为 16 TB。如果您的系统要求更大的 GFS2 文件系统，请联络您的 Red Hat 服务代表。

当决定您的文件系统大小时，您应该考虑您的恢复需求。在一个很大的文件系统中运行 **fsck.gfs2** 命令需要很长时间，且消耗大量内存。另外，在磁盘或者磁盘子系统失败事件中，恢复时间受您的备份介质速度的限制。有关 **fsck.gfs2** 所需内存量的详情请参考 第 4.11 节 “修复文件系统”。

在集群中进行配置时，可使用 High Availability Add-On 配置和管理工具对Red Hat GFS2 节点进行配置和管理。然后Red Hat GFS2 会在集群的 GFS2 节点间提供数据共享，即在 GFS2 节点间提供单一、一致的文件系统名称查看。这样就允许不同节点中的进程可以共享 GFS2 文件，其方法与同一节点中的进程可共享本地文件系统中的文件相同，没有明显的差别。有关 High Availability Add-On 的详情请参考*《配置和管理 Red Hat 集群》*。

在 LVM 之外使用 GFS2 文件系统时，Red Hat 只支持在 CLVM 逻辑卷中创建的 GFS2 文件系统。CLVM 包含在 Resilient Storage Add-On 中。这是在集群范围内部署 LVM，由在集群中管理 LVM 逻辑卷的 CLVM 守护进程 **clvmd** 启用。该守护进程可让 LVM2 在集群间管理逻辑卷，允许集群中的所有节点共享该逻辑卷。有关 LVM 卷管理器的详情请参考*《LVM 管理》*。

gfs2.ko 内核模块部署 GFS2 文件系统，该文件在 GFS2 集群节点中载入。

注意

将 GFS2 文件系统配置为集群文件系统时，必须确定该集群中的所有节点都可访问共享存储。在不对称集群配置中，即有些节点可访问共享存储而另一些不可以的情况下部支持此功能。这并不需要将所有节点都挂载到 GFS2 文件系统中。

本章提供一些基本、简短资料作为参考，帮助您了解 GFS2，其中包括以下小节：

- 第 1.1 节 “新的和更改的功能”
- 第 1.2 节 “设置 GFS2 前的准备”

- 第 1.4 节 “GFS 和 GFS2 之间的差别”
- 第 1.3 节 “安装 GFS2”
- 第 2.9 节 “GFS2 节点锁定”

1.1. 新的和更改的功能

本小节列出了 Red Hat Enterprise Linux 6 初始发行本，以及后续版本中所包含的 GFS2 文件系统，以及 GFS2 文档的新功能和更改的功能。

1.1.1. Red Hat Enterprise Linux 6.0 中新的和更改的功能

Red Hat Enterprise Linux 6.0 包括以下文档和功能更新和更改。

- 在 Red Hat Enterprise Linux 6 发行本中，Red Hat 不支持将 GFS2 作为单节点文件系统使用。
- 在 Red Hat Enterprise Linux 6 发行本中，GFS2 文件系统中的 **gfs2_convert** 命令较之在 GFS 中有所改进。有关这个命令的详情请参考 附录 B, *将文件系统从 GFS 转换为 GFS2*。
- Red Hat Enterprise Linux 6 支持 **discard**、**nodiscard**、**barrier**、**nobarrier**、**quota_quantum**、**statfs_quantum** 和 **statfs_percent** 挂载选项。有关挂载 GFS2 文件系统的详情请参考 第 4.2 节 “挂载文件系统”。
- Red Hat Enterprise Linux 6 版本的这个文档包含一个新的部分 第 2.9 节 “GFS2 节点锁定”。这部分描述了一些 GFS2 文件系统的内部信息。

1.1.2. Red Hat Enterprise Linux 6.1 中新的和更改的功能

Red Hat Enterprise Linux 6.1 包括以下文档和功能更新和更改。

- 从 Red Hat Enterprise Linux 6.1 发行本开始，GFS2 支持标准 Linux 配额工具。GFS2 配额管理文档请参考 第 4.5 节 “GFS2 配额管理”。

  在 Red Hat Enterprise Linux 之前的发行本中，GFS2 需要使用 **gfs2_quota** 命令管理配额。现在在 附录 A, *使用 **gfs2_quota** 命令执行 GFS2 配额管理*中记录了 **gfs2_quota** 命令。
- 这个文档现在包含一个新章节，第 5 章 *诊断并修正 GFS2 文件系统的问题*。
- 对整篇文档进行了技术修正和说明。

1.1.3. Red Hat Enterprise Linux 6.2 中新的和更改的功能

Red Hat Enterprise Linux 6.2 包括以下文档和功能更新及更改。

- 从 Red Hat Enterprise Linux 6.2 开始，GFS2 支持 **tunegfs2** 命令，它代替了 **gfs2_tool** 命令的一些功能。有关详情请查看 **tunegfs2** man page。

  更新以下小节，提供不需要 **gfs2_tool** 命令的管理步骤：

  - 第 4.5.4 节 “使用 **quotasync** 命令同步配额” 和 第 A.3 节 “使用 **gfs2_quota** 命令同步配额” 现在论述如何使用 **quota_quantum=** 挂载选项修改 **quota_quantum** 参数的默认值，即 60 秒。

- 第 4.10 节 “在文件系统中挂起一个动作” 现在论述如何使用 **dmsetup suspend** 命令挂起对文件系统的写入活动。

- 本文档包括一个新附录 附录 C, *GFS2 跟踪点和 debug glock 文件*。这个附录论述了 **glock debugfs** 界面和 GFS2 跟踪点。该附录面向熟悉文件系统，并想要了解更多 GFS2 设计，以及如何 debug GFS2 具体问题的高级用户。

1.1.4. Red Hat Enterprise Linux 6.3 中新的和更改的功能

从 Red Hat Enterprise Linux 6.3 开始，这个文档包含新的一章 第 2 章 *GFS2 配置及操作注意事项*，提供优化 GFS2 性能的推荐方法，其中包括生成、使用和维护 GFS2 文件系统的建议。

另外，还对整个文档进行小的说明和修正。

1.1.5. Red Hat Enterprise Linux 6.4 中新的和更改的功能

在 Red Hat Enterprise Linux 6.4 发行本中，已对 第 2 章 *GFS2 配置及操作注意事项* 更新了一些说明。

1.1.6. Red Hat Enterprise Linux 6.6 中新的和更改的功能

从 Red Hat Enterprise Linux 6.6 发行本开始，本文档包含一个新的章节 第 6 章 *在 Pacemaker 集群中配置 GFS2 文件系统*。本章提供可设置包含 GFS2 文件系统的 Pacemaker 集群所需步骤概述。

另外，还对整个文档进行小的说明和修正。

1.2. 设置 GFS2 前的准备

安装和设定 GFS2 前，请记录 GFS2 文件系统的以下主要特征：

GFS2 节点

决定使用集群中的哪个节点挂载 GFS2 文件系统。

文件系统数

决定最初创建的 GFS2 文件系统数（以后可以添加更多的文件系统）。

文件系统名称

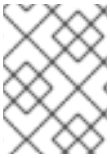
为每个文件系统取一个独特名称。该名称必须与集群中的所有 **lock_dlm** 文件系统不同。每个文件系统名称都要求使用参数变量格式。例如：在本书中的一些示例步骤中使用 **mydata1** 和 **mydata2**。

日志

决定 GFS2 文件系统的日志数。每个挂载到 GFS2 文件系统的节点都需要一个日志。GFS2 允许之后动态添加日志作为挂载文件系统的附加服务器。有关在 GFS2 文件系统中添加日志的详情请参考 第 4.7 节 “在文件系统中添加日志”。

存储设备和分区

决定在文件系统中用来创建逻辑卷（使用 CLVM）的存储设备和分区。



注意

在同一目录中同时进行多个生成和删除操作时，可能会看到 GFS2 的性能下降。如果这会造成系统性能问题，则应该尽量将某个节点中的文件生成和删除指定到特定目录。

有关生成、使用和维护 GFS2 文件系统的推荐方法，请参考 第 2 章 *GFS2 配置及操作注意事项*。

1.3. 安装 GFS2

除 Red Hat High Availability Add-On 所需软件包外，还必须为 GFS2 安装 **gfs2-utils**，为 Clustered Logical Volume Manager（CLVM）安装 **lvm2-cluster**。**lvm2-cluster** 和 **gfs2-utils** 软件包是 ResilientStorage 频道的一部分，必须在安装这些软件包前启用该频道。

您可以使用以下 **yum install** 命令安装 Red Hat High Availability Add-On 软件包：

```
# yum install rgmanager lvm2-cluster gfs2-utils
```

有关 Red Hat High Availability Add-On 及集群管理的一般信息请查看 *《集群管理》* 手册。

1.4. GFS 和 GFS2 之间的差别

这部分列出了相比 GFS，GFS2 提供的改进和变化。

从 GFS 迁移到 GFS2 时，需使用 **gfs2_convert** 程序将 GFS 文件系统转换为 GFS2。有关 **gfs2_convert** 程序的详情请参考 附录 B, *将文件系统从 GFS 转换为 GFS2*。

1.4.1. GFS2 命令名称

通常 GFS2 和 GFS 的功能是一致的，只是在文件系统命令名称中要将 GFS 改为 GFS2。表 1.1 “GFS 和 GFS2 命令” 中显示了功能等同的 GFS 和 GFS2 命令和功能。

表 1.1. GFS 和 GFS2 命令

GFS 命令	GFS2 命令	描述
mount	**mount**	挂载文件系统。系统可以确定文件系统的类型是 GFS 还是 GFS2。有关 GFS2 挂载选项的详情请参考 gfs2_mount(8) man page。
umount	**umount**	卸载文件系统
fsck **gfs_fsck**	**fsck** **fsck.gfs2**	检查并修复卸载的文件系统。
gfs_grow	**gfs2_grow**	增大挂载的文件系统。
gfs_jadd	**gfs2_jadd**	在挂载的文件系统中添加日志。

GFS 命令	GFS2 命令	描述
`gfs_mkfs` `mkfs -t gfs`	`mkfs.gfs2` `mkfs -t gfs2`	在存储设备中创建文件系统。
`gfs_quota`	`gfs2_quota`	在挂载的文件系统中管理配额。从 Red Hat Enterprise Linux 6.1 发行本开始，GFS2 支持标准 Linux 配额工具。有关在 GFS2 中进行配额管理的详情请参考 第 4.5 节 “GFS2 配额管理”。

`gfs_tool`	`tunegfs2`
	挂载参数
	`dmsetup suspend`
		配置、调整文件系统或者收集文件系统信息。从 Red Hat Enterprise Linux 6.2 开始支持 `tunegfs2` 命令。另外还有 `gfs2_tool` 命令。

`gfs_edit`	`gfs2_edit`	显示、输出或者编辑文件系统内部结构。`gfs2_edit` 命令可用于 GFS 文件系统，也可用于 GFS2 文件系统。
`gfs_tool setflag jdata/inherit_jdata`	`chattr +j` (preferred)	在文件或者目录中启用日志功能。
`setfacl/getfacl`	`setfacl/getfacl`	为文件或者目录设置或者获得文件访问控制。
`setfattr/getfattr`	`setfattr/getfattr`	设置或者获得文件的扩展属性。

GFS2 文件系统命令支持选项的完整列表请参考那些命令的 man page。

1.4.2. GFS 和 GFS2 之间的其他不同之处

这部分总结了在 第 1.4.1 节 “GFS2 命令名称” 中没有论述的 GFS 和 GFS2 管理中的其他不同之处。

上下文关联路径名

GFS2 文件系统不支持上下文关联路径名，该路径名允许生成指向各种目的地文件或者目录的符号链接。在 GFS2 中，您可以使用 `mount` 命令的 `bind` 选项实现这个功能。有关在 GFS2 中管理路径名的详情请参考 第 4.12 节 “绑定挂载以及上下文关联路径名”。

gfs2.ko 模块

使用 GFS 文件系统的内核模块是 `gfs.ko`。使用 GFS2 文件系统的内核模块是 `gfs2.ko`。

在 GFS2 中启用配额强制

在 GFS2 文件系统中，默认禁用配额强制，必须明确配置方可启用该功能。有关启用和禁用配额强制的详情请参考 第 4.5 节 “GFS2 配额管理”。

数据日志记录

GFS2 file systems support the use of the **chattr** command to set and clear the **j** flag on a file or directory. Setting the **+j** flag on a file enables data journaling on that file. Setting the **+j** flag on a directory means "inherit jdata", which indicates that all files and directories subsequently created in that directory are journaled. Using the **chattr** command is the preferred way to enable and disable data journaling on a file.

动态添加日志

在 GFS2 文件系统中，日志是存在于文件系统之外的内嵌元数据，这就需要在添加日志前，扩展包含该文件系统的逻辑卷大小。在 GFS2 文件系统中，日志是纯文本文件（虽然是隐藏的）。这意味着对于 GFS2 文件系统来说，只要在该文件系统中有可放置附加日志的空间，就可以动态添加日志将其作为附加服务器挂载到文件系统中。有关在 GFS2 文件系统中添加日志的详情请参考 第 4.7 节 “在文件系统中添加日志”。

删除 atime_quantum 参数

GFS2 文件系统不支持 **atime_quantum** 可调节参数，GFS 文件系统可使用该参数指定 **atime** 更新的频率。在 GFS2 中支持 **relatime** 和 **noatime** 挂载选项。建议您使用 **relatime** 挂载选项获得与在 GFS 中使用 **atime_quantum** 参数时得到的类似行为。

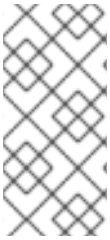
mount 命令的 data= 选项

在挂载 GFS2 文件系统时，您可以指定 **mount** 命令的 **data=ordered** 或者 **data=writeback** 选项。当设定 **data=ordered** 时，某个事务修改的用户数据会在该事务被提交到磁盘前被冲入磁盘。这样可以让用户无法在崩溃后的文件中看到未初始化的块。设定 **data=writeback** 时，用户数据会在磁盘有数据后的任何时候被写入磁盘。这不会提供 **ordered** 模式可提供的一致性保障，但对某些工作负载来说可稍微加快一些速度。默认设置是 **ordered** 模式。

gfs2_tool 命令

gfs2_tool 为 GFS2 支持的选项组与 **gfs_tool** 命令为 GFS 支持的选项组不同：

- **gfs2_tool** 命令支持 **journals** 参数，它可输出当前配置日志有关信息，其中包括文件系统包含的日志数。
- **gfs2_tool** 命令不支持 **counters** 标签，而 **gfs_tool** 命令可使用该标签显示 GFS 统计。
- The **gfs2_tool** command does not support the **inherit_jdata** flag. To flag a directory as "inherit jdata", you can set the **jdata** flag on the directory or you can use the **chattr** command to set the **+j** flag on the directory. Using the **chattr** command is the preferred way to enable and disable data journaling on a file.



注意

从 Red Hat Enterprise Linux 6.2 开始，GFS2 支持 **tunegfs2** 命令，该命令代替了 **gfs2_tool** 命令的一些功能。有关详情请参考 **tunegfs2**(8) man page。**gfs2_tool** 命令的 **settune** 和 **gettune** 功能已经由 **mount** 命令的命令行选项代替，这样可在需要时使用 **fstab** 文件进行设置。

gfs2_edit 命令

gfs2_edit 命令为 GFS2 支持的选项组与 **gfs_edit** 命令为 GFS 支持的选项组不同。有关该命令每个版本具体支持的选项，请参考 **gfs2_edit** 和 **gfs_edit** man page。

1.4.3. GFS2 性能改进

GFS2 文件系统的很多功能与 GFS 文件系统在界面上没有什么不同，但文件系统性能会有提高。

GFS2 文件系统在以下方面提供改进的文件系统性能：

- 在大量使用单一目录时有较好的性能。
- 更快的同步 I/O 操作
- 更快的缓存读取（无锁定消耗）
- 对预先分配的文件有更快的直接 I/O （提供合理的较大 I/O 值，比如 4M 大的块）
- 普遍更快的 I/O 操作
- 执行 **df** 命令的速度更快，因为 **statfs** 调用的速度更快。
- 与 GFS 相比，改进了 **atime** 模式以减少 **atime** 生成的写入 I/O 操作数量。

GFS2 文件系统在以下方面提供更广泛和主流的支持：

- GFS2 是上游内核（整合到 2.6.19）的一部分。
- GFS2 支持以下功能。
  - 扩展的文件属性（**xattr**）
  - 通过标准 **ioctl()** 调用设置 **lsattr()** 和 **chattr()** 属性
  - 纳秒时间戳

GFS2 在文件系统的内在效率方面提供以下改进。

- GFS2 使用更少的内核内存。
- GFS2 需要非元数据生成数

  分配 GFS2 元数据不需要读取。多个日志中的元数据块副本由从锁定释放前的日志中调用的块进行管理。
- GFS2 的日志管理程序更为简单，它对未链接的内节点或者配额修改一无所知。
- **gfs2_grow** 和 **gfs2_jadd** 命令使用锁定防止多个事件同时运行。

- 为类似 **creat()** 和 **mkdir()** 的调用简化 ACL 编码。
- 在不重新挂载日志的情况下，恢复未链接的内节点以及配额和 **statfs** 的更改。

第 2 章 GFS2 配置及操作注意事项

全局文件系统 2（GFS2）文件系统允许集群中的多台计算机（“节点”）协同共享同一存储。为达到此合作目的，并在节点间保持数据一致，这些节点会为文件系统资源采用集群范围内的锁定方案。这个锁定方案使用类似 TCP/IP 的沟通方案交换锁定的信息。

您可以使用本章中所论述的建议方法改进性能，其中包括对创建、使用和维护 GFS2 文件系统的建议。

重要

确定您的 Red Hat High Availability Add-On 部署满足您的需要并可获得支持。部署前请联络授权Red Hat 代表确认您的配置。

2.1. 格式化注意事项

本小节提供如何格式化 GFS2 文件系统以达到最佳性能的建议。

2.1.1. 文件系统大小：越小越好

GFS2 是基于 64 位架构，理论上可部署 8EB 的文件系统。但目前对 64 位架构硬件支持的最大 GFS2 文件系统为 100TB，对 32 位架构硬件支持的最大 GFS2 文件系统为 16TB。

注：即使 GFS2 超大文件系统是可能的，但并不意味着建议您采用。一般来说 GFS2 是越小越好：十个 1TB 大小的文件系统要好过一个 10TB 大小的文件系统。

保持较小的 GFS2 文件系统有以下几个原因：

- 备份每个文件系统所需时间较短。
- 如果需要使用 **fsck.gfs2** 命令检查该文件系统，则所需时间较少。
- 如果需要使用 **fsck.gfs2** 命令检查该文件系统，则所需内存较少。

另外，需要维护的资源组越少，意味着性能更佳。

当然，如果 GFS2 文件系统太小，则可能会造成空间溢出，也会造成一定影响。请在决定文件系统大小前考虑自身使用情况。

2.1.2. 块大小：默认（4K）块是首选

从 Red Hat Enterprise Linux 6 开始，**mkfs.gfs2** 命令会尝试根据设备拓扑估算最佳块大小。通常，4K 块是首选块大小，因为 4K 是 Linux 的默认页大小（内存）。与其他文件系统不同，GFS2 使用 4K 内核缓存执行其大多数操作。如果您的块大小为 4K，则内核操作缓存的动作就少。

建议您使用可形成最高性能的默认块大小。

2.1.3. 日志数：每个挂载的节点一个日志

GFS2 要求集群中每个需要挂载该文件系统的节点都有一个日志。例如：如果您有一个有 16 个节点的集群，但只需要在其中的 2 个节点中挂载该文件系统，那么您就只需要两个日志。如果您需要挂载第三个节点，您总是可以使用**gfs2_jadd** 命令添加日志。在 GFS2 中您可以随时添加日志。

2.1.4. 日志大小：默认（128MB）通常是最佳选择

在运行 **mkfs.gfs2** 命令生成 GFS2 文件系统时要指定日志的大小。如果没有指定大小，则默认为 128MB，这对大多数应用程序都是最佳选择。

有些系统管理员可能认为 128MB 太大，并尝试将该日志大小减小到 8MB，或者保守地保留为 32MB。虽然可能正常工作，但仍会严重影响性能。与许多日志文件系统一样，每次 GFS2 写入元数据时，都会在元数据到位前提交到日志中。这样是为保证如果系统崩溃或者断电，则可恢复所有元数据，因为挂载时会自动使用日志替换。但如果使用 8MB 的日志，则无法记录太多的文件系统活动，同时当日志写满后，性能就会下降，因为 GFS2 必须等待写入存储。

一般推荐使用默认日志大小，即 128MB。如果文件系统太小（比如说 5GB），128MB 的日志就不太合适。如果文件系统较大，且有充分的空间，使用 256MB 日志可能会改进性能。

2.1.5. 资源组大小和数量

使用 **mkfs.gfs2** 命令创建 GFS2 文件系统时，它将存储分成统一的片段，即资源组。它尝试估算最佳资源组大小（范围在32MB 到 2GB 之间）。您可以使用 **mkfs.gfs2** 命令的 **-r** 选项覆盖默认值。

最佳资源组大小取决于如何使用该文件系统。请注意该文件会有多满，或者是否会被严重分割成碎片。

请尝试不同大小的资源组查看最佳性能。最好是在产品中部署 GFS2 前在测试集群中进行试验。

如果文件系统有太多资源组（每个都太小），则块分配会浪费太多时间搜索数以万计（或者十万计）的资源组才能找到空余的块。您的系统越满，则需要搜索的资源组越多，且它们都需要集群范围锁。这就会降低性能。

如果您的文件系统只有很少几个资源组（每个都很大），块分配可能会更频繁地访问同一资源组锁，这样也会影响性能。例如：如果您有一个 10GB 文件系统，分成 5 个 2GB 的资源组，您集群中的节点会比将同一文件系统分成 320 个资源组，每个 32MB 更频繁地访问那 5 个资源组。尤其是在文件系统快满的时候更为严重，因为每个块分配可能都必须在找到可用块之前查看几个资源组。GFS2 尝试从两个方面解决这个问题：

- 首先，当资源组全满时，它会记住并尝试避免在今后的分配中检查它（除非某个块是从该资源组中释放的）。如果从不删除文件，竞争会不那么严重。但如果应用程序不断在快满的文件系统中删除块并分配新块，竞争将会非常激烈，并严重影响性能。
- 其次，当在现有文件系统中添加新块时（例如：附加），GFS2 会尝试将同一资源组中的新块放在一起作为文件。这样做可提高性能：在旋转的磁盘中，如果它们放在一起则所需时间较少。

最糟糕的情况是在有中央目录时，则所有节点都在该目录中创建文件，因为所有节点将不断尝试锁定同一资源组。

2.2. 文件系统碎片

Red Hat Enterprise Linux 6.4 引进了 GFS2 中改进文件系统碎片管理的方法。在Red Hat Enterprise Linux 6.4 中，同时写入的结果是产生较少的文件碎片，并籍此获得更好的性能。

虽然 Red Hat Enterprise Linux 中没有用于 GFS2 的碎片清除工具，您可以通过文件碎片工具识别它们，将其复制到临时文件中，并重新命名该临时文件以替换原始文件，这样就可以清除碎片。（只要写入是按顺序进行的，这个步骤还可以用于 Red Hat Enterprise Linux 6.4 以前的版本。）

2.3. 块分配问题

本小节提供与在 GFS2 文件系统中进行块分配相关的问题概述。尽管那些只写入数据的应用程序通常不在乎如何或者在哪里分配块，但稍微了解一些块分配的只是可帮助您优化性能。

2.3.1. 在文件系统中保留空余空间

当 GFS2 文件系统接近写满时，块分配程序就很难为新要分配的块找到剩余空间。结果是分配程序放弃的块就会尝试挤进资源组的末端，或者挤入更像文件碎片的小片中。这个文件碎片就可能造成性能问题。另外，当 GFS2 快满时，GFS2 块分配程序会花更多的时间搜索多个资源组，并增加锁定竞争，这在有足够剩余空间的文件系统中是不必要的。这也会造成性能问题。

鉴于以上这些原因，我们建议您不要在 85% 已满的系统中运行文件系统，但这个限制值根据负载的不同而有所变化。

2.3.2. 尽可能让每个节点分配其自身文件

由于分布式锁定管理器（DLM）的工作方式，如果所有文件都是由一个节点分配，而其他节点需要向那些文件中添加块，就会造成更多的锁定竞争。

在 GFS（版本 1）中，所有锁定都是由中央锁定管理器进行管理，其任务是控制整个集群的锁定。这就造成统一锁定管理器（GULM）可能会出问题，因为这是一个单点失败。GFS2 的替换锁定方案 DLM 是在整个集群中分布锁定。如果集群中的任意节点失败，其锁定会由其他节点恢复。

使用 DLM，第一个锁定资源（比如文件）的节点成为该节点的“主锁定”。其他节点可以锁定那个资源，但它们必须首先向主锁定要求授予权限。每个节点都知道哪个锁定是哪个节点的主锁定，且每个节点都知道它为哪个节点发放了锁定授权。锁定主节点中的锁比锁定其他节点中的锁要快得多，因为后者必须停止并请求主锁定的授权。

因为在很多文件系统中，GFS2 分配程序会尝试将同一文件中块放在一起以减少磁头的移动，并极大提高性能。将块分配到文件的节点很可能需要为新的块使用并锁定同一资源组（除非那个资源组中所有的块都在使用中）。如果锁定包含将其数据分配到数据块的资源组，则系统将运行更迅速（即如果您让首先打开该文件的节点执行所有新块写入操作，则系统运行会更迅速）。

2.3.3. 尽可能预先分配

如果预先分配文件，就可以同时避免块分配，文件系统的运行也更有效。GFS2 较新的版本包含 **`fallocate`**(1) 系统调用，您可以使用这个命令预先分配数据块。

2.4. 集群注意事项

决定您系统中包含的节点数时，请注意在高可用性和性能之间要有所取舍。如果有大量的节点，则很难形成大规模负载。因此，Red Hat 不支持在超过 16 个节点的集群文件系统部署中使用 GFS2。

Deploying a cluster file system is not a "drop in" replacement for a single node deployment. We recommend that you allow a period of around 8-12 weeks of testing on new installations in order to test the system and ensure that it is working at the required performance level. During this period any performance or functional issues can be worked out and any queries should be directed to the Red Hat support team.

我们建议考虑部署集群的客户在部署前请Red Hat 支持团队审核其配置，这样可以避免之后可能存在的支持问题。

2.5. 用法注意事项

本小节提供关于 GFS2 常规推荐用法。

2.5.1. 挂载选项：**noatime** 和 **nodiratime**

通常建议使用 **noatime** 和 **nodiratime** 挂载 GFS2 文件系统。这让 GFS2 每次访问时花较少的时间更新磁盘节点。

2.5.2. DLM 调试选项：增加 DLM 表格大小

DLM 设备几个标签在集群的节点间管理、协调以及传递锁定信息。增大 DLM 标签的容量应该可以提高性能。在 Red Hat Enterprise Linux 6.1 以及之后的版本中，已增大这些标签的默认容量，但您可以使用以下命令手动增加这些标签的容量：

```
echo 1024 > /sys/kernel/config/dlm/cluster/lkbtbl_size
echo 1024 > /sys/kernel/config/dlm/cluster/rsbtbl_size
echo 1024 > /sys/kernel/config/dlm/cluster/dirtbl_size
```

这些命令将在重启后失效，因此您必须将其添加到一个启动脚本中，并在挂载任意 GFS2 文件系统前执行这些脚本，否则这些更改将被忽略，且没有任何提示。

有关 GFS2 节点锁定的详情请参考 第 2.9 节 “GFS2 节点锁定”。

2.5.3. VFS 调试选项：搜索和测试

与其他所有 Linux 系统类似，GFS2 位于顶层的虚拟文件系统（VFS）中。您可以调节 VFS 层以改进底层 GFS2 性能，方法是使用 **sysctl**(8) 命令。例如：可根据情况调整 **dirty_background_ratio** 和 **vfs_cache_pressure** 值。要使用当前值，则需要使用以下命令：

```
sysctl -n vm.dirty_background_ratio
sysctl -n vm.vfs_cache_pressure
```

以下命令可调整这些数值：

```
sysctl -w vm.dirty_background_ratio=20
sysctl -w vm.vfs_cache_pressure=500
```

您可以永久性更改这些参数值，方法是编辑 **/etc/sysctl.conf** 文件。

要找到您使用案例的最佳值，您需要搜索各种 VFS 选项，并在部署到产品中以前，在测试集群节点中进行测试。

2.5.4. SELinux：不要在 GFS2 中使用 SELinux

安全加强 Linux（SELinux）是为大多数情况推荐的安全选项，但不支持在 GFS2 中使用。SELinux 保存每个文件系统目标所使用的扩展属性的信息。GFS2 可以读取、写入并维护这些扩展属性，但也会极大降低其速度。因此必须在 GFS2 文件系统中关闭 SELinux。

2.5.5. 使用 GFS2 设定 NFS

由于 GFS2 锁定子系统额外的复杂性及其集群本质，使用 GFS2 设置 NFS 需要注意很多方面，并要格外小心。本小节论述了您在使用 GFS2 文件系统配置 NFS 时应该注意的问题。

警告

如果使用 GFS2 导出 NFS，且该 NFS 客户端程序使用 POSIX 锁，那么就必须使用 **localflocks** 选项挂载该文件系统。预期的效果是这样可以强制 POSIX 锁从每台服务器转到本地：即非集群，且各自独立。（如果 GFS2 尝试从 NFS 跨集群中的节点使用 POSIX 锁，目前还有很多问题需要解决。）对于在 NFS 客户端中运行的程序，本地的 POSIX 锁意味着如果是从两台服务器中挂载两个客户端，则它们可以同时持有同一锁。如果所有客户端都使用一台服务器挂载 NFS，那么就不存在不同服务器单独提供同一锁定的问题。如果您不确定是否要使用 **localflocks** 选项挂载您的文件系统，则不要使用该选项。在集群的环境中使用锁总是会更安全一些。

除锁定注意事项外，您还应在使用 GFS2 文件系统配置 NFS 服务时注意以下问题：

- Red Hat 只支持使用有以下特征并附带 active/passive 配置的 NFSv3 系统配置 Red Hat High Availability Add-On：
  - 后端文件系统是一个在 2 到 16 个节点集群中运行的 GFS2 文件系统。
  - 将 NFSv3 服务器被定义为一次从独立集群节点中导出整个 GFS2 文件系统的服务。
  - NFS 服务器可以在从一个集群节点到另一个节点（active/passive 配置）间进行故障切换。
  - *除非*通过 NFS 服务器，否则不允许任何对 GFS2 文件系统的访问。这包括本地 GFS2 文件系统访问以及所有通过 Samba 或者集群的 Samba 的访问。
  - 该系统中没有 NFS 额度支持。

  这个配置为该文件系统提供 HA，并减少系统停机时间，因为当 NFS 服务器从一个节点到另一个节点失败时，失败的节点不需要执行 **fsck** 命令。
- GFS2 的 NFS 导出中 **fsid=** NFS 选项是强制的。
- 如果您的集群出现问题（例如：该集群变得额度不足且 fencing 无法工作），则会停止集群的逻辑卷以及 GFS2 文件系统，且在该集群有足够额度前不可能有任何访问。您在决定是否使用简单的故障切换解决方案（比如：在这个步骤中规定的方法是否最适合您的系统）时考虑这个可能性。

2.5.6. 通过 GFS2 进行的 Samba（SMB 或者 Windows）文件服务

从 Red Hat Enterprise Linux 6.2 发行本开始，您可以在有 CTDB 的 GFS2 文件系统中使用 Samba（SMB 或者 Windows）文件服务，该文件系统应允许 active/active 配置。有关集群的 Samba 配置的详情请查看 *《集群管理》* 文档。

目前尚不支持同时访问 Samba 中与 Samba 以外共享的数据。目前不支持 GFS2 集群租赁，该服务可延迟 Samba 文件服务。

2.6. 文件系统备份

常规备份您的 GFS2 文件系统以防万一很重要，不要考虑文件系统的大小。很多系统管理员感到很安全是因为他们使用 RAID、multipath、镜像、快照以及其他形式的冗余，但永远没有足够安全这个说法。

生成备份可能会有问题，因为备份一个节点或者一组节点的过程通常包括按顺序读取整个文件系统。如果在单一节点中进行，则该节点将在缓存中保留所有信息直到集群中的其他节点开始请求锁定。在集群中运行此类备份程序是可对性能产生负面影响的操作。

备份完成后立即放弃缓存，这样可减少其他节点重新获得其集群锁/缓存所有权所需时间。但这仍不是最佳方法，因为其他节点将停止缓存备份进程开始前就已开始的缓存。您可以在备份完成后使用以下命令放弃缓存：

```
echo -n 3 > /proc/sys/vm/drop_caches
```

如果该集群在获得每个节点备份其各自拥有的文件就会更迅速，因为这样就将该任务分配到节点中进行。您还可以使用在没有具体节点的目录中使用 **rsync** 命令的脚本达到此目的。

备份 GFS2 的最佳方法是在 SAN 中创建硬件快照，将该快照放到另一个系统中，并在那里进行备份。该备份系统应使用 **-o lockproto=lock_nolock** 挂载该快照，因为它不在同一集群中。

2.7. 硬件注意事项

您应在部署 GFS2 文件系统时注意以下硬件注意事项。

- 使用高质量存储选项

  GFS2 可以在便宜的共享存储中运行，比如 iSCSI 或者 FCoE，但使用带较大缓存容量的较高质量的存储可获得更好的性能。Red Hat 在使用光纤连接的 SAN 存储中进行大多数质量、健全以及性能测试。基本原则是在部署某个产品前进行测试总是要好些。

- 部署前测试网络设备

  更高质量、更快速的网络设备可让集群沟通和 GFS2 运行更迅速，且更可靠。但您不一定要购买昂贵的硬件。有些昂贵的网络交换机有传送多播数据包的问题，这些数据包是用来传递 **fcntl** 锁（flocks），而较便宜的日用网络交换机有时更迅速且可靠。最好是在将是被部署到产品中时对其进行测试。

2.8. 性能问题：查看 RED HAT 客户门户网站

For information on best practices for deploying and upgrading Red Hat Enterprise Linux clusters using the High Availability Add-On and Red Hat Global File System 2 (GFS2) refer to the article "Red Hat Enterprise Linux Cluster, High Availability, and GFS Deployment Best Practices" on Red Hat Customer Portal at https://access.redhat.com/site/articles/40051.

2.9. GFS2 节点锁定

要获得最佳 GFS2 文件系统性能，则需要理解其操作的基本原理。单节点文件系统与缓存一同使用，其目的是在频繁使用请求的数据时可消除磁盘访问延迟。Linux 页面缓存（以及缓冲缓存）提供这个缓存功能。

使用 GFS2，每个节点都可有其自身的页面缓存，该缓存中包含 on-disk 数据的一部分。GFS2 使用 *glocks*（发音为 gee-locks）锁定机制维护节点间缓存的完整性。glock 子系统提供缓存管理功能，该功能使用*分布式锁管理器*（DLM）部署作为基础沟通层。

glocks 在每个内节点中为缓存提供保护，因此在每个内节点中都有一个锁定用来控制缓冲层。如果为那个 glock 赋予共享模式（DLM 锁定模式：PR），那么那个 glock 保护下的数据可同时被一个或者多个节点缓存，这样多有节点就都有到该数据的本地访问。

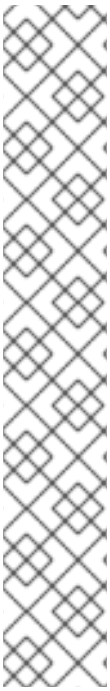
如果为 glock 赋予专用模式（DLM 锁定模式：EX），那么只有一个节点可缓存那个 glock 保护的数据。

所有修改数据的操作（例如 **write** 系统调用）都使用这个模式。

如果另一个节点请求 glock，但无法立刻获得，那么 DLM 会向该节点发送一条信息，或者向目前使用 glock 并妨碍新的请求的节点发送信息，要求它们释放其锁定。释放 glock（大多数文件系统操作标准）需要很长时间。释放共享 glock 只需要使该缓存无效，相对缓冲的数据来说速度较快。

释放专用 glock 需要 log flush，并向磁盘写回所有更改的数据，之后要使每个共享的 glock 失效。

单一节点文件系统与 GFS2 之间的区别在于单一节点文件系统只有一个缓存，而 GFS2 在每个节点中都有独立的缓存。在这两种情况下，对缓冲数据访问的延迟程度类似，但如果另一个节点之前缓冲了同样的数据，GFS2 对非缓冲数据访问的延迟要大得多。



注意

Due to the way in which GFS2's caching is implemented the best performance is obtained when either of the following takes place:

- 在所有节点中都使用只读方式使用内节点。
- 只在单一节点中写入或者修改内节点。

请注意：在创建和删除文件的过程中插入和删除目录条目也算在内，因为要写入到目录内节点。

也可能不遵守这个规则，但并不常发生。过度忽略这个规则会对性能有严重影响。

如果您在 GFS2 中使用 read/write 映射 **mmap()** 某个文件，但只读取它，那么这只计为读取。而在 GFS 中会将其计为写入，因此 GFS2 使用 **mmap()** I/O 时更灵活。

如果您没有设定 **noatime mount** 参数，那么读取也会导致写入来更新文件时间戳。文件建议所有 GFS2 用户应该使用 **noatime** 挂载，除非对 **atime** 有具体要求。

2.9.1. Posix 锁定问题

使用 Posix 锁定时要注意以下问题：

- 使用 Flokcs 将获得比使用 Posix 锁更迅速的处理。
- GFS2 中使用 Posix 锁的程序应避免使用 **GETLK** 功能，因为在集群环境中，进程 ID 可能是用于该集群的不同节点。

2.9.2. 使用 GFS2 调节性能

通常可以更改出问题的程序保存其数据的方法，这样可获得可观的性能优势。

典型的问题程序例子有电子邮件服务器。通常会制定包含每个用户的文件的 spool 目录（**mbox**），或者为每个用户创建一个目录，其中有包含所有信息的文件（**maildir**）。当有来自 IMAP 的请求时，理想的情况是为每个用户赋予一个对特定节点的亲和性。那么将会使用那个节点中的缓存处理他们查看和删除电子邮件信息的请求。显然，如果那个节点失败，那么可在不同的节点中重启该会话。

When mail arrives via SMTP, then again the individual nodes can be set up so as to pass a certain user's mail to a particular node by default. If the default node is not up, then the message can be saved directly into the user's mail spool by the receiving node. Again this design is intended to keep particular sets of files cached on just one node in the normal case, but to allow direct access in the case of node failure.

This setup allows the best use of GFS2's page cache and also makes failures transparent to the application, whether **imap** or **smtp**.

备份通常是另一个令人纠结的问题。如果可能，最好直接从节点备份每个节点的工作集合，这样可缓存具体的内节点组。如果您可以定期运行的备份脚本，且与在 GFS2 中运行的应用程序反应时间完全一致，那么很有可能集群无法最有效地使用页面缓存。

Obviously, if you are in the (enviable) position of being able to stop the application in order to perform a backup, then this won't be a problem. On the other hand, if a backup is run from just one node, then after it has completed a large portion of the file system will be cached on that node, with a performance penalty for subsequent accesses from other nodes. This can be mitigated to a certain extent by dropping the VFS page cache on the backup node after the backup has completed with following command:

```
echo -n 3 >/proc/sys/vm/drop_caches
```

但这并不是一个好的解决方案，最好的方案是保证在每个节点中的工作集合要么是共享的（大多数为集群内只读），要么是从单一节点的大量访问。

2.9.3. 使用 GFS2 锁定转储排除 GFS2 性能故障

If your cluster performance is suffering because of inefficient use of GFS2 caching, you may see large and increasing I/O wait times. You can make use of GFS2's lock dump information to determine the cause of the problem.

本小节提供 GFS 锁定转储概述。有关 GFS2 锁定转储的详情请参考 附录 C, *GFS2 跟踪点和 debug glock 文件*。

GFS2 锁定转储信息可从 **debugfs** 文件中获得，您可以根据路径名找到该文件，假设 **debugfs** 是挂载在 **/sys/kernel/debug/** 中：

```
/sys/kernel/debug/gfs2/fsname/glocks
```

文件的有多行组成，每个以 G: 开始的行代表一个 glock。接下来的行使用一个空格缩进，代表文件中最新与 glock 关联的信息项目。

当程序出现问题时，使用 **debugfs** 文件的最佳方法是使用 **cat** 命令获得该文件内容副本（如果您的 RAM 较大，且有很多缓冲的内节点，则需要较长时间），以后再查看得到的数据。

注意

复制两份 **debugfs** 文件会很有帮助，两个副本的间隔可在几秒甚至一分钟左右。比较与同一 glock 号关联的两个追踪信息，您可以了解负载是否在增长（就是说只是速度慢了）或者它在哪里卡住了（这通常是由 bug 造成，您应该立即向 Red Hat 支持提交报告）。

Lines in the **debugfs** file starting with H: (holders) represent lock requests either granted or waiting to be granted. The flags field on the holders line f: shows which: The 'W' flag refers to a waiting request, the 'H' flag refers to a granted request. The glocks which have large numbers of waiting requests are likely to be those which are experiencing particular contention.

表 2.1 “Glock 标签” 显示不同 glock 标签的含义，表 2.2 “Glock 拥有者标签” 按其在 glock 转储中出现的顺序显示不同 glock 拥有者标签的含义。

表 2.1. Glock 标签

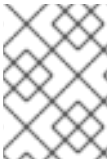
标签	名称	含义
b	阻断	Valid when the locked flag is set, and indicates that the operation that has been requested from the DLM may block. This flag is cleared for demotion operations and for "try" locks. The purpose of this flag is to allow gathering of stats of the DLM response time independent from the time taken by other nodes to demote locks.
d	等待降级	递延（远程）降级请求
D	降级	降级请求（本地或者远程）
f	清除日志	释放这个 glock 前需要提交该日志
F	冻结	忽略来自远程节点的回复 - 正在恢复。这个标签与文件系统停滞无关，它使用不同的机制，但只用于恢复。
i	使进程无效	这个 glock 下无效页面的进程中
l	启动	设定何时将 DLM 锁定与这个 glock 关联
l	锁定的	这个 glock 处于更改状态中
L	LRU	当 glock 在 LRU 列表中时设置
o	对象	glock 与某个对象关联时设定（即用于类型 2 glock 的内节点以及用于类型 3 glock 的资源组）
p	降级中	该 glock 正在响应降级请求
q	排队的	拥有者排队等待 glock 时设定，并在持有 glock 但没有拥有者时清除。是用于计算 glock 最小拥有时间的算法的一部分。
r	回复等待	从远程节点中接收的回复正在等待过程中
y	脏数据	释放这个 glock 前要刷新到磁盘中的数据

表 2.2. Glock 拥有者标签

标签	名称	含义
a	Async	不等待 glock 结果（以后轮询结果）
A	任意	接受所有兼容锁模式
c	没有缓存	取消锁定时立即降级 DLM 锁定

标签	名称	含义
e	没有过期日期	忽略之后的锁定取消请求
E	准确	必须有准确的锁定模式
F	第一	设定赋予这个锁定的第一个拥有者
H	拥有者	表示赋予请求的锁定
p	优先权	在队列头入队的拥有者
t	尝试	A "try" lock
T	Try 1CB	A "try" lock that sends a callback
W	等待	等待请求完成的设置

确定造成问题的 glock 后，下一步是要找到关联的内节点。glock 号（G: 行中的 n:）指的就是这个，其格式为 *type/number*，如果 *type* 是 2，那么 glock 就是一个内节点 glock，且 *number* 就是内节点号。要追踪内节点，您可以运行 **find -inum *number***，其中 *number* 是将 glock 文件中的十六进制格式转换为十进制格式的内节点号。



注意

如果您在有锁定冲突的文件系统中运行 **find**，事情可能会变得更糟糕。当您查找冲突的内节点时，最好在运行 **find** 前停止该程序。

表 2.3 “Glock 类型” 显示不同 glock 类型含义。

表 2.3. Glock 类型

类型号	1	使用
1	Trans	事务锁定
2	内节点	内节点元数据和数据
3	Rgrp	资源组元数据
4	Meta	超级块
5	Iopen	内节点的最近探测
6	Flock	**flock**(2) 系统调用
8	Quota	配额操作

类型号	1	使用
9	Journal	日志互斥

如果识别的 glock 是不同的类型，那么最可能是类型 3：（资源组）。如果您在正常负载情况看到大量进程正在等待其他 glock 类型，请向 Red Hat 支持提交报告。

如果您看到在资源组锁定中有大量等待的请求，那么可能有很多原因。其中之一是在文件系统中相对于资源组有大量的内节点。另一个原因就是该文件系统可能接近饱和（按平均计算，需要较长的搜索）。在这两种情况下可通过添加更多存储以及使用 **gfs2_grow** 命令扩大该文件系统进行改善。

第 3 章 开始

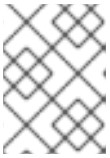
本章论述了初始设定 GFS2 的步骤，其中包括以下部分：

- 第 3.1 节 “前提任务”
- 第 3.2 节 “初始设定任务”

3.1. 前提任务

应在设置 Red Hat GFS2 前完成以下任务：

- 确定已记录 GFS2 节点的主要特点（请参考 第 1.2 节 “设置 GFS2 前的准备”）。
- 确定同步了 GFS2 节点中的时钟。建议使用 Red Hat Enterprise Linux 发行本中提供的网络时间协议（NTP）软件。



注意

GFS2 节点中的系统时钟间的差别必须在几分钟之内，这样可防止内节点时间戳更新。不必要的内节点时间戳更新会严重影响集群的性能。

- 要在集群的环境中使用 GFS2，就必须将系统配置为使用集群的逻辑卷管理器（CLVM），它是一组 LVM 逻辑卷管理器的集群扩展。要使用 CLVM，则必须运行包含 **clvmd** 守护进程的 Red Hat Cluster Suite 软件。有关使用 CLVM 的详情，请参考 *《管理逻辑卷管理器》*。有关安装和管理 Red Hat Cluster Suite 的详情请参考 *《集群管理》*。

3.2. 初始设定任务

初始 GFS2 设定包含以下任务：

1. 设定逻辑卷。
2. 生成 GFS2 文件系统。
3. 挂载文件系统。

开始按照以下步骤设定 GFS2。

1. 使用 LVM 为每个 Red Hat GFS2 文件系统生成逻辑卷。

注意

您可以使用 Red Hat Cluster Suite 中的 **init.d** 脚本自动激活和失活逻辑卷。有关 **init.d** 脚本的详情请参考 *《配置和管理 Red Hat 集群》*。

2. 在第一步生成的逻辑卷中创建 GFS2 文件系统。为每个文件系统选择唯一的名称。有关创建 GFS2 文件系统的详情请参考 第 4.1 节 “生成文件系统”。

   您可以使用以下格式之一创建一个集群 GFS2 文件系统：

```
mkfs.gfs2 -p lock_dlm -t ClusterName:FSName -j NumberJournals
BlockDevice
```

```
mkfs -t gfs2 -p lock_dlm -t LockTableName -j NumberJournals
BlockDevice
```

有关创建 GFS2 文件系统的详情请参考 第 4.1 节 “生成文件系统”。

3. 在每个节点中挂载 GFS2 文件系统。有关挂载 GFS2 文件系统的详情请参考 第 4.2 节 “挂载文件系统”。

   命令用法：

   mount *BlockDevice MountPoint*

   mount -o acl *BlockDevice MountPoint*

   -o acl 选项允许操作文件 ACL。如果挂载某个文件系统是没有使用 **-o acl** 挂载选项，用户可以查看 ACL（使用 **getfacl** 命令），但不可以设定它们（使用 **setfacl** 命令）。

注意

您可以使用 Red Hat High Availability Add-On（Red Hat 高可用性附加组件）中的 **init.d** 脚本自动挂载和卸载 GFS2 文件系统。

第 4 章 管理 GFS2

本章论述了管理 GFS2 的任务和命令，由以下小节组成：

4.1. 生成文件系统

您可使用 **mkfs.gfs2** 命令创建 GFS2 文件系统。您还可以使用指定了 **-t gfs2** 选项的 **mkfs** 命令。文件系统是在活跃的 LVM 卷中创建的。运行 **mkfs.gfs2** 命令时需要以下信息：

- 锁定协议/模块名称（集群的锁定协议为 **lock_dlm**）
- 集群名称（当作为集群配置的一部分运行时）
- 日志数目（每个可能挂载文件系统的节点都需要一个日志）

当创建 GFS2 文件系统时，您可以直接使用 **mkfs.gfs2**，或者使用带 ***-t*** 参数的 **mkfs** 命令，并使用 **gfs2** 文件系统选项将文件系统类型指定为 **gfs2**。

注意

当您使用 **mkfs.gfs2** 命令创建 GFS2 文件系统时，您不能缩小该文件系统。但您可以使用 **gfs2_grow** 命令增大现有文件系统的大小，如 第 4.6 节 “增大的文件系统” 所述。

用法

当创建集群的 GFS2 文件系统时，您可以使用以下任意格式之一：

```
mkfs.gfs2 -p LockProtoName -t LockTableName -j NumberJournals BlockDevice
-
```

```
mkfs -t gfs2 -p LockProtoName -t LockTableName -j NumberJournals
BlockDevice
```

您可以使用以下任意格式之一创建本地 GFS2 文件系统：

注意

在 Red Hat Enterprise Linux 6 发行本中，Red Hat 不支持将 GFS2 作为单节点文件系统使用。

```
mkfs.gfs2 -p LockProtoName -j NumberJournals BlockDevice
```

```
mkfs -t gfs2 -p LockProtoName -j NumberJournals BlockDevice
```

警告

请确定您非常熟悉 ***LockProtoName*** 和 ***LockTableName*** 参数的使用。不正确的 ***LockProtoName*** 和 ***LockTableName*** 参数使用可能导致文件系统或者锁定空间崩溃。

LockProtoName

指定要使用的锁定协议名称，集群的锁定协议为 **lock_dlm**。

LockTableName

这个参数是用来指定集群配置中的 GFS2 文件系统。它有两部分，用冒号隔开（没有空格）如下：***ClusterName:FSName***

- ***ClusterName***，用来创建 GFS2 文件系统的集群名称。
- ***FSName***，文件系统名称，长度可在 1-16 个字符之间。该名称必须与集群中所有 **lock_dlm** 文件系统以及每个本地节点中的所有文件系统（**lock_dlm** 和 **lock_nolock**）不同。

Number

指定由 **mkfs.gfs2** 命令生成的日志数目。每个要挂载文件系统的节点都需要一个日志。对于 GFS2 文件系统来说，以后可以添加更多的日志而不会增大文件系统，如 第 4.7 节 “在文件系统中添加日志” 所述。

BlockDevice

指定逻辑卷或者物理卷。

示例

在这些示例中，**lock_dlm** 是文件系统使用的锁定协议，因为这是一个集群的文件系统。集群名称为 **alpha**，文件系统名为**mydata1**。文件系统包含八个日志，日志是在 **/dev/vg01/lvol0** 中生成的。

```
mkfs.gfs2 -p lock_dlm -t alpha:mydata1 -j 8 /dev/vg01/lvol0
```

```
mkfs -t gfs2 -p lock_dlm -t alpha:mydata1 -j 8 /dev/vg01/lvol0
```

在这些示例中，生成了第二个 **lock_dlm** 文件系统，它可用于集群 **alpha**。文件系统名为**mydata2**。文件系统包含八个日志，日志是在 **/dev/vg01/lvol1** 中生成的。

```
mkfs.gfs2 -p lock_dlm -t alpha:mydata2 -j 8 /dev/vg01/lvol1
```

```
mkfs -t gfs2 -p lock_dlm -t alpha:mydata2 -j 8 /dev/vg01/lvol1
```

完整选项

表 4.1 “命令选项：**mkfs.gfs2**” 描述 **mkfs.gfs2** 命令选项（标签和参数）。

表 4.1. 命令选项：mkfs.gfs2

标签	参数	描述
-c	***Megabytes***	Sets the initial size of each journal's quota change file to ***Megabytes***.
-D		启用调试输出。
-h		帮助，显示可用选项。
-J	***MegaBytes***	以 MB 为单位指定日志大小。默认日志大小为 128MB，最小为 8MB。较大的日志可提高性能，但会比较小的日志占用更多的内存。
-j	***Number***	指定由 **mkfs.gfs2** 命令生成的日志数目。挂载文件系统的每个节点都需要一个日志。如果没有指定该选项，则会生成一个日志。对于 GFS2 文件系统，您可以后添加附加日志而不会增大文件系统。
-O		防止 **mkfs.gfs2** 命令在写入文件系统前进行确认。
-p	***LockProtoName***	指定要使用的锁定协议名称，可识别的锁定协议包括： **lock_dlm** – 标准锁定模式，用于集群的文件系统。 **lock_nolock** – 当 GFS2 作为本地文件系统作用时使用（只有一个节点）。
-q		静默，什么都不显示。

标签	参数	描述
-r	***MegaBytes***	以 MB 为单位指定源组大小，最小源组值为 32MB，最大源组值为 2048MB。在大型文件系统中源组越大性能越高。如果没有指定这个信息，mkfs.gfs2 会根据文件系统大小选择源组大小：中等大小的文件系统的源组为 256MB，大一点的文件系统会有较大的源组以获得更好的性能。
-t	***LockTableName***	在您使用 **lock_dlm** 协议时用来指定锁定表格字段的唯一识别程序，**lock_nolock** 协议不使用这个参数。 这个参数有两个部分，用冒号隔开（没有空格）如下：***ClusterName:FSName***。 ***ClusterName*** 是用来创建 GFS2 文件系统的集群名称，只有集群成员有使用此文件系统的权限。集群名称可使用 **Cluster Configuration Tool** 在文件 **/etc/cluster/cluster.conf** 中设定，并在 Red Hat Cluster Suite 集群管理 GUI 的 **Cluster Status Tool** 中显示。 ***FSName***，文件系统名称，长度可在 1-16 个字符之间，且必须不同于集群中的其他文件系统名。
-u	***MegaBytes***	Specifies the initial size of each journal's unlinked tag file.
-V		显示命令版本信息。

4.2. 挂载文件系统

在您挂载 GFS2 文件系统前，该文件系统必须存在（请参考 第 4.1 节 “生成文件系统”），该文件系统所属卷必须是被激活，且必须启动了集群和锁定系统支持（请参考*《配置和管理 Red Hat 集群》*）。达到这些要求后，您就可以将这个 GFS2 文件系统挂载到任意 Linux 文件系统中。

注意

在没有启动集群管理器（**cman**）时尝试挂载 GFS2 文件系统会产生以下出错信息：

```
[root@gfs-a24c-01 ~]# mount -t gfs2 -o noatime
/dev/mapper/mpathap1 /mnt
gfs_controld join connect error: Connection refused
error mounting lockproto lock_dlm
```

要控制文件 ACL，您必须使用 **-o acl** 挂载选项挂载文件系统。如果挂载文件系统时没有使用 **-o acl** 选项，用户可以查看 ACL（使用 **getfacl**），但不能对其进行设置（使用 **setfacl**）。

用法

不使用 ACL 控制挂载

```
mount BlockDevice MountPoint
```

使用 ACL 控制挂载

```
mount -o acl BlockDevice MountPoint
```

`-o acl`

允许控制文件 ACL 的具体 GFS2 选项。

`BlockDevice`

指定 GFS2 文件系统所在的块设备。

`MountPoint`

指定要挂载 GFS2 文件系统的目录。

示例

在这个示例中，位于 **`/dev/vg01/lvol0`** 的 GFS2 文件系统被挂载到 **`/mygfs2`** 目录中。

```
mount /dev/vg01/lvol0 /mygfs2
```

完整用法

```
mount BlockDevice MountPoint -o option
```

`-o option` 参数包含 GFS2 具体选项（请参考 表 4.2 “GFS2 特定挂载选项”）或者可接受的标准 Linux **`mount -o`** 选项，或者两者之和。多个 ***`option`*** 参数可使用逗号分开，没有空格。

注意

`mount` 命令是 Linux 系统命令。除了使用这部分论述的 GFS2 具体选项，您还可以使用其他标准 **`mount`** 命令选项（例如：**`-r`**）。有关其他 Linux **`mount`** 命令选项请参考 **`mount`** man page。

表 4.2 “GFS2 特定挂载选项” 描述在挂载时可传递给 GFS2 的 GFS2 特定 **`-o option`** 选项值。

注意

这个表格包含了只用于本地文件系统选项的描述。但请注意：在 Red Hat Enterprise Linux 6 发行本中Red Hat 不支持将 GFS2 作为单节点文件系统使用。Red Hat 将继续在挂载集群文件系统快照时支持单节点 GFS2 文件系统（例如：用于备份）。

表 4.2. GFS2 特定挂载选项

选项	描述
acl	允许控制文件 ACL。如果挂载文件系统时没有使用 **acl** 挂载选项，那么用户可以查看 ACL（使用 **getfacl**），但不能对其进行设置（使用 **setfacl**）。
data=[ordered|writeback]	当设置 **data=ordered** 时，事务中修改的用户数据会将该事物递交到磁盘前被冲入磁盘。这样可以让用户无法在崩溃后的文件中看到未初始化的块。设定 **data=writeback** 时，用户数据会在磁盘有数据后的任何时候被写入磁盘。这样无法提供 **ordered** 模式可提供的一致性保障，但可稍微提高一些工作负载的速度。默认设置为 **ordered** 模式。
ignore_local_fs **警告：**在共享 GFS2 文件系统时*不*应该使用这个选项。	强制 GFS2 将文件系统视为多主机文件系统。默认情况是使用 **lock_nolock** 自动打开 **localflocks** 标签。
localflocks **警告：**在共享 GFS2 文件系统时*不*应该使用这个选项。	告知 GFS2 让 VFS（虚拟文件系统）层完成所有 flock 和 fcntl 操作。**lock_nolock** 可自动打开 **localflocks** 标签。
lockproto=*LockModuleName*	允许用户指定文件系统要使用的锁定协议。如果没有指定 ***LockModuleName***，则会从文件系统的超级块中读取锁定协议。
locktable=*LockTableName*	允许用户指定文件系统要使用的锁定表。
quota=[off/account/on]	为文件系统打开或者关闭配额。在 **account** 状态中配置配额可让文件系统正确维护对每个 UID/GID 使用统计，忽略限制和警告值。默认值为 **off**。
errors=panic|withdraw	当指定 **errors=panic** 时，文件系统错误将导致内核 panic。默认的行为与指定 **errors=withdraw** 一致，即将该系统从文件系统中退出，且直到重启前都无法访问。在有些情况下，该系统仍保持运行。有关 GFS2 退出功能请参考 第 4.14 节 “GFS2 收回功能”。
discard/nodiscard	Causes GFS2 to generate "discard" I/O requests for blocks that have been freed. These can be used by suitable hardware to implement thin provisioning and similar schemes.

选项	描述
barrier/nobarrier	当清洗日志时 GFS2 会发送 I/O 屏障。默认值为**on**。如果基础设备不支持 I/O 屏障则会自动将这个选项改为 **off**。强烈建议您随时在 GFS2 中使用 I/O 屏障，除非块设备设计为不使用，这样它就不会丢失其写入缓存内容（例如：如果它在 UPS 中或者没有写入缓存）。
quota_quantum=*secs*	在将更改的配额信息写入配额文件前将其保存在某个节点的秒数。这是设定此参数的首选方法。该数值是一个大于 0 的整数。默认为 60 秒。设定为较短的间隔会让配额信息更快地更新，且更不可能让某些人超过其配额。较长的间隔可让文件系统操作更迅速有效地包括配额。
statfs_quantum=*secs*	设置 **statfs** 慢速版本的首选方法是将 **statfs_quantum** 设定为 0。默认值为 30 秒，该值设定了将 **statfs** 与主 **statfs** 文件同步前的最大时间段。可将该值调整为更迅速但不准确的 **statfs** 值，也可将其设定为更慢但更准确的值。当将该选项设定为 0 时，**statfs** 将总是报告真实值。
statfs_percent=*value*	提供在没有超时时，将 **statfs** 信息与主 **statfs** 文件同步前该信息更改的最大比例值。如果将 **statfs_quantum** 设定为 0，那么会忽略这个设置。

4.3. 卸载文件系统

可使用与卸载 Linux 文件系统相同的方法卸载 GFS2 文件系统 – 即使用 **umount** 命令。

注意

umount 是 Linux 系统命令。有关此命令的详情请参考 Linux **umount** 命令 man page。

用法

```
umount MountPoint
```

MountPoint

指定当前挂载 GFS2 文件系统的目录。

4.4. 挂载 GFS2 文件系统时的具体注意事项

该系统不会了解那些在系统关闭的过程中卸载的，通过手动挂载而不是使用 **fstab** 文件中的条目自动挂载的 GFS2 文件系统。因此，GFS2 脚本不会卸载 GFS2 文件系统。在运行 GFS2 关闭脚本后，标准关闭进程会杀死所有保留的用户进程，包括集群基础结构，并尝试卸载该文件系统。没有集群基础构架这个卸载会失败，且该系统会停滞。

要防止卸载 GFS2 文件系统时的系统停滞，您应该进行以下操作之一：

- 总是使用 **fstab** 文件中的条目挂载 GFS2 文件系统。
- 如果已经手动使用 **mount** 命令挂载了 GFS2 文件系统，请确定在重启或者关闭该系统前手动使用 **umount** 命令卸载该文件系统。

如果在这些情况下关闭系统的过程中，卸载文件系统时该文件系统停滞，请执行硬件重启。这样到不会丢失数据，因为在关闭进程的初期已经同步了该文件系统。

4.5. GFS2 配额管理

文件系统配额是用来限制某个用户或者组使用的文件系统空间。在设置前对用户或者组没有配额限制。当使用 **quota=on** 或者 **quota=account** 选项时，GFS2 会不断跟踪每个用户或者组使用的空间，即使没有设定限制也是如此。GFS2 以互动的方式更新配额信息，因此系统崩溃并不需要重建配额用量。

为防止性能下降，GFS2 节点只会定时为配额文件更新同步。fuzzy 配额核算可允许用户或者组稍微超过设定的限制。为最小化这种情况，GFS2 会在接近 hard 配额限制时动态缩短同步周期。

注意

从 Red Hat Enterprise Linux 6.1 发行本开始，GFS2 支持标准 Linux 配额工具。要使用这个工具，您需要安装 **quota** RPM。这是在 GFS2 中管理配额的首选方法，且应该在所有使用配额新部署的 GFS2 中使用。本小节记录了如何使用这些工具管理 GFS2 配额。

在 Red Hat Enterprise Linux 之前的发行本中，GFS2 文件系统中的 **gfs2_quota** 命令管理配额。有关 **gfs2_quota** 命令的详情请参考 附录 A, *使用 **gfs2_quota** 命令执行 GFS2 配额管理*。

4.5.1. 配置磁盘配额

请使用以下步骤实施磁盘配额：

1. 设置配额的强制或者计数模式。
2. 使用当前块使用信息初始化配额数据库文件。
3. 分配配额策略。（在计数模式中不强制这些策略。）

在以下小节中会详细讨论这些步骤的具体内容。

4.5.1.1. 将配额设定为强制或者计数模式

在 GFS2 文件系统中，默认禁用配额。要为文件系统启用配额，请在挂载文件系统时指定 **quota=on** 选项。

有可能在没有强制限制和警告值的情况下为每个用户和组跟踪磁盘用量并维护配额核算。要做到这一点，请使用 **quota=account** 选项挂载文件系统。

用法

要挂载启用配额的文件系统，请在挂载文件系统时使用 **quota=on** 选项。

```
mount -o quota=on BlockDevice MountPoint
```

要在挂载文件系统时即使没有强制配额限制也要使用配额计数维护，则请在挂载文件系统时指定 **quota=account** 选项。

```
mount -o quota=account BlockDevice MountPoint
```

要在挂载文件系统时禁用配额，请使用 **quota=off** 选项挂载文件系统。这是默认设置。

```
mount -o quota=off BlockDevice MountPoint
```

quota={on|off|account}

on - 指定挂载文件系统时启用配额。

off - 指定挂载文件系统时禁用配额。

account - 即使在没有强制配额限制的情况下，也指定根据文件系统维护用户和组的用量统计。

BlockDevice

指定 GFS2 文件系统所在的块设备。

MountPoint

指定要挂载 GFS2 文件系统的目录。

示例

在这个示例中，**/dev/vg01/lvol0** 中的 GFS2 文件系统被挂载到 **/mygfs2** 目录中并启用了配额。

```
mount -o quota=on /dev/vg01/lvol0 /mygfs2
```

在这个示例中，**/dev/vg01/lvol0** 中的 GFS2 文件系统被挂载到 **/mygfs2** 目录中并启用了配额计数，但没有强制。

```
mount -o quota=account /dev/vg01/lvol0 /mygfs2
```

4.5.1.2. 创建配额数据库文件

挂载了每个启用了配额的文件系统后，该系统就可以使用磁盘配额。但是该系统本身还不支持配额。下一步就是要运行 **quotacheck** 命令。

The **quotacheck** command examines quota-enabled file systems and builds a table of the current disk usage per file system. The table is then used to update the operating system's copy of disk usage. In addition, the file system's disk quota files are updated.

要在该文件系统中创建配额文件，请使用 **quotacheck** 命令的 **-u** 和 **-g** 选项。必须为用户和组指定这两个选项方可进行初始化。如果为 **/home** 文件系统启用配额，则请在 **/home** 目录中生成该文件：

```
quotacheck -ug /home
```

4.5.1.3. 为每个用户分配配额

最后一步是使用 **edquota** 命令分配磁盘配额。请注意：如果您使用计数模式挂载文件系统（即指定

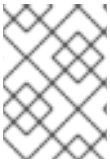
quota=account 选项），则不强制使用配额。

请在 shell 提示符后成为 root 用户，并执行以下命令为用户配置配额：

```
edquota username
```

为每个需要配额的用户执行这个步骤。例如：如果在 **/etc/fstab** 中为 **/home** 分区（在下面的示例中为 **/dev/VolGroup00/LogVol02**）启用配额，并执行 **edquota testuser** 命令，则在系统默认的编辑器中会显示以下内容：

```
Disk quotas for user testuser (uid 501):
Filesystem                blocks     soft     hard    inodes   soft
hard
/dev/VolGroup00/LogVol02  440436        0        0
```



注意

edquota 使用由 **EDITOR** 环境变量定义的文本编辑器。要更改编辑器，请在 **~/.bash_profile** 文件中将 **EDITOR** 环境变量设定为到您选择的编辑器的完整路径。

第一列是启用了配额的文件系统的名称。第二列显示目前该用户使用的块数。后两列是为该用户在该文件系统中设定的软限制和硬限制。

软块限制定义可使用的最大磁盘空间量。

硬块限制是用户或者组可以使用的绝对最大磁盘空间量。达到这个上限后就再没有可以使用的磁盘空间了。

GFS2 文件系统不为内节点维护配额，因此这些列不适用于 GFS2 文件系统，为空白。

如果有任何值为 0，就是没有设定那个限制。您可以使用文本编辑器更改所需限制。例如：

```
Disk quotas for user testuser (uid 501):
Filesystem                blocks     soft     hard    inodes   soft
hard
/dev/VolGroup00/LogVol02  440436   500000   550000
```

请使用这个命令确认是否为该用户设定了配额：

```
quota testuser
```

4.5.1.4. 为每个组分配配额

还可以根据组分配配额。请注意：如果您使用计数模式挂载您的文件系统（指定 **account=on** 选项），则不强制配额。

请使用以下命令为 **devel** 组设定组配额（设定组配额前就存在该组）：

```
edquota -g devel
```

这个命令在文本编辑器中显示该组的现有配额：

```
Disk quotas for group devel (gid 505):
Filesystem                blocks    soft     hard    inodes   soft   hard
/dev/VolGroup00/LogVol02  440400       0        0
```

GFS2 文件系统不为内节点维护配额，因此这些列不适用于 GFS2 文件系统，为空白。您可以修改这些限制，然后保存文件。

请使用以下命令确认是否设定了组配额：

```
quota -g devel
```

4.5.2. 管理磁盘配额

如果使用配额，则需要对其进行维护 – 大多数是查看是否超过了配额，并确定配额是准确的。

Of course, if users repeatedly exceed their quotas or consistently reach their soft limits, a system administrator has a few choices to make depending on what type of users they are and how much disk space impacts their work. The administrator can either help the user determine how to use less disk space or increase the user's disk quota.

您可以运行 **repquota** 程序创建磁盘用量报告。例如：命令 **repquota /home** 可有这样的输出：

```
*** Report for user quotas on device /dev/mapper/VolGroup00-LogVol02
Block grace time: 7days; Inode grace time: 7days
   Block limits   File limits
User  used soft hard grace used soft hard grace
----------------------------------------------------------------------
root      --      36       0       0              4     0     0
kristin   --     540       0       0            125     0     0
testuser  --  440400  500000  550000          37418     0     0
```

请使用以下命令查看所有启用了配额的文件系统的磁盘用量报告（选项 **-a**）：

```
repquota -a
```

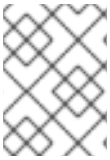
虽然这个报告很好理解，但有几点需要注意。在每个用户后面显示的 **--** 可让您迅速确定是否超过块限制。如果超过块软限制，在输出结果的第一个 **-** 位置会出现 **+**。第二个 **-** 表示内节点限制，但 GFS2 文件系统不支持内节点限制，因此那个字符会保留为 **-**。GFS2 文件系统不支持宽限期，因此 **grace** 一列将为空白。

备注：不考虑基础文件系统，NFS 都不支持 **repquota** 命令。

4.5.3. 保持配额准确

如果您已经禁用配额运行文件系统一段时间并要再次启用配额，您应该运行 **quotacheck** 命令创建、检查并修复配额文件。另外，如果您认为配额文件不准确，您可以运行 **quotacheck** 命令，因为有可能在系统崩溃时没有完全卸载文件系统。

有关 **quotacheck** 命令的详情请查看 **quotacheck** man page。



注意

当文件系统在所有节点中相对停滞的时候运行 **quotacheck**，因为磁盘活性会影响计算的配额值。

4.5.4. 使用 quotasync 命令同步配额

GFS2 在其磁盘自身内部文件中保存所有配额信息。GFS2 节点不需要在每次文件系统写入时更新这个配额文件，它会每 60 秒更新一次这个配额文件。这在避免节点间在写入配额文件时发生冲突是很有必要的，这种冲突会降低节点性能。

当用户或组接近其配额限制时，GFS2 动态降低配额文件更新的时间间隔以防止超过限额。配额同步时间间隔通常是一个可调整的参数 **quota_quantum**。可使用 **quota_quantum=** 挂载选项更改其默认的 60 秒，如 表 4.2 “GFS2 特定挂载选项” 所述。必须在每个节点及每次挂载文件系统时都设置 **quota_quantum** 参数。卸载时不会保留 **quota_quantum** 参数。可使用 **mount -o remount** 更新 **quota_quantum** 值。

您可以使用 **quotasync** 命令在 GFS2 执行自动更新时将某个节点的配额信息与磁盘配额文件同步。

用法

同步配额信息

```
quotasync [-ug] -a|mntpnt...
```

u

同步该用户的配额文件。

g

同步该组的配额文件

a

同步所有目前启用配额并支持同步的文件系统。缺少 **-a** 时，应是定文件系统挂载点。

mntpnt

指定要执行动作的 GFS2 文件系统。

调整同步时间间隔

```
mount -o quota_quantum=secs,remount BlockDevice MountPoint
```

MountPoint

指定要执行动作的 GFS2 文件系统。

secs

指定 GFS2 对常规配额文件进行同步化的新时间周期。数值越小，竞争越激烈，同时还会降低性能。

示例

这个示例与为文件系统 **/mnt/mygfs2** 在磁盘配额文件中运行的节点同步所有缓冲的脏配额。

```
# quotasync -ug /mnt/mygfs2
```

这个示例是在将该文件系统重新挂载到逻辑卷 **/dev/volgroup/logical_volume** 时，为文件系统 **/mnt/mygfs2** 将常规配额文件更新的时间间隔默认值改为 1 小时（即 3600 秒）。

```
# mount -o quota_quantum=3600,remount /dev/volgroup/logical_volume
/mnt/mygfs2
```

4.5.5. 参考

有关磁盘配额的详情请参考以下命令的 **man page**：

- **quotacheck**
- **edquota**
- **repquota**
- **quota**

4.6. 增大的文件系统

gfs2_grow 是在其文件系统所在设备被扩展后，用来扩展 GFS2 文件系统的命令。在现有 GFS2 文件系统中运行 **gfs2_grow** 命令，则会填满目前文件系统终点和新初始化的 GFS2 文件系统扩展设备终点之间的所有剩余空间。当完成填充工作后，会为文件系统更新源索引。集群中的所有节点则可以使用以添加的额外存储空间。

gfs2_grow 必须在挂载的文件系统中运行，但只需要在集群的一个节点中运行。其他节点可感觉到扩展的发生，并可自动使用新的空间。

注意

您使用 **mkfs.gfs2** 命令创建 GFS2 文件系统后，您就无法缩小该文件系统的大小。

用法

```
gfs2_grow MountPoint
```

MountPoint

指定要执行动作的 GFS2 文件系统。

注释

在运行 **gfs2_grow** 命令前请您：

- 备份文件系统中的重要数据。
- 运行 **df *MountPoint*** 命令确定要进行扩展的文件系统的容量。

- 有关生成 LVM 逻辑卷的详情请参考 *《管理逻辑卷管理器》*。

运行 **gfs2_grow** 命令后，请运行 **df** 命令查看文件系统中新的可用空间。

示例

在这个示例中扩展了 **/mygfs2fs** 目录中的文件系统。

```
[root@dash-01 ~]# gfs2_grow /mygfs2fs
FS: Mount Point: /mygfs2fs
FS: Device:      /dev/mapper/gfs2testvg-gfs2testlv
FS: Size:        524288 (0x80000)
FS: RG size:     65533 (0xfffd)
DEV: Size:       655360 (0xa0000)
The file system grew by 512MB.
gfs2_grow complete.
```

完整用法

```
gfs2_grow [Options] {MountPoint | Device} [MountPoint | Device]
```

MountPoint

指定要挂载 GFS2 文件系统的目录。

Device

指定文件系统的设备节点。

表 4.3 “扩展文件系统是可用的 GFS2 具体选项” 描述在扩展 GFS2 文件系统时所要使用的 GFS2 具体选项。

表 4.3. 扩展文件系统是可用的 GFS2 具体选项

选项	描述
-h	帮助，显示简短用法信息。
-q	静默，降低详细等级。
-r MegaBytes	指定新资源组大小，默认值为 256MB。
-T	测试。完成所有计算，但不要向磁盘中写入数据，也不要扩展文件系统。
-V	显示命令版本信息。

4.7. 在文件系统中添加日志

gfs2_jadd 命令可用来在 GFS2 文件系统中添加日志。您可以在任意点动态在 GFS2 文件系统中添加日志，且不需要扩展基础逻辑卷。**gfs2_jadd** 必须在挂载的文件系统中运行，但只需要在集群的一个节点中运行。其他节点可感觉到扩展的发生。

注意

如果某个 GFS2 文件系统已满，则 **gfs2_jadd** 将会失败，即使将逻辑卷包含的文件系统扩展到超过该文件系统的大小也是如此。这是因为在 GFS2 文件系统中，日志是纯文本文件，而不是嵌入的元数据，因此只是增大基础逻辑卷大小不会为日志提供空间。

在向 GFS 文件系统中添加日志前，您可以使用 **gfs2_tool** 命令的 **journals** 选项找出 GFS2 文件系统目前含有多少日志。以下示例显示挂载在 **/mnt/gfs2** 的文件系统中的日志数目和大小。

```
[root@roth-01 ../cluster/gfs2]# gfs2_tool journals /mnt/gfs2
journal2 - 128MB
journal1 - 128MB
journal0 - 128MB
3 journal(s) found.
```

用法

```
gfs2_jadd -j Number MountPoint
```

Number

指定要添加的新日志数目。

MountPoint

指定要挂载 GFS2 文件系统的目录。

示例

这个示例中，在 **/mygfs2** 目录的文件系统中添加了一个日志。

```
gfs2_jadd -j1 /mygfs2
```

这个示例中，在 **/mygfs2** 目录的文件系统中添加了两个日志。

```
gfs2_jadd -j2 /mygfs2
```

完整用法

```
gfs2_jadd [Options] {MountPoint | Device} [MountPoint | Device]
```

MountPoint

指定要挂载 GFS2 文件系统的目录。

Device

指定文件系统的设备节点。

表 4.4 “添加日志时可用的 GFS2 具体选项” 描述用来在 GFS2 文件系统中添加日志的 GFS2 具体选项。

表 4.4. 添加日志时可用的 GFS2 具体选项

标签	参数	描述
-h		帮助，显示简短用法信息。
-J	***MegaBytes***	以 MB 为单位指定新日志的大小。默认日志大小为 128MB，最小值为 32MB。要在文件系统添加不同大小的日志，必须为每个不同大小的日志运行 **gfs2_jadd** 命令。指定的大小会不断下降，因此在生成文件系统时会指定多个日志区段。
-j	***Number***	用 **gfs2_jadd** 命令指定要添加的新日志数目，默认值为 1。
-q		静默，降低详细等级。
-V		显示命令版本信息。

4.8. 数据日志

Ordinarily, GFS2 writes only metadata to its journal. File contents are subsequently written to disk by the kernel's periodic sync that flushes file system buffers. An **fsync()** call on a file causes the file's data to be written to disk immediately. The call returns when the disk reports that all data is safely written.

数据日志可缩短非常小的文件的 **fsync()** 时间，因为文件数据在写入元数据外还要写入日志。随着文件的增大这个优势会明显降低。写入中等到大文件时打开数据日志会非常慢。

依赖 **fsync()** 同步文件数据的应用程序可能因使用数据日志而使性能有所提高。在被标记的目录及其所有子目录中生成的 GFS2 文件可自动启用数据日志。现有长度为 0 的文件也可以打开或者关闭其数据日志功能。

Enabling data journaling on a directory sets the directory to "inherit jdata", which indicates that all files and directories subsequently created in that directory are journaled. You can enable and disable data journaling on a file with the **chattr** command.

下面的命令在 **/mnt/gfs2/gfs2_dir/newfile** 文件中启用了数据日志，并查看是否正确设定了标签。

```
[root@roth-01 ~]# chattr +j /mnt/gfs2/gfs2_dir/newfile
[root@roth-01 ~]# lsattr /mnt/gfs2/gfs2_dir
---------j--- /mnt/gfs2/gfs2_dir/newfile
```

以下命令禁用了 **/mnt/gfs2/gfs2_dir/newfile** 文件中的数据日志，并查看是否正确设定了标签。

```
[root@roth-01 ~]# chattr -j /mnt/gfs2/gfs2_dir/newfile
[root@roth-01 ~]# lsattr /mnt/gfs2/gfs2_dir
------------- /mnt/gfs2/gfs2_dir/newfile
```

您还可以使用 **chattr** 命令在目录中设定 **j** 标签。当您为某个目录设定此标签时，以后在那个目录中创建的所有文件和目录也都会进行日志操作。下面的一组命令可在 **gfs2_dir** 目录中设定 **j** 标签，然后查看是否正确设定了该标签。此后，该命令会在 **/mnt/gfs2/gfs2_dir** 目录中生成一个名为 **newfile** 新

文件，然后查看是否将为该文件设定了 **j** 标签。因为为该目录设定了 **j** 标签，那么应该也为 **newfile** 启用了日志操作。

```
[root@roth-01 ~]# chattr -j /mnt/gfs2/gfs2_dir
[root@roth-01 ~]# lsattr /mnt/gfs2
---------j--- /mnt/gfs2/gfs2_dir
[root@roth-01 ~]# touch /mnt/gfs2/gfs2_dir/newfile
[root@roth-01 ~]# lsattr /mnt/gfs2/gfs2_dir
---------j--- /mnt/gfs2/gfs2_dir/newfile
```

4.9. 配置 ATIME 更新

每个文件内节点和目录内节点都有三个与之相关的时间戳：

- **ctime** – 最后一次修改内节点状态的时间
- **mtime** – 最后一次修改文件（或者目录）数据的时间
- **atime** – 最后一次访问文件（或者目录）数据的时间

如果启用了 **atime** 更新，因为在 GFS2 和其他 Linux 文件系统中这是默认设置，那么每次在读取文件时都需要更新其内节点。

因为有些应用程序使用 **atime** 提供的信息，那些更新可能会需要大量不必要的写入流量和文件锁定流量。那个流量可能会降低性能，因此请尽量关闭或降低 **atime** 更新频率。

有两个可用来降低 **atime** 更新效果的方法：

- 使用 **relatime**（relative atime）挂载，可在前一个 **atime** 更新比 **mtime** 或者 **ctime** 更新旧时更新 **atime**。
- 使用 **noatime** 挂载，可在那个文件系统中禁用 **atime** 更新。

4.9.1. 使用 relatime 挂载

当挂载文件系统时，可指定 **relatime**（relative atime）Linux 挂载选项。如果前一个 **atime** 更新比 **mtime** 或者 **ctime** 更新旧，这就可指定更新 **atime**。

用法

```
mount  BlockDevice MountPoint -o relatime
```

BlockDevice

指定 GFS2 文件系统所在的块设备。

MountPoint

指定要挂载 GFS2 文件系统的目录。

示例

在这个示例中，GFS2 文件系统位于 **/dev/vg01/lvol0**，并挂载到目录 **/mygfs2**。只有在前一个 **atime** 更新比 **mtime** 或者 **ctime** 更新旧时才会进行 **atime** 更新。

```
mount /dev/vg01/lvol0 /mygfs2 -o relatime
```

4.9.2. 使用 noatime 挂载

当挂载文件系统时，可指定 Linux 挂载选项 **noatime**，它可在那个文件系统中禁用 **atime** 更新。

用法

```
mount BlockDevice MountPoint -o noatime
```

BlockDevice

指定 GFS2 文件系统所在的块设备。

MountPoint

指定要挂载 GFS2 文件系统的目录。

示例

在这个示例中，GFS2 文件系统位于 **/dev/vg01/lvol0**，并挂载到关闭了 **atime** 更新的 **/mygfs2** 目录中。

```
mount /dev/vg01/lvol0 /mygfs2 -o noatime
```

4.10. 在文件系统中挂起一个动作

您可以使用 **gfs2_tool freeze** 命令挂起对某个文件系统的写入动作。挂起写入动作允许使用基于硬件的设备快照捕获处于一致状态的文件系统。**dmsetup resume** 命令可终止挂起。

用法

Start Suspension

```
dmsetup suspend MountPoint
```

End Suspension

```
dmsetup resume MountPoint
```

MountPoint

指定文件系统。

示例

这个示例挂起了对文件系统 **/mygfs2** 的写入动作。

```
# dmsetup suspend /mygfs2
```

这个示例终止了对文件系统 **/mygfs2** 写入动作的挂起。

```
# dmsetup resume /mygfs2
```

4.11. 修复文件系统

当挂载文件系统节点失败时，文件系统日志允许快速恢复。但如果存储设备断电或者断开物理连接，则会发生文件系统崩溃。（无法使用日志进行存储子系统失败修复。）当这种崩溃发生时，您可以使用 **fsck.gfs2** 命令修复 GFS2 文件系统。

重要

fsck.gfs2 命令必须只能在从所有节点中卸载的文件系统中运行。

重要

不应在引导时使用 **fsck.gfs2** 命令检查 GFS2。**fsck.gfs2** 命令不能在引导时确定是否在集群的另一个节点中挂载该文件系统。应只在系统引导后手动运行 **fsck.gfs2** 命令。

To ensure that the **fsck.gfs2** command does not run on a GFS2 file system at boot time, modify the **/etc/fstab** file so that the final two columns for a GFS2 file system mount point show "0 0" rather than "1 1" (or any other numbers), as in the following example:

```
/dev/VG12/lv_svr_home   /svr_home       gfs2
defaults,noatime,nodiratime,noquota     0 0
```

注意

如果您以前有在 GFS 文件系统中使用 gfs_fsck 命令的经验，请注意 **fsck.gfs2** 命令在以下方面和之前发布的 **gfs_fsck** 有所不同：

- 在运行 **fsck.gfs2** 时按 **Ctrl+C** 会中断进程并显示提示信息，询问您是否要取消该命令，跳过剩余操作或者继续该进程。
- 您可以使用 **-v** 标签提高详细等级。添加第二个 **-v** 标签会再次提高等级。
- 您可以使用 **-q** 标签降低详细等级。添加第二个 **-q** 标签会再次降低等级。
- **-n** 会以只读方式打开某个文件系统并自动对所有查询回答 **no**。该选项提供了在不允许 **fsck.gfs2** 命令生效的前提下使用命令找出错误的方法。

有关其他命令选项详情请参考 **fsck.gf2** man page。

运行 **fsck.gfs2** 命令要求系统内存高于操作系统使用的内存而低于内核使用的内存。GFS2 文件系统中的每块内存本身需要大约 5 比特额外内存，或者 5/8 字节。因此要估算在您的文件系统中运行 **fsck.gfs2** 命令所需内存字节数，您可以确定包含的文件系统块的数量，然后乘以 5/8。

例如：要确定在大小为 16TB，块大小为 4K 的 GFS2 文件系统中运行 **fsck.gfs2** 命令大约需要多少内存，首先要确定该文件系统中包含多少内存块，可使用 16TB 除以 4K：

```
 17592186044416 / 4096 = 4294967296
```

这个文件系统中包含 **4294967296** 个块，再乘以 **5/8** 就可确定需要多少内存：

```
4294967296 * 5/8 = 2684354560
```

该文件系统大约需要 2.6GB 可用内存方可运行 **fsck.gfs2** 命令。请注意：如果块大小为 1K，那么运行 **fsck.gfs2** 命令将需要四倍于以上值的内存，即大约 11GB。

用法

```
fsck.gfs2 -y BlockDevice
```

-y

-y 标签可使对所有问题的回答都为 **yes**。如果指定 **-y** 标签，**fsck.gfs2** 命令则不会在进行修改前提示您回答问题。

BlockDevice

指定 GFS2 文件系统所在的块设备。

示例

在这个示例中，修复了位于块设备 **/dev/testvol/testlv** 中的 GFS2 文件系统。所有关于修复查询的回答都自动为 **yes**。

```
[root@dash-01 ~]# fsck.gfs2 -y /dev/testvg/testlv
Initializing fsck
Validating Resource Group index.
Level 1 RG check.
(level 1 passed)
Clearing journals (this may take a while)...
Journals cleared.
Starting pass1
Pass1 complete
Starting pass1b
Pass1b complete
Starting pass1c
Pass1c complete
Starting pass2
Pass2 complete
Starting pass3
Pass3 complete
Starting pass4
Pass4 complete
Starting pass5
Pass5 complete
Writing changes to disk
fsck.gfs2 complete
```

4.12. 绑定挂载以及上下文关联路径名

GFS2 文件系统不提供对上下文关联路径名（CDPN）的支持，CDPN 允许您生成指向不同目的地文件或者目录的符号链接。在 GFS2 中使用 **mount** 命令的 **bind** 选项实现此功能。

mount 命令的 **bind** 选项允许您在不同位置重新挂载部分文件结构，且使其在初始位置仍可用。该命令的格式如下：

```
mount --bind olddir newdir
```

执行此命令后，***olddir*** 目录中的内容在两个位置可用：***olddir*** 和 ***newdir***。您还可以使用这个选项生成在两个位置可用的独立文件。

例如：在执行以下命令后，**/root/tmp** 中的内容将和之前挂载的 **/var/log** 目录内容一致。

```
[root@menscryfa ~]# cd ~root
[root@menscryfa ~]# mkdir ./tmp
[root@menscryfa ~]# mount --bind /var/log /root/tmp
```

另外，您可以使用 **/etc/fstab** 文件中的条目在挂载时得到同样的结果。**/etc/fstab** 中的以下条目可使 **/root/tmp** 的内容和 **/var/log** 目录中的内容一致。

```
/var/log                /root/tmp               none    bind
0 0
```

在您挂载文件系统后，您可以使用 **mount** 命令查看该文件系统是否被挂载了，如下示例所示：

```
[root@menscryfa ~]# mount | grep /tmp
/var/log on /root/tmp type none (rw,bind)
```

对于支持上下文关联路径名的文件系统，您可以将 **/bin** 目录定义为上下文关联路径名，并根据系统构架将其解析为以下路径之一：

```
/usr/i386-bin
/usr/x86_64-bin
/usr/ppc64-bin
```

您可以通过生成空 **/bin** 目录得到同样的结果。然后使用脚本或者在 **/etc/fstab** 文件中的条目，将每个独立构架目录使用 **mount -bind** 命令挂载到 **/bin** 目录。例如：您可以使用以下命令作为脚本中的一行：

```
mount --bind /usr/i386-bin /bin
```

另外，您还可以使用以下行作为 **/etc/fstab** 文件的条目：

```
/usr/1386-bin             /bin                none    bind            0 0
```

绑定挂载可为您提供比上下文关联路径名更大的灵活性，因为您可以使用此特性根据您定义的条件挂载不同的目录（比如文件系统的 **%fill** 值）。上下文关联路径名对其可处理的环境有更多的限制。请注意：您将需要根据条件（比如 **%fill**）编写您自己的挂载脚本。

警告

当您使用 **bind** 选项挂载文件系统，且起始文件系统以 **rw** 挂载时，新的文件系统也会被以 **rw** 形式挂载，即使您使用的是 **ro** 标签，**ro** 则被静默忽略了。在这种情况下，可能会在 **/proc/mounts** 目录中将新的文件系统标记为 **ro** 而引起误导。

4.13. 绑定挂载和文件系统挂载顺序

当您使用 **mount** 命令的 **bind** 选项时，您必须确定使用正确的顺序挂载该文件系统。在以下示例中，必须在 **/tmp** 目录中执行绑定挂载前挂载 **/var/log** 目录：

```
# mount --bind /var/log /tmp
```

按以下方法决定文件系统挂载顺序：

- 通常文件系统挂载顺序由 **fstab** 文件中文件系统出现的顺序决定。例外情况包括使用 **_netdev** 标签挂载的文件系统或者有自身**初始化**脚本的文件系统。
- 有自身**初始化**脚本的文件系统在初始化进程后期挂载，即在挂载 **fstab** 文件中的文件系统之后挂载。
- 使用 **_netdev** 标签挂载的文件系统会在该系统中启用网络时挂载。

如果您的配置需要创建绑定挂载以便挂载 GFS2 文件系统，您可以命令 **fstab** 文件进行如下操作：

1. 挂载绑定挂载所需本地文件系统。
2. 绑定挂载要挂载 GFS2 文件系统的目录。
3. 挂载 GFS2 文件系统。

如果您的配置需要您在 GFS2 文件系统中绑定挂载本地目录或者文件系统，在 **fstab** 文件中列出正确的文件系统顺序也不会正确挂载文件系统，因为在 GFS2 **初始化**脚本运行前不会挂载该 GFS2 文件系统。在这种情况下，您应该在**初始化**脚本中写入命令执行绑定挂载，这样在挂载 GFS2 文件系统前就不会发生绑定挂载。

以下脚本是自定义**初始化**脚本示例。这个脚本在 GFS2 文件系统的两个目录中执行绑定挂载。在这个示例中，在 **/mnt/gfs2a** 有一个 GFS2 挂载点，可在集群启动并运行**初始化**脚本时执行挂载。

在这个示例脚本中 **chkconfig** 状态值说明：

- 345 说明启动该脚本的运行等级
- 29 是启动优先性，在这个示例中表示该脚本会在启动时在 GFS2 **初始化**脚本之后运行，后者的启动优先性为 26
- 73 为停止优先性，在这个示例中表示该脚本会在关闭时在 GFS2 脚本之后停止，后者的停止优先性为 74

启动和停止值表示您可以通过执行 **service start** 和 **service stop** 命令手动执行所示动作。例如：如果该脚本名为 **fredwilma**，则您可以执行 **service fredwilma start**。

应将这个脚本放到 **/etc/init.d** 目录并拥有与该目录中其他脚本相同的权限。您可以执行 **chkconfig on** 命令将该脚本链接到所示的运行等级中。例如：如果该脚本名为 **fredwilma**，那么您可以执行 **chkconfig fredwilma on**。

```
#!/bin/bash
#
# chkconfig: 345 29 73
# description: mount/unmount my custom bind mounts onto a gfs2
subdirectory
#
#
### BEGIN INIT INFO
# Provides:
### END INIT INFO

. /etc/init.d/functions
case "$1" in
  start)
 # In this example, fred and wilma want their home directories
 # bind-mounted over the gfs2 directory /mnt/gfs2a, which has
 # been mounted as /mnt/gfs2a
 mkdir -p /mnt/gfs2a/home/fred &> /dev/null
 mkdir -p /mnt/gfs2a/home/wilma &> /dev/null
 /bin/mount --bind /mnt/gfs2a/home/fred /home/fred
 /bin/mount --bind /mnt/gfs2a/home/wilma /home/wilma
        ;;

  stop)
 /bin/umount /mnt/gfs2a/home/fred
 /bin/umount /mnt/gfs2a/home/wilma
        ;;

  status)
        ;;

  restart)
        $0 stop
        $0 start
        ;;

  reload)
        $0 start
        ;;
  *)
        echo $"Usage: $0 {start|stop|restart|reload|status}"
        exit 1
esac

exit 0
```

4.14. GFS2 收回功能

GFS2 *收回*功能是集群 GFS2 文件系统的数据完整功能。如果 GFS2 内核模块探测到 GFS2 文件系统中存在不一致性，并伴随 I/O 操作，则该文件系统对该集群来说就不可用。该 I/O 操作会停止，同时该系统会等待下一个出错的 I/O 操作，防止进一步的破坏。当出现这种情况时，您可以停止手动停止任意服务或者程序，然后重启并重新挂载 GFS2 文件系统以便重新执行日志操作。如果问题仍存在，您可以在集群的所有节点中卸载该文件系统并使用 **fsck.gfs2** 命令执行文件系统恢复。GFS2 收回功能没有内核 panic 那么严重，但可造成使用另一个节点 fence 这个节点。

如果您使用启用了启动脚本的 **gfs2** 配置您的系统，且在 **/etc/fstab** 文件中包含 GFS2 文件系统，则会在重启时重新挂载该 GFS2 文件系统。如果因为文件系统崩溃造成 GFS2 文件系统收回，则建议您在重新挂载该文件系统前运行 **fsck.gfs2** 命令。在这种情况下，要防止您的文件系统在引导时重新挂载，您可以执行以下步骤：

1. 使用以下命令暂时在受影响的节点中禁用启动脚本：

   ```
   # chkconfig gfs2 off
   ```

2. 重启受影响的节点，启动集群软件。此时将不会挂载该 GFS2 文件系统。

3. 在集群的所有节点中卸载该文件系统。

4. 只在确定没有文件系统崩溃的一个节点的文件系统中运行 **fsck.gfs2**。

5. 运行以下命令在受影响的节点中重新启用该启动脚本：

   ```
   # chkconfig gfs2 on
   ```

6. 在集群的所有节点中重新挂载 GFS2 文件系统。

可能造成 GFS2 收回的不一致性示例为错误的块计数。当 GFS 内核删除文件系统中的某个文件上，它会系统地删除与那个文件关联的所有数据和元数据。完成此操作后，它会检查块计数。如果该块计数不是 1（意思是只剩下磁盘内节点自己），则表明文件系统不一致，因为块计数与找到的块列表不匹配。

You can override the GFS2 withdraw function by mounting the file system with the **-o errors=panic** option specified. When this option is specified, any errors that would normally cause the system to withdraw cause the system to panic instead. This stops the node's cluster communications, which causes the node to be fenced.

在内部，GFS2 通过让内核向 **gfs_controld** 守护进程发送请求撤回的信息启动撤回功能。**gfs_controld** 会运行 **dmsetup** 程序替换该文件系统中的设备映射器错误目标以防止进一步访问块设备。然后它会告诉内核操作完成。这是 GFS2 支持要求总是在 GFS2 使用 CLVM 设备的理由，否则就不可能插入设备映射器目标。

设备映射器错误目标的目的是确保将来所有的 I/O 操作都将有一个 I/O 出错信息会让该文件系统以较旧的方式卸载。结果是当出现撤回时，通常会在系统日志中看到来自设备映射器的 I/O 出错信息。

偶尔在 **dmsetup** 程序不可能根据要求插入错误目标则撤回会失败。如果撤回时缺少内存，且由于造成撤回的问题而无法重新使用内存会发生这种情况。

撤回并不总是意味着 GFS2 中有错误。有时撤回功能可由与基础块设备有关的设备 I/O 错误引发。强烈建议您检查日志来查看发生撤回是否是这样的原因。

第 5 章 诊断并修正 GFS2 文件系统的问题

本章提供常见 GFS2 问题以及如何处理这些问题的信息。

5.1. GFS2 文件系统出现性能缓慢

您会发现 GFS2 文件系统显示出比 ext3 文件系统缓慢的性能。在某些情况下 GFS2 的性能可能受到一些因素的影响。本文档中有关于处理 GFS2 性能问题的信息。

5.2. GFS2 文件系统挂起并需要在一个节点中重启

如果您的 GFS2 文件系统挂起，且不返回由此运行的命令，但重启一个返回常态的具体节点，这可能表示有锁定问题或者 bug。出现这种情况时您应该收集以下数据：

- 在每个节点中为该文件系统执行的 gfs2 锁定转储：

  ```
  cat /sys/kernel/debug/gfs2/fsname/glocks >glocks.fsname.nodename
  ```

- 每个节点中为该文件系统执行的 DLM 锁定转储：您可以使用以下 **dlm_tool** 命令收集这个信息：

  ```
  dlm_tool lockdebug -sv lsname.
  ```

  在这个命令中，*lsname* 是有问题的附加系统 DLM 使用的锁定空间名称。您可以在 **group_tool** 命令的输出结果中找到这个值。

- **sysrq -t** 命令的输出结果。

- **/var/log/messages** 文件的内容。

您收集到数据后，可以在 Red Hat 支持生成一个 ticket，并提供您收集的数据。

5.3. GFS2 文件系统挂起并需要重启所有节点

如果您的 GFS2 文件系统挂起且不会返回由此运行的命令，则需要您在使用它之前重启该集群中的所有节点，并检查以下问题。

- 您可能会有一个失败的 fence。GFS2 文件系统将停滞以保证在失败的 fence 事件中数据的完整性。检查信息日志查看挂起时是否有失败的 fence。请确定您正确了配置的 fencing。

- GFS2 文件系统可能已经撤回。检查信息日志查看关键字 **withdraw**，看看是否有来自 GFS2 表示已经撤回的文件系统的信息和 calltrace。撤回表示可能为系统崩溃、存储失败或者是一个 bug。卸载该文件系统，更新 **gfs2-utils** 软件包并在该文件系统中执行 **fsck** 命令以便返回该服务。在 Red Hat 支持生成一个支持 ticket。通知它们您有 GFS2 撤回问题，并提供有日志的 sosreport。

  有关 GFS2 撤回功能的详情请参考 第 4.14 节 “GFS2 收回功能”。

- 这个出错信息表示有锁定问题或者 bug。如果出现这种情况之一，请收集数据并在 Red Hat 支持生成一个支持 ticket，如 第 5.2 节 “GFS2 文件系统挂起并需要在一个节点中重启” 所述。

5.4. GFS2 文件系统不挂载新添加的集群节点

如果在集群中添加一个新节点，且发现无法在那个节点中挂载 GFS2 文件系统，那么相较尝试访问 GFS2 文件系统的节点，您的 GFS2 文件系统中的日志可能较少。必须在每个要挂载文件系统的 GFS2 主机中都有一个日志（使用 **spectator** 挂载选项组挂载的 GFS2 文件系统除外）。可以使用 **gfs2_jadd** 命令在 GFS2 文件系统中添加日志，如 第 4.7 节 “在文件系统中添加日志” 所述。

5.5. 空格代表在空文件系统中使用

如果您有一个空 GFS2 文件系统，**df** 命令将显示使用的空间。这是因为 GFS2 文件系统日志消耗磁盘空间（日志数乘以日志大小）。如果创建有大量日志的 GFS2 文件系统或者指定大日志，那么将在执行 **df** 时看到空间已经被使用（日志数乘以日志大小）。即使没有指定较大的日志数或者大日志，小 GFS2 文件系统（1GB 或者更小）将使用大量默认 GFS2 日志大小的空间。

第 6 章 在 PACEMAKER 集群中配置 GFS2 文件系统

以下为设置使用 GFS2 文件系统的 Pacemaker 集群所需步骤。

在每个节点中安装集群软件、GFS2 及集群的软件包后，在每个节点中启动 **cman**、**clvmd** 和 **pacemaker** 服务并创建 Pacemaker 集群。必须为该集群配置 fencing。有关配置 Pacemaker 集群的详情请查看 *《使用 Pacemaker 配置 Red Hat High Availability Add-On》*。

1. 将全局 Pacemaker 参数 **no_quorum_policy** 设定为 **freeze**。

注意

默认情况下会将 **no-quorum-policy** 值设定为 **stop**，说明丢失一个仲裁，同时将停止剩余分区中的所有资源。通常这个选项是最安全同时也是最佳选项，但与大多数资源不同，GFS2 需要仲裁功能。丢失仲裁后，使用 GFS2 挂载的应用程序及 GFS2 挂载本身都将正常停止。所有在无仲裁的情况下停止执行资源的尝试都将失败，并最终造成每次丢失仲裁时都 fence 整个集群。

为解决这个问题，您可以在使用 GFS2 时设置 **no-quorum-policy=freeze**。这意味着丢失仲裁时，重新获得仲裁钱剩余分区不会做任何操作。

```
# pcs property set no-quorum-policy=freeze
```

2. 确定在 **/etc/lvm/lvm.conf** 文件中将锁定类型设定为 3 以支持集群的锁定后，创建集群的 LV，并使用 GFS2 文件系统格式化该卷。确定为集群中的每个节点生成足够的日志。

```
# pvcreate /dev/vdb
# vgcreate -Ay -cy cluster_vg /dev/vdb
# lvcreate -L5G -n cluster_lv cluster_vg
# mkfs.gfs2 -j2 -p lock_dlm -t rhel7-demo:gfs2-demo
/dev/cluster_vg/cluster_lv
```

3. 配置 **clusterfs** 资源。

   您不应在 **/etc/fstab** 文件中软件文件系统，因为会将其作为 Pacemaker 集群资源管理。可将挂载选择指定为使用 **options=*options*** 配置的资源的一部分。有关所有配置选项详情请运行 **pcs resource describe Filesystem** 命令。

   这个创建集群资源的命令将指定 **noatime** 挂载选项。

```
# pcs resource create clusterfs Filesystem
device="/dev/cluster_vg/cluster_lv" directory="/var/mountpoint"
fstype="gfs2" "options=noatime" op monitor interval=10s on-
fail=fence clone interleave=true
```

4. 确认如预期挂载 GFS2。

```
# mount |grep /mnt/gfs2-demo
/dev/mapper/cluster_vg-cluster_lv on /mnt/gfs2-demo type gfs2
(rw,noatime,seclabel)
```

5. （自选）重启所有集群节点确认 gfs2 持续性及恢复。

附录 A. 使用 GFS2_QUOTA 命令执行 GFS2 配额管理

从 Red Hat Enterprise Linux 601 发行本开始，GFS2 支持标准 Linux 配额设备。要使用这个功能，您需要安装 **quota** RPM。这是在 GFS2 中管理配额的首选方法，并应在所有使用配额的 GFS2 新部署中使用。有关使用标准 Linux 配额设备的详情请参考 第 4.5 节 “GFS2 配额管理”。

在 Red Hat Enterprise Linux 之前的版本中，GFS2 需要 **gfs2_quota** 命令来管理配额。这个附录记录了使用 **gfs2_quota** 命令进行 GFS2 文件系统配额管理的内容。

A.1. 使用 GFS2_QUOTA 命令设定配额

每个用户 ID（UID）或者组 ID（GID）都有两个配额设置：*硬限制*和*软限制*。

硬限制是可以使用的空间数量。该文件系统不会让用户或者组使用超过该数量的磁盘空间。硬限制值为*零*表示没有限制。

软限制通常是一个小于硬限制的值。文件系统会在达到软限制时通知用户或组，警告他们正在使用的空间量。软限制值为 *0* 表示没有强制限制。

您可以使用 **gfs2_quota** 命令设定限制。只需要在 GFS2 挂载的单一节点中运行此命令即可。

默认情况下不会在 GFS2 文件系统设定配额强制。要启用配额计数，请在挂在 GFS2 文件系统时使用 **mount** 命令的 **quota=** 选项，如 第 A.4 节 “启用/禁用配额强制” 所述。

用法

Setting Quotas, Hard Limit

```
gfs2_quota limit -u User -l Size -f MountPoint
```

```
gfs2_quota limit -g Group -l Size -f MountPoint
```

Setting Quotas, Warn Limit

```
gfs2_quota warn -u User -l Size -f MountPoint
```

```
gfs2_quota warn -g Group -l Size -f MountPoint
```

User

限制或者警告的用户 ID。可以是密码文件中的用户名，也可以是 UID 号。

Group

限制或者警告的组 ID。可以是组文件中的组名称或者 GID 号。

Size

指定要限制或者警告的新值。默认情况下该值以 MB 为单位。使用 **-k**、**-s** 和 **-b** 标签可分别将单位改为 kb，扇区和文件系统块。

MountPoint

指定要执行动作的 GFS2 文件系统。

示例

这个示例在文件系统 **/mygfs** 中将用户 *Bert* 的硬限制设定为 1024MB（1GB）。

```
# gfs2_quota limit -u Bert -l 1024 -f /mygfs2
```

这个示例在文件系统 **/mygfs** 中将组 ID 21 的软限制设定为 **50kb**。

```
# gfs2_quota warn -g 21 -l 50 -k -f /mygfs2
```

A.2. 使用 GFS2_QUOTA 命令显示配额限制和用量

可使用 **gfs2_quota get** 命令为具体用户或者组显示配额限制和当前用量。还可使用 **gfs2_quota list** 命令显示配额文件的整个内容，其中所有 ID 都是非 0 硬限制或者列出的值。

用法

Displaying Quota Limits for a User

```
gfs2_quota get -u User -f MountPoint
```

Displaying Quota Limits for a Group

```
gfs2_quota get -g Group -f MountPoint
```

Displaying Entire Quota File

```
gfs2_quota list -f MountPoint
```

User

显示具体用户信息的用户 ID。可以使用密码文件中的用户名或者 UID 号。

Group

显示具体组信息的组 ID。可以是组文件中的组名称或者 GID 号。

MountPoint

指定要执行动作的 GFS2 文件系统。

命令输出

gfs2_quota 命令显示 GFS2 配额信息如下：

```
user User: limit:LimitSize warn:WarnSize value:Value

group Group: limit:LimitSize warn:WarnSize value:Value
```

LimitSize、***WarnSize*** 和 ***Value*** 数（值）默认使用 MB 为单位。在命令行中使用 **-k**、**-s** 或者 **-b** 标签可分别将单位更改为 kb、扇区或者文件系统块。

User

与该数据关联的用户名或者 ID。

Group

与该数据关联的组名称或者 ID。

LimitSize

用户或者组的硬限制。如果没有设定限制则该值为 0。

Value

用户或者组实际使用的磁盘空间量。

注释

显示配额信息时，如果在 **gfs2_quota** 中添加 **-n** 选项，则该命令不会将 UID 或者 GID 解析为名称。

Space allocated to GFS2's hidden files can be left out of displayed values for the root UID and GID by adding the **-d** option to the command line. This is useful when trying to match the numbers from **gfs2_quota** with the results of a **du** command.

示例

这个示例显示所有设定了限制或者使用文件系统 **/mygfs2** 所有空间的用户和组的配额信息。

```
# gfs2_quota list -f /mygfs2
```

这个示例为文件系统 **/mygfs2** 的组 **users** 显示扇区中的配额信息。

```
# gfs2_quota get -g users -f /mygfs2 -s
```

A.3. 使用 GFS2_QUOTA 命令同步配额

GFS2 在其磁盘的内部文件中保存所有配额信息。GFS2 节点不会在每次写入系统文件时更新这个配额文件，默认情况下它每 60 秒更新一次配额文件。这样可避免节点在写入配额文件时出现竞争，这种竞争可导致性能下降。

当用户或组接近其配额限制时，GFS2 动态降低配额文件更新的时间间隔以防止超过限额。配额同步时间间隔通常是一个可调整的参数 **quota_quantum**。可使用 **quota_quantum=** 挂载选项更改其默认的 60 秒，如 表 4.2 “GFS2 特定挂载选项” 所述。必须在每个节点及每次挂载文件系统时都设置 **quota_quantum** 参数。卸载时不会保留 **quota_quantum** 参数。可使用 **mount -o remount** 更新 **quota_quantum** 值。

您可以使用 **gfs2_quota sync** 命令会在 GFS2 执行自动更新时将某个节点中的配额信息与磁盘中的配额文件同步。

用法

Synchronizing Quota Information

```
gfs2_quota sync -f MountPoint
```

MountPoint

指定要执行动作的 GFS2 文件系统。

Tuning the Time Between Synchronizations

```
mount -o quota_quantum=secs,remount BlockDevice MountPoint
```

MountPoint

指定要执行动作的 GFS2 文件系统。

secs

指定 GFS2 常规配额文件同步的新时间间隔。较小的值可增加竞争并降低性能。

示例

这个示例是与在文件系统 **/mygfs2** 中运行的节点同步配额信息。

```
# gfs2_quota sync -f /mygfs2
```

这个示例是在将该文件系统重新挂载到逻辑卷 **/dev/volgroup/logical_volume** 时，为文件系统 **/mnt/mygfs2** 将常规配额文件更新的时间间隔默认值改为 1 小时（即 3600 秒）。

```
# mount -o quota_quantum=3600,remount /dev/volgroup/logical_volume
/mnt/mygfs2
```

A.4. 启用/禁用配额强制

在 GFS2 文件系统中默认为禁用配额强制。要为文件系统启用配额强制，您可以指定 **quota=on** 选项挂载文件系统。

用法

```
mount -o quota=on BlockDevice MountPoint
```

要在挂载文件系统时禁用配额强制，请指定 **quota=off** 选项挂载文件系统。这是默认设置。

```
mount -o quota=off BlockDevice MountPoint
```

-o quota={on|off}

挂载文件系统时指定是启用还是禁用配额强制。

BlockDevice

指定 GFS2 文件系统所在的块设备。

MountPoint

指定应挂载 GFS2 文件系统的目录。

示例

在这个示例中，将 **/dev/vg01/lvol0** 中的 GFS2 文件系统挂载到 **/mygfs2** 目录并启用配额强制。

```
# mount -o quota=on /dev/vg01/lvol0 /mygfs2
```

A.5. 启用配额计数

可在不强制的情况下为每个用户和组追踪磁盘用量并维护配额计数。要做到这一点，需要在挂载文件系统时指定 **quota=account** 选项。

用法

```
mount -o quota=account BlockDevice MountPoint
```

-o quota=account

指定由该文件系统维护的用户和组用量统计，即使没有强制配额限制。

BlockDevice

指定 GFS2 文件系统所在的块设备。

MountPoint

指定应挂载 GFS2 文件系统的目录。

示例

在这个示例中将 **/dev/vg01/lvol0** 中的 GFS2 文件系统挂载到 **/mygfs2** 目录并启用配额计数。

```
# mount -o quota=account /dev/vg01/lvol0 /mygfs2
```

附录 B. 将文件系统从 GFS 转换为 GFS2

由于 Red Hat Enterprise Linux 6 不支持 GFS 文件系统，因此您必须使用 **gfs2_convert** 命令将现有 GFS 文件系统升级到 GFS2 文件系统。请注意：您必须在升级到Red Hat Enterprise Linux 6 之前在Red Hat Enterprise Linux 5 中执行这个转换过程。

警告

在转换 GFS 文件系统前，您必须备份您的文件系统，因为转换过程是不可逆的，且在转换过程中出现的错误可导致进程意外终结，从而使文件系统不可用。

在转换 GFS 文件系统前，您必须使用 **gfs_fsck** 命令检查文件系统并修复所有错误。

如果由于停电或者其它问题导致 GFS 到 GFS2 的转换中断，请重启转换工具。在转换完成前不要在文件系统中执行 **fsck.gfs2** 命令。

当您转换全部或者接近全部文件系统时，可能没有足够的空间放置 GFS2 文件系统数据结构。在这种情况下，所有日志都统一缩小为适应可用空间的大小。

B.1. 上下文关联路径名转换

GFS2 文件系统不提供对上下文关联路径名（CDPN）的支持，CDPN 允许您生成指向不同目的文件或者目录的符号链接。您可使用 **mount** 命令的 **bind** 选项获得 CDPN 在 GFS2 文件系统中的相同功能。

gfs2_convert 命令识别 CDPN，并使用有相同名称的空目录替换它们。要配置绑定挂载替换 CDPN，则需要您了解要替换的 CDPN 目标的完整路径。转换您的文件系统前，您可以使用 **find** 命令识别该链接。

以下命令列出指向 **hostname** CDPN 的符号链接：

```
[root@smoke-01 gfs]# find /mnt/gfs -lname @hostname
/mnt/gfs/log
```

同样，您可以为其它 CDPN（**mach**、**os**、**sys**、**uid**、**gid**、**jid**）执行 **find** 命令。请注意：因为 CDPN 名称的格式可以是 **@hostname** 或者 **{hostname}**，所以您需要为每个变体运行 **find** 命令。

有关 GFS2 中绑定挂载和文本独立路径名的详情请参考 第 4.12 节 “绑定挂载以及上下文关联路径名”。

B.2. GFS 到 GFS2 转换步骤

采用以下步骤将 GFS 文件系统转换为 GFS2 文件系统。

1. 在 Red Hat Enterprise Linux 系统中，请备份您的现有 GFS 文件系统。
2. 将 GFS 文件系统从集群中的所有节点中卸载。
3. 在 GFS 文件系统中执行 **gfs_fsck** 命令以确定没有文件系统崩溃。

4. 执行 **gfs2_convert*gfsfilesystem***。该系统会显示警告信息并在将 ***gfsfilesystem*** 转换为 GFS2 之前进行确认。

5. 升级到 Red Hat Enterprise Linux 6。

以下示例是在块设备 **/dev/shell_vg/500g** 中将 GFS 文件系统转换为 GFS2 文件系统。

```
[root@shell-01 ~]#  /root/cluster/gfs2/convert/gfs2_convert
/dev/shell_vg/500g
gfs2_convert version 2 (built May 10 2010 10:05:40)
Copyright (C) Red Hat, Inc.  2004-2006  All rights reserved.

Examining file system...................
This program will convert a gfs1 filesystem to a gfs2 filesystem.
WARNING: This can't be undone.  It is strongly advised that you:

   1. Back up your entire filesystem first.
   2. Run gfs_fsck first to ensure filesystem integrity.
   3. Make sure the filesystem is NOT mounted from any node.
   4. Make sure you have the latest software versions.
Convert /dev/shell_vg/500g from GFS1 to GFS2? (y/n)y
Converting resource groups...................
Converting inodes.
24208 inodes from 1862 rgs converted.
Fixing file and directory information.
18 cdpn symlinks moved to empty directories.
Converting journals.
Converting journal space to rg space.
Writing journal #1...done.
Writing journal #2...done.
Writing journal #3...done.
Writing journal #4...done.
Building GFS2 file system structures.
Removing obsolete GFS1 file system structures.
Committing changes to disk.
/dev/shell_vg/500g: filesystem converted successfully to gfs2.
```

附录 C. GFS2 跟踪点和 DEBUG GLOCK 文件

本附录论述了 glock **debugfs** 界面和 GFS2 跟踪点。主要面向熟悉文件系统内部，并想要了解更多 GFS2 设计及如何 debug GFS2 具体问题的高级用户。

C.1. GFS2 跟踪点类型

There are currently three types of GFS2 tracepoints: *glock* (pronounced "gee-lock") tracepoints, *bmap* tracepoints and *log* tracepoints. These can be used to monitor a running GFS2 file system and give additional information to that which can be obtained with the debugging options supported in previous releases of Red Hat Enterprise Linux. Tracepoints are particularly useful when a problem, such as a hang or performance issue, is reproducible and thus the tracepoint output can be obtained during the problematic operation. In GFS2, glocks are the primary cache control mechanism and they are the key to understanding the performance of the core of GFS2. The bmap (block map) tracepoints can be used to monitor block allocations and block mapping (lookup of already allocated blocks in the on-disk metadata tree) as they happen and check for any issues relating to locality of access. The log tracepoints keep track of the data being written to and released from the journal and can provide useful information on that part of GFS2.

跟踪点的设计是尽量通用，就是说在使用 Red Hat Enterprise Linux 6 的过程中不需要更改 API。另一方面，这个界面的用户应了解这是一个 debug 界面，而且不是常规 Red Hat Enterprise Linux 6 API 组件的一部分，同时因为 Red Hat 不保证不会更改 GFS2 中的跟踪点界面。

跟踪点是 Red Hat Enterprise Linux 6 的通用特性，且可完全适用于 GFS2。特别是可将其用于 **blktrace** 架构部署，同时 **blktrace** 跟踪点还可与 GFS2 合用获得系统性能的更全面信息。根据跟踪点所处级别，它们可在很短时间内产生大量数据。其设计旨在启用它们时在系统中添加最小负载，但也不可避免地有一些影响。使用各种方法进行的过滤可帮助减少数据，并帮助集中获得对了解具体情况有帮助的信息。

C.2. 跟踪点

您可在 **/sys/kernel/debug/tracing/** 目录中找到跟踪点，假设将 **debugfs** 挂载到 **/sys/kernel/debug** 目录的标准位置。**events** 子目录包含可指定的所有跟踪事件，同时如果载入 **gfs2** 模块，则会有 **gfs2** 子目录，该子目录中包含下一级子目录，每个 GFS2 事件一个目录。**/sys/kernel/debug/tracing/events/gfs2** 目录的内容应类似如下：

```
[root@chywoon gfs2]# ls
enable            gfs2_bmap       gfs2_glock_queue         gfs2_log_flush
filter            gfs2_demote_rq  gfs2_glock_state_change  gfs2_pin
gfs2_block_alloc  gfs2_glock_put  gfs2_log_blocks          gfs2_promote
```

请运行以下命令启用所有 GFS2 跟踪点：

```
[root@chywoon gfs2]# echo -n 1
>/sys/kernel/debug/tracing/events/gfs2/enable
```

要启用具体跟踪点，则会在每个独立事件的子目录中都有一个 **enable** 文件。同样也会有一个 **filter** 文件，可将其用来为每个事件或者一组事件设置事件过滤器。下面会具体解释独立事件的含义。

跟踪点的输出结果有 ASCII 或二进制两种格式。这个附录目前不介绍二进制界面。ASCII 界面有两种使用方法。您可以运行以下命令列出环缓冲的内容：

```
[root@chywoon gfs2]# cat /sys/kernel/debug/tracing/trace
```

这个界面适用于在某一段时间内使用长时间运行进程的情况，在一些事件后，想要查看缓冲中最新捕获的信息。另一个界面 **/sys/kernel/debug/tracing/trace_pipe** 可用于需要所有输出结果的情况。事件发生时即可在这个文件中读取，在这个界面中没有可用的历史记录。输出结果的格式在两个界面中是一样的，本附录后面的小节中将为每个 GFS2 事件进行具体描述。

可使用 **trace-cmd** 程序读取跟踪点数据。有关这个程序的详情请参考 第 C.10 节 “参考资料”。**trace-cmd** 程序可以类似的方式用于 **strace** 程序，例如：在从各种资源中收集跟踪数据是运行命令。

C.3. GLOCKS

要了解 GFS2，最重要的概念，也是与其他文件系统不同的概念就是 glocks。就源代码而言，glock 是一个数据结构，可将 DLM 和缓存带入一个单一静态机器。每个 glock 都与单一 DLM 锁定有 1:1 对应关系，并为那个锁定状态提供缓存，这样来自文件系统单一节点的竞争操作就不会重复调用 DLM，因此可帮助避免不必要的网络流量。glocks 有两个大类，即缓冲元数据的一类和不缓冲元数据的一类。内节点 glock 和资源组 glock 都可缓冲元数据，其他 glock 类型则不缓冲元数据。内节点 glock 还包括缓冲元数据以外的数据，并拥有所有 glock 中最复杂的逻辑。

表 C.1. Glock 模型和 DLM 锁定模型

Glock 模型	DLM 锁定模型	备注
UN	IV/NL	未锁定（没有与 glock 或者 NL 锁定关联的 DLM 锁定，具体要看 I 标签）
SH	PR	共享（读保护）锁定
EX	EX	排它锁
DF	CW	直接 I/O 和文件系统停滞时使用的延迟（同时写入）

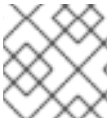
Glocks remain in memory until either they are unlocked (at the request of another node or at the request of the VM) and there are no local users. At that point they are removed from the glock hash table and freed. When a glock is created, the DLM lock is not associated with the glock immediately. The DLM lock becomes associated with the glock upon the first request to the DLM, and if this request is successful then the 'I' (initial) flag will be set on the glock. 表 C.4 “Glock 标签” shows the meanings of the different glock flags. Once the DLM has been associated with the glock, the DLM lock will always remain at least at NL (Null) lock mode until the glock is to be freed. A demotion of the DLM lock from NL to unlocked is always the last operation in the life of a glock.

注意

从 Red Hat Enterprise Linux 5 可开始 DLM 锁行为的某些方面有了变化，即有时会将附加到 glock 的 DLM 锁完全解锁，因此Red Hat Enterprise Linux 5 有不同的机制保证在需要时保留 LVM（锁值块）。由于在 GFS2 中合并 **lock_dlm** 锁模块（不会与 DLM 本身混淆）而使得Red Hat Enterprise Linux 6 使用的新方案成为可能。

Each glock can have a number of "holders" associated with it, each of which represents one lock request from the higher layers. System calls relating to GFS2 queue and dequeue holders from the glock to protect the critical section of code.

使用 glock 的机器是基于工作队列（workqueue）。出于性能考虑，更倾向于使用小任务（tasklet），但在当前部署中需要从禁止其使用的上下文中提交 I/O。



注意

工作队列有其自身的跟踪点，如需要可与 GFS2 跟踪点联合使用。

表 C.2 “Glock 模式和数据类型” 显示每个 glock 模块下可能缓存的状态是什么，以及缓存的状态是否 dirty。这可适用于内节点和资源组锁，虽然资源组锁中没有数据内容，只有元数据。

表 C.2. Glock 模式和数据类型

Glock 模型	缓存数据	缓存元数据	脏数据	脏数据
UN	否	否	否	否
SH	是	是	否	否
DF	否	是	否	否
EX	是	是	是	是

C.4. GLOCK DEBUGFS 界面

glock **debugfs** 界面允许 glock 和 holder 内部状态的虚拟化，同时还包含一些在某些情况下被锁定的对象详情小结。文件的每一行都以 G 打头：无缩进（代表 glock 本身）或者以不同字母打头，缩进一个空格，代表文件中紧挨着它的 glock 关联的结构（H：代表 holder，I：代表内节点，R：代表资源组）。该文件应有类似如下示例中的内容：

```
G:  s:SH n:5/75320 f:I t:SH d:EX/0 a:0 r:3
 H: s:SH f:EH e:0 p:4466 [postmark] gfs2_inode_lookup+0x14e/0x260 [gfs2]
G:  s:EX n:3/258028 f:yI t:EX d:EX/0 a:3 r:4
 H: s:EX f:tH e:0 p:4466 [postmark] gfs2_inplace_reserve_i+0x177/0x780
[gfs2]
 R: n:258028 f:05 b:22256/22256 i:16800
G:  s:EX n:2/219916 f:yfI t:EX d:EX/0 a:0 r:3
 I: n:75661/219916 t:8 f:0x10 d:0x00000000 s:7522/7522
G:  s:SH n:5/127205 f:I t:SH d:EX/0 a:0 r:3
 H: s:SH f:EH e:0 p:4466 [postmark] gfs2_inode_lookup+0x14e/0x260 [gfs2]
G:  s:EX n:2/50382 f:yfI t:EX d:EX/0 a:0 r:2
G:  s:SH n:5/302519 f:I t:SH d:EX/0 a:0 r:3
 H: s:SH f:EH e:0 p:4466 [postmark] gfs2_inode_lookup+0x14e/0x260 [gfs2]
G:  s:SH n:5/313874 f:I t:SH d:EX/0 a:0 r:3
 H: s:SH f:EH e:0 p:4466 [postmark] gfs2_inode_lookup+0x14e/0x260 [gfs2]
G:  s:SH n:5/271916 f:I t:SH d:EX/0 a:0 r:3
 H: s:SH f:EH e:0 p:4466 [postmark] gfs2_inode_lookup+0x14e/0x260 [gfs2]
G:  s:SH n:5/312732 f:I t:SH d:EX/0 a:0 r:3
 H: s:SH f:EH e:0 p:4466 [postmark] gfs2_inode_lookup+0x14e/0x260 [gfs2]
```

The above example is a series of excerpts (from an approximately 18MB file) generated by the command **cat /sys/kernel/debug/gfs2/unity:myfs/glocks >my.lock** during a run of the postmark benchmark on a single node GFS2 file system. The glocks in the figure have been selected in

order to show some of the more interesting features of the glock dumps.

glock 状态可以是 EX（独家，exclusive）、DF（延迟，deferred）、SH（共享，shared）或者 UN（解锁）。这些状态都直接与 DLM 所模式对应，UN 除外，因为它可能表示 DLM null 锁状态，或者 GFS2 没有 DLM 锁（请参考上面的具体 I 标签解释）。 glock 的 s: 字段表示请求的模式。如果有锁定，则 holder 会在其标签（f: 字段）中有 H 字节。否则它就会有 W（等待）字节。

The n: field (number) indicates the number associated with each item. For glocks, that is the type number followed by the glock number so that in the above example, the first glock is n:5/75320; that is, an **iopen** glock which relates to inode 75320. In the case of inode and **iopen** glocks, the glock number is always identical to the inode's disk block number.

注意

debugfs glock 文件中的 glock 数（n: 字段）是十六进制，但跟踪点输出则以十进制列出它们。这是有历史原因：glock 数一直是以十六进制编写，但为跟踪点选择的是十进制数，这样是为了方便与其它跟踪点输出（例如 **blktrace** 的输出）和 **stat**(1) 的输出进行比较。

有关 holder 和 glock 标签的完整列表请参考 表 C.4 “Glock 标签” 和 表 C.5 “Glock holder 标签”。glock **debugfs** 界面中还没有锁值块的内容。

表 C.3 “Glock 类型” 演示了不同 glock 类型的含义。

表 C.3. Glock 类型

类型数	锁类型	用法
1	trans	事务锁
2	inode	内节点元数据和数据
3	rgrp	资源组元数据
4	meta	超级块
5	iopen	内节点最后一次探测
6	flock	**flock**(2) syscall
8	quota	配额操作
9	journal	日志互斥

最重要的 glock 标签之一是 I（锁定）标签。这是字节锁，是用来在执行状态更改时对 glock 状态进行仲裁访问。在状态机器要通过 DLM 发送远程锁请求时会设置该标签，且只有在完成操作后才会清除。有时这可以意味着已发出一个以上锁请求，在间隔间出现各种无效。

表 C.4 “Glock 标签” 演示了不同 glock 标签的含义。

表 C.4. Glock 标签

标签	名称	含义
d	等待降级	递延（远程）降级请求
D	降级	降级请求（本地或者远程）
f	清除日志	释放这个 glock 前需要提交该日志
F	冻结	回复忽略的远程节点 -- 恢复进行中。
i	失效进行中	正在让这个 glock 中的页面失效
I	Initial	设定什么时候 DLM 锁与这个 glock 关联
l	Locked	glock 正在更改状态的过程中
L	LRU	当 glock 在 LRU 列表中时设置
o	对象	glock 与某个对象关联时设定（即用于类型 2 glock 的内节点以及用于类型 3 glock 的资源组）
p	降级中	glock 正在与降级请求响应
q	排队的	拥有者排队等待 glock 时设定，并在持有 glock 但没有拥有者时清除。是用于计算 glock 最小拥有时间的算法的一部分。
r	回复等待	从远程节点中接收的回复正在等待过程中
y	脏数据	释放这个 glock 前要刷新到磁盘的数据

当从以与本地节点冲突的模式获得锁定的某个节点收到远程 callback 时，会设定标签 D（降级）或者 d（降级等待）标签。要防止在竞争具体锁时出现匮乏情况，为每个锁都分配了最小保留时间。如果某个节点的锁没有最小保留时间，则允许保留该锁直到时间间隔过期。

如果时间间隔过期，则会设定 D（demote，降级）标签，并记录请求的状态。这样，下次就不会在 holder 队列中设置锁，该锁就会被降级。如果时间间隔没有过期，那么就会设置 d（降级等待）标签，并在超过最短保留时间时设定 D（降级）标签。

将为 glock 分配 DLM 锁时会为其设置 I（initial）标签。这种情况会在第一次使用 glock 时发生，然后这个 I 标签会一直保留到最终释放 glock（即解开 DLM 锁）为止。

C.5. GLOCK HOLDER

表 C.5 “Glock holder 标签” 演示了不同 glock holder 标签的含义。

表 C.5. Glock holder 标签

标签	名称	含义
a	Async	不等待 glock 结果（稍后会得到结果）
A	Any	接受所有兼容锁模式
c	没有缓存	取消锁定时立即降级 DLM 锁定
e	没有过期日期	忽略随后的取消锁定请求
E	准确	必须有 exact 锁定模式
F	第一	设定赋予这个锁定的第一个拥有者
H	拥有者	表示赋予请求的锁定
p	优先权	在队列头入队的拥有者
t	Try	A "try" lock
T	Try 1CB	A "try" lock that sends a callback
W	Wait	等待请求完成的设置

最重要的拥有者标签之前提到的是 H（拥有者）和 W（等待），因为是将它们分别设定在已分配锁定请求和已排队锁定请求中。该列表中拥有者的顺序分厂重要。如果有任何已分配拥有者，他们总是位于队列的前端，后面跟着的是已排队拥有者。

如果没有任何已分配拥有者，那么该列表中的第一个拥有者将触发下一个状态更改。因为降级请求总是比该文件系统中的请求有更高的优先权，但那并不一定会有状态请求更改。

The glock subsystem supports two kinds of "try" lock. These are useful both because they allow the taking of locks out of the normal order (with suitable back-off and retry) and because they can be used to help avoid resources in use by other nodes. The normal t (try) lock is basically just what its name indicates; it is a "try" lock that does not do anything special. The T (**`try 1CB`**) lock, on the other hand, is identical to the t lock except that the DLM will send a single callback to current incompatible lock holders. One use of the T (**`try 1CB`**) lock is with the **`iopen`** locks, which are used to arbitrate among the nodes when an inode's **`i_nlink`** count is zero, and determine which of the nodes will be responsible for deallocating the inode. The **`iopen`** glock is normally held in the shared state, but when the **`i_nlink`** count becomes zero and **`->delete_inode`**() is called, it will request an exclusive lock with T (**`try 1CB`**) set. It will continue to deallocate the inode if the lock is granted. If the lock is not granted it will result in the node(s) which were preventing the grant of the lock marking their glock(s) with the D (demote) flag, which is checked at **`->drop_inode`**() time in order to ensure that the deallocation is not forgotten.

This means that inodes that have zero link count but are still open will be deallocated by the node on which the final **`close`**() occurs. Also, at the same time as the inode's link count is decremented to zero the inode is marked as being in the special state of having zero link count but still in use in the resource

group bitmap. This functions like the ext3 file system3's orphan list in that it allows any subsequent reader of the bitmap to know that there is potentially space that might be reclaimed, and to attempt to reclaim it.

C.6. GLOCK 跟踪点

跟踪点也用于确认对缓冲控制的修正，方法是将其与 blktrace 输出结果一同使用，同时要了解磁盘布局。这样就有可能检查任何给出的 I/O 是否已使用正确的锁定发出或者完成，且目前尚无竟态出现。

gfs2_glock_state_change 跟踪点是最需要了解的一个。它跟踪 glock 的每次状态更改，从创建之初直到到使用 **gfs2_glock_put** 降级，以及最后使用 NL 取消传输锁定。总是在状态更改发生前设定 l（锁定的）glock 标签，直到最后完成后才会清除。在状态更改期间绝不会有任何分配的拥有者（H glock 拥有者标签）。如果有任何排队的拥有者，他们总是出于 W（等待中）专题。当状态更改完成后，可在清除 l glock 标签前为拥有者分配标签，这是最后的操作。

gfs2_demote_rq 跟踪点一直跟踪本地和远程降级请求。假设在该节点在还有足够的内存，本地降级请求很少见，同时大多数经常由卸载或者偶尔的内存回收产生。远程降级请求数是节点间就某个具体内节点或者资源组进行竞争的指数。

当为拥有者分配一个锁定后会调用 **gfs2_promote**，这发生在状态更改的最后阶段，或者由于 glock 状态已被缓存到适当的形式的锁定而可以立即分配的锁定请求中。如果该拥有者是这个 glock 的第一个使用者，那么为那个拥有者设定 f（第一）标签。目前只在资源组中使用。

C.7. BMAP 跟踪点

块映射是所有文件系统的任务核心。GFS2 使用传统基于 bitmap 的系统，每个块占用 2 个字节。该跟踪点的主要目的是可在这个子系统中监控分配和映射块所使用的时间。

每个 bmap 操作会调用两次 **gfs2_bmap** 跟踪点：一次是在启动时显示 bmap 请求，一次是在结束时显示结果。这更方便匹配请求和结果，并测量匹配块以及文件系统的不同部分、不同文件差值或者甚至不同的文件所需时间。还可以查看相比请求的扩展，所返回扩展的平均大小。

要保证随时跟踪分配的块，**gfs2_block_alloc** 不仅在分配块时调用，在释放块时也调用。因为分配是根据要进行块分配的内节点进行，可使用它跟踪在实际文件系统中哪些物理块属于哪个文件。这在与 **blktrace** 合用时非常有帮助，后者将显示有问题的 I/O 模式，然后使用通过这个跟踪点获得的映射返回相关内节点。

C.8. 记录跟踪点

这个子系统中的跟踪点跟踪被添加到日志（**gfs2_pin**）以及从该日志中删除的块，同时还在日志（**gfs2_log_flush**）记录此事务所消耗的时间。这在解决日志性能问题是非常有帮助。

gfs2_log_blocks 跟踪点在日志中记录保留的块的踪迹，这样可帮助您查看日志对于该负载是否太小。例如：

gfs2_ail_flush 跟踪点（Red Hat Enterprise Linux 6.2 以及之后的版本）在跟踪清除 AIL 列表的起始时间上与 **gfs2_log_flush** 跟踪点类似。AIL 列表包含已列入该日志但还没有写入的缓冲，同时这是一个周期性清除以便释放更多的空间供文件系统使用，或者在请求同步或者 fsync 时使用。

C.9. GLOCK 统计

GFS2 保留统计数据以帮助跟踪文件系统中的操作。这可以让您准确捕捉性能问题。

GFS2 保留两个计数器：

- **dcount**，记录要求的 DLM 操作数。这显示有多少数据已进行平均/方差计算。
- **qcount**，记录要求的 **syscall** 级操作数。通常 **qcount** 会大于等于 **dcount**。

另外，GFS2 还保留三个平均/方差对。这些平均/方差对是平滑指数预估值，所采用算法是计算网络节点中轮询数的算法。GFS2 中保留的这些平均值和方差对不按照比例计算，而是以纳秒整数值为单位计算。

- srtt/srttvar：非阻塞操作的平滑轮询时间
- srttb/srttvarb：阻塞操作的平滑轮询时间
- irtt/irttvar：请求间隔时间（例如：DLM 请求之间的间隔时间）

A non-blocking request is one which will complete right away, whatever the state of the DLM lock in question. That currently means any requests when (a) the current state of the lock is exclusive (b) the requested state is either null or unlocked or (c) the "try lock" flag is set. A blocking request covers all the other lock requests.

较长时间适用于 IRTT，同时较短的时间适用于 RTT。

在两个 **sysfs** 文件中保存统计数据：

- The **glstats** file. This file is similar to the **glocks** file, except that it contains statistics, with one glock per line. The data is initialized from "per cpu" data for that glock type for which the glock is created (aside from counters, which are zeroed). This file may be very large.
- The **lkstats** file. This contains "per cpu" stats for each glock type. It contains one statistic per line, in which each column is a cpu core. There are eight lines per glock type, with types following on from each other.

C.10. 参考资料

有关跟踪点和 GFS2 **glocks** 文件详情请参考以下资源：

- 本附录包含 Steve Whitehouse 于 2009 年度 Linux 研讨会中的报告，该报告的具体地址如下：http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=blob;f=Documentation/filesystems/gfs2-glocks.txt;h=0494f78d87e40c225eb1dc1a1489acd891210761;hb=HEAD。
- 有关 glock 内部锁定规则的详情请参考 http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=blob;f=Documentation/filesystems/gfs2-glocks.txt;h=0494f78d87e40c225eb1dc1a1489acd891210761;hb=HEAD。
- 有关事件跟踪的详情请参考 http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=blob;f=Documentation/trace/events.txt;h=09bd8e9029892e4e1d48078de4d076e24e
- 有关 **trace-cmd** 程序的详情请参考 http://lwn.net/Articles/341902/。

附录 D. 修订历史

修订 7.1-3.2 **Mon Jan 09 2015** **Leah Liu**
完成翻译、校对。

修订 7.1-3 **Tue Dec 16 2014** **Steven Levine**
更新，在 RHEL 6 初始页面中应用 sort_order。

修订 7.1-3 **Tue Dec 16 2014** **Steven Levine**
Updating to implement sort_order on the RHEL 6 splash page.

修订 7.0-9 **Wed Oct 8 2014** **Steven Levine**
6.6 GA 发行本

修订 7.0-9 **Wed Oct 8 2014** **Steven Levine**
Version for 6.6 GA release

修订 7.0-8 **Thu Aug 7 2014** **Steven Levine**
6.6 Beta 发行本

修订 7.0-8 **Thu Aug 7 2014** **Steven Levine**
Version for 6.6 Beta release

修订 7.0-4 **Thu Jul 17 2014** **Steven Levine**
解决 #1102591
添加在 Pacemaker 集群中配置 GFS2 的步骤

修订 7.0-4 **Thu Jul 17 2014** **Steven Levine**
Resolves #1102591
Adds procedure for configuring GFS2 in a Pacemaker cluster

修订 7.0-3 **Wed Jul 16 2014** **Steven Levine**
解决 #1035119
更新 Glock 标签表并在 Glock 统计中添加一个小节

修订 7.0-3 **Wed Jul 16 2014** **Steven Levine**
Resolves #1035119
Updates table of Glock flags and adds a section on Glock statistics

修订 7.0-1 **Thu Jun 5 2014** **Steven Levine**
6.6 发行本初稿

修订 7.0-1 **Thu Jun 5 2014** **Steven Levine**
First draft for 6.6 release

修订 6.0-6 **Wed Nov 13 2013** **Steven Levine**
6.5 GA 发行本

修订 6.0-6 **Wed Nov 13 2013** **Steven Levine**
Version for 6.5 GA release

修订 6.0-5 **Fri Sep 27 2013** **Steven Levine**
6.5 Beta 发行本

修订 6.0-5 **Fri Sep 27 2013** **Steven Levine**
Version for 6.5 Beta release

修订 6.0-3 **Fri Sep 27 2013** **Steven Levine**
解决 #960841
声明在 GFS2 文件系统中不支持 SELinux。

修订 6.0-3 **Fri Sep 27 2013** **Steven Levine**
Resolves #960841
Clarifies lack of support for SELinux with GFS2 filesystems.

修订 6.0-1 **Fri Sep 06 2013** **Steven Levine**
添加有关 Samba 和 GFS2 的备注

修订 6.0-1 **Fri Sep 06 2013** **Steven Levine**
Adding note about Samba and GFS2

修订 5.0-7 **Mon Feb 18 2013** **Steven Levine**
6.4 GA 发行本

修订 5.0-7 **Mon Feb 18 2013** **Steven Levine**
Version for 6.4 GA release

修订 5.0-5 **Mon Nov 26 2012** **Steven Levine**
6.4 Beta 发行本

修订 5.0-5 **Mon Nov 26 2012** **Steven Levine**
Version for 6.4 Beta release

修订 5.0-4 **Tue Nov 13 2012** **Steven Levine**
解决 #860324
更新 GFS2 配置以及操作注意事项一章，有一些小的说明。

解决 #807057
添加注释，即建议您在部署前请授权 Red Hat 代表确认您的配置。

修订 5.0-4 **Tue Nov 13 2012** **Steven Levine**
Resolves #860324
Updates chapter on GFS2 configuration and operational considerations with small clarifications.

Resolves #807057
Adds note recommending consultation with an authorized Red Hat representative to verify your configuration prior to deployment.

修订 5.0-1 **Mon Oct 15 2012** **Steven Levine**
更新操作注意事项一章。

修订 5.0-1 **Mon Oct 15 2012** **Steven Levine**
Updated chapter on operational considerations.

修订 4.0-2 **Thu Mar 28 2012** **Steven Levine**
发布版本 6.3 GA

修订 4.0-2 **Thu Mar 28 2012** **Steven Levine**
Version for 6.3 GA release

修订 4.0-1 **Thu Mar 28 2012** **Steven Levine**

解决：#782482, #663944
文件 GFS2 配置和操作注意事项一章。

解决：#757742
说明在 GFS2 中使用 CLVM 的必要性。

解决：#786621
修改一些小的排版错误。

修订 4.0-1 **Thu Mar 28 2012** **Steven Levine**
Resolves: #782482, #663944
Adds new chapter on GFS2 configuration and operational considerations.

Resolves: #757742
Clarifies necessity for using GFS2 with CLVM.

Resolves: #786621
Fixes small typographical error.

修订 3.0-2 **Thu Dec 1 2011** **Steven Levine**
Red Hat Enterprise Linux 6.2 的 GA 发行版本

修订 3.0-2 **Thu Dec 1 2011** **Steven Levine**
Release for GA of Red Hat Enterprise Linux 6.2

修订 3.0-1 **Mon Sep 19 2011** **Steven Levine**
Red Hat Enterprise Linux 6.2 Beta 发行本初始修订

解决：#704179
编写支持 **tunegfs2** 命令的文档。

解决：#712390
添加 GFS2 跟踪点一章。

解决：#705961
解决小的排版错误。

修订 3.0-1 **Mon Sep 19 2011** **Steven Levine**
Initial revision for Red Hat Enterprise Linux 6.2 Beta release

Resolves: #704179
Documents support for the **tunegfs2** command.

Resolves: #712390
Adds new appendix on GFS2 tracepoints.

Resolves: #705961
Resolves minor typographical errors.

修订 2.0-1 **Thu May 19 2011** **Steven Levine**

Red Hat Enterprise Linux 6.1 的初始发行本

解决：#549838
编写 Red Hat Enterprise Linux 6.1 中标准 Linux 配额工具支持文档。

解决：#608750
澄清 GFS2 撤回功能。

解决：#660364
修正最大 GFS2 文件系统大小信息。

解决：#687874
在 GFS2 故障排除中添加新的一章。

解决：#664848
添加从 GFS 转换为 GFS2 之前查找上下文独立路径名称的信息。

修订 2.0-1 **Thu May 19 2011** **Steven Levine**

Initial release for Red Hat Enterprise Linux 6.1

Resolves: #549838
Documents support for standard Linux quota facilities in Red Hat Enterprise Linux 6.1.

Resolves: #608750
Clarifies description of GFS2 withdraw function.

Resolves: #660364
Corrects maximum GFS2 file system size information.

Resolves: #687874
Adds new chapter on GFS2 troubleshooting.

Resolves: #664848
Adds information on finding Context-Dependent Path Names before converting from GFS to GFS2.

修订 1.0-1 **Wed Nov 15 2010** **Steven Levine**

Red Hat Enterprise Linux 6 初始发行本

修订 1.0-1 **Wed Nov 15 2010** **Steven Levine**

Initial release for Red Hat Enterprise Linux 6

索引

符号

A

D

F

G

M

P

Q