Red Hat Enterprise Linux 6

Global File System 2

Red Hat Global File System 2

Ausgabe 7

Last Updated: 2017-10-12

Red Hat Enterprise Linux 6 Global File System 2

Red Hat Global File System 2
Ausgabe 7

Rechtlicher Hinweis

Zusammenfassung

Dieses Buch liefert Informationen über die Konfiguration und Wartung von Red Hat GFS2 (Red Hat Global File System 2) für Red Hat Enterprise Linux 6.

Inhaltsverzeichnis

EINFÜHRUNG

Dieses Buch liefert Informationen über die Konfiguration und Wartung von Red Hat GFS2 (Red Hat Global File System 2), das Bestandteil des Red Hat Resilient Storage Add-Ons ist.

1. ZIELGRUPPE

Dieses Buch richtet sich in erster Linie an Linux-Systemadministratoren, die mit den folgenden Arbeitsvorgängen bereits vertraut sind:

- Verfahren zur Linux-Systemadministration, einschließlich Kernel-Konfiguration
- Installation und Konfiguration von gemeinsam genutzten Speichernetzwerken, wie z. B. Fibre-Channel-SANs

2. ERGÄNZENDE DOKUMENTATION

Werfen Sie einen Blick auf die folgenden Quellen für weitere Informationen zur Verwendung von Red Hat Enterprise Linux:

- *Installationshandbuch* – Liefert Informationen bezüglich der Installation von Red Hat Enterprise Linux 6.
- *Deployment Guide* – Liefert Informationen zur Bereitstellung, zur Konfiguration und zur Verwaltung von Red Hat Enterprise Linux 6.
- *Storage Administration Guide* – Liefert Informationen über die effiziente Verwaltung von Speichergeräten und Dateisystemen auf Red Hat Enterprise Linux 6.

Weitere Informationen über das High Availability Add-On und das Resilient Storage Add-On für Red Hat Enterprise Linux 6 finden Sie in den folgenden Quellen:

- *High Availability Add-On Overview* – Liefert einen allgemeinen Überblick über das High Availability Add-On.
- *Cluster-Administration* – Liefert Informationen zur Installation, Konfiguration und Verwaltung des High Availability Add-Ons.
- *Administration des Logical Volume Manager* – Liefert eine Beschreibung des Logical Volume Managers (LVM), inklusive Informationen zum Einsatz von LVM in einer Cluster-Umgebung.
- *DM Multipath* – Liefert Informationen über die Verwendung der Device-Mapper-Multipath-Funktionen von Red Hat Enterprise Linux.
- *Verwaltung des Load Balancer Add-Ons* – Liefert Informationen zur Konfiguration von Hochleistungssystemen und -diensten mit dem Red Hat Load Balancer Add-On, einer Gruppe integrierter Software-Komponenten, die Linux Virtual Server (LVS) bereitstellen, um IP-Lasten über eine Gruppe realer Server zu verteilen.
- *Versionshinweise* – Liefert Informationen über die jeweils aktuelle Release der Red Hat Produkte.

Red Hat Cluster Suite Dokumentation und andere Red Hat Dokumente stehen als HTML-, PDF- und RPM-Versionen auf der Red Hat Enterprise Linux Dokumentations-CD und online unter https://access.redhat.com/site/documentation/ zur Verfügung.

3. WIR FREUEN UNS AUF IHR FEEDBACK!

Falls Sie einen Fehler in diesem Handbuch finden oder eine Idee haben, wie dieses verbessert werden könnte, freuen wir uns über Ihr Feedback! Bitte reichen Sie einen Fehlerbericht in Bugzilla (http://bugzilla.redhat.com/) für das Produkt **Red Hat Enterprise Linux 6** und die Komponente **doc-Global_File_System_2** ein. Vergewissern Sie sich beim Einreichen eines Fehlerberichts, dass Sie die Kennung des Handbuchs mit angeben:

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

Falls Sie uns einen Vorschlag zur Verbesserung der Dokumentation senden möchten, sollten Sie hierzu möglichst genaue Angaben machen. Wenn Sie einen Fehler gefunden haben, geben Sie bitte die Nummer des Abschnitts und einen Ausschnitt des Textes an, damit wir diesen leicht finden können.

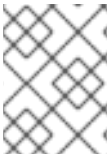
KAPITEL 1. ÜBERBLICK ÜBER GFS2

Das Red Hat GFS2-Dateisystem ist Bestandteil des Resilient Storage Add-Ons. Es ist ein natives Dateisystem, das sich direkt mit der Dateisystem-Schnittstelle des Linux-Kernels (VFS-Schicht) verbindet. Wenn GFS2 als Cluster-Dateisystem implementiert wird, verwendet es verteilte Metadaten und multiple Journale. Red Hat unterstützt die Verwendung des GFS2-Dateisystems nur wie vom High Availability Add-On implementiert.

ANMERKUNG

Obwohl das GFS2-Dateisystem sowohl auf einem eigenständigen System als auch als Teil einer Cluster-Konfiguration implementiert werden kann, unterstützt Red Hat den Einsatz von GFS2 nicht für Ein-Knoten-Systeme. Red Hat unterstützt jedoch eine Reihe von leistungsstarken Ein-Knoten-Dateisystemen, die für den Einsatz auf einzelnen Knoten optimiert sind und dadurch meist einen geringeren Overhead als ein Cluster-Dateisystem haben. Red Hat empfiehlt den Einsatz eines dieser Dateisysteme anstelle von GFS2 in Fällen, in denen nur ein einzelner Knoten das Dateisystem einhängen muss.

Red Hat unterstützt auch weiterhin Ein-Knoten-GFS2-Dateisysteme zum Einhängen von Snapshots von Cluster-Dateisystemen (z. B. zwecks Datensicherung).



ANMERKUNG

Red Hat unterstützt GFS2 nicht für Cluster-Dateisystem-Bereitstellungen mit mehr als 16 Knoten.

GFS2 basiert auf einer 64-Bit-Architektur, die theoretisch ein 8 EB Dateisystem handhaben kann. Allerdings beträgt die derzeit maximal unterstützte Größe für ein GFS2-Dateisystem für 64-Bit-Hardware 100 TB. Die derzeit maximal unterstützte Größe für ein GFS2-Dateisystem für 32-Bit-Hardware beträgt 16 TB. Falls Ihr System ein größeres GFS2-Dateisystem erfordert, setzen Sie sich bitte mit Ihrem Red Hat Service-Vertreter in Verbindung.

Wenn Sie die Größe Ihres Dateisystems festlegen, sollten Sie auch Ihre Ansprüche an die Wiederherstellung berücksichtigen. Das Ausführen des **`fsck.gfs2`**-Befehls auf einem sehr großen Dateisystem kann eine lange Zeit in Anspruch nehmen und zudem viel Speicherkapazität verbrauchen. Darüber hinaus wird bei einem Ausfall der Festplatte oder eines Untersystems die Wiederherstellungszeit durch die Schnelligkeit Ihres Sicherungsmediums begrenzt. Weitere Informationen über die Menge an Arbeitsspeicher, die der **`fsck.gfs2`**-Befehl benötigt, finden Sie in Abschnitt 4.11, »Reparieren eines Dateisystems«.

Wenn Red Hat GFS2-Knoten als Teil eines Clusters konfiguriert werden, können diese mithilfe der Konfigurations- und Verwaltungstools des High Availability Add-Ons konfiguriert und verwaltet werden. Red Hat GFS2 ermöglicht in diesem Fall eine gemeinsame Verwendung von Daten unter den GFS2-Knoten in einem Cluster, wobei eine einzelne, konsistente Ansicht des Dateisystem-Namensraums über die GFS2-Knoten hinweg geboten wird. Dies ermöglicht es Prozessen auf verschiedenen Knoten, GFS2-Dateien in der gleichen Art und Weise gemeinsam zu verwenden, wie Prozesse desselben Knotens Dateien auf dem lokalen Dateisystem gemeinsam verwenden können – und das ohne erkennbaren Unterschied. Weitere Informationen über das High Availability Add-On finden Sie unter *Konfiguration und Verwaltung eines Red Hat Clusters*.

Obwohl ein GFS2-Dateisystem auch außerhalb von LVM eingesetzt werden kann, unterstützt Red Hat nur solche GFS2-Dateisysteme, die auf einem logischen CLVM-Datenträger angelegt wurden. CLVM ist Bestandteil des Resilient Storage Add-Ons und ist eine clusterweite Implementierung von LVM, betrieben vom CLVM-Daemon **`clvmd`**, der logische LVM-Datenträger im Cluster verwaltet. Der Daemon

erlaubt die Verwendung von LVM2, um logische Datenträger im Cluster zu verwalten, und ermöglicht es allen Knoten im Cluster, die logischen Datenträger gemeinsam zu verwenden. Weitere Informationen zur Datenträgerverwaltung mit LVM finden Sie im Handbuch *Administration des Logical Volume Manager*.

Das Kernel-Modul **gfs2.ko** implementiert das GFS2-Dateisystem und wird auf GFS2-Cluster-Knoten geladen.

ANMERKUNG

Wenn Sie ein GFS2-Dateisystem als ein Cluster-Dateisystem konfigurieren, müssen Sie sicherstellen, dass alle Knoten im Cluster Zugriff auf den gemeinsamen Speicher haben. Asymmetrische Cluster-Konfigurationen, bei denen einige Knoten Zugriff auf den Speicher haben und andere nicht, werden nicht unterstützt. Es ist jedoch nicht nötig, dass alle Knoten das GFS2-Dateisystem auch tatsächlich selbst einhängen.

Dieses Kapitel liefert einige kurze, grundlegende Informationen zum besseren Verständnis von GFS2. Es umfasst die folgenden Abschnitte:

- Abschnitt 1.1, »Neue und veränderte Funktionen«
- Abschnitt 1.2, »Vor der Einrichtung von GFS2«
- Abschnitt 1.4, »Unterschiede zwischen GFS und GFS2«
- Abschnitt 1.3, »Installation von GFS2«
- Abschnitt 2.9, »GFS2-Knotensperrung«

1.1. NEUE UND VERÄNDERTE FUNKTIONEN

Dieser Abschnitt führt die neuen bzw. veränderten Funktionen des GFS2-Dateisystems und der GFS2-Dokumentation auf, die in der ersten Release und nachfolgenden Releases von Red Hat Enterprise Linux 6 enthalten sind.

1.1.1. Neue und veränderte Funktionen für Red Hat Enterprise Linux 6.0

Red Hat Enterprise Linux 6.0 führt die folgenden Änderungen und Aktualisierungen an Dokumentationen und Funktionen ein.

- Für die Red Hat Enterprise Linux 6 Release unterstützt Red Hat nicht länger den Einsatz von GFS2 als Ein-Knoten-Dateisystem.
- In der Red Hat Enterprise Linux 6 Release wurde der **gfs2_convert**-Befehl zur Aktualisierung von einem GFS- auf ein GFS2-Dateisystem verbessert. Für weitere Informationen über diesen Befehl siehe Anhang B, *Konvertieren eines Dateisystems von GFS auf GFS2*.
- Die Red Hat Enterprise Linux 6 Release unterstützt die Einhängeoptionen **discard**, **nodiscard**, **barrier**, **nobarrier**, **quota_quantum**, **statfs_quantum** und **statfs_percent**. Für weitere Informationen über das Einhängen eines GFS2-Dateisystems siehe Abschnitt 4.2, »Einhängen eines Dateisystems«.
- Die Red Hat Enterprise Linux 6 Version dieses Dokuments enthält einen neuen Abschnitt, Abschnitt 2.9, »GFS2-Knotensperrung«. Dieser Abschnitt beschreibt einige Interna des GFS2-Dateisystems.

1.1.2. Neue und veränderte Funktionen für Red Hat Enterprise Linux 6.1

Red Hat Enterprise Linux 6.1 führt die folgenden Änderungen und Aktualisierungen an Dokumentationen und Funktionen ein.

- Ab der Red Hat Enterprise Linux 6.1 Release unterstützt GFS2 die standardmäßigen Linux-Funktionen für Festplattenkontingente. GFS2-Kontingentverwaltung ist in Abschnitt 4.5, »Verwalten von GFS2-Festplattenkontingenten« dokumentiert.

  In früheren Releases von Red Hat Enterprise Linux erforderte GFS2 den **gfs2_quota**-Befehl zur Verwaltung von Festplattenkontingenten. Die Dokumentation für den **gfs2_quota**-Befehl finden Sie jetzt in Anhang A, *GFS2-Kontingentverwaltung mit dem Befehl **gfs2_quota***.

- Dieses Dokument enthält nun ein neues Kapitel, Kapitel 5, *Diagnose und Behebung von Problemen mit GFS2-Dateisystemen*.

- Im gesamten Dokument wurden kleinere Fehler korrigiert und einige technische Sachverhalte verdeutlicht.

1.1.3. Neue und veränderte Funktionen für Red Hat Enterprise Linux 6.2

Red Hat Enterprise Linux 6.2 führt die folgenden Änderungen und Aktualisierungen an Dokumentationen und Funktionen ein.

- Ab der Red Hat Enterprise Linux 6.2 Release unterstützt GFS2 den **tunegfs2**-Befehl, der einige der Funktionen des **gfs2_tool**-Befehls ersetzt. Für weitere Informationen siehe die man-Seite für **tunegfs2**.

  Die folgenden Abschnitte wurden aktualisiert und bieten Verwaltungsverfahren, die nicht die Verwendung des **gfs2_tool**-Befehls erfordern:

  - Abschnitt 4.5.4, »Synchronisieren von Kontingenten mit dem **quotasync**-Befehl« und Abschnitt A.3, »Synchronisieren von Kontingenten mit dem **gfs2_quota**-Befehl« beschreiben jetzt, wie der **quota_quantum**-Parameter von seinem Standardwert von 60 Sekunden mittels Einhängeoption **quota_quantum=** geändert werden kann.

  - Abschnitt 4.10, »Unterbrechen der Aktivität auf einem Dateisystem« beschreibt nun, wie man Schreibaktivität auf einem Dateisystem mit dem Befehl **dmsetup suspend** unterbrechen kann.

- Dieses Dokument enthält einen neuen Anhang, Anhang C, *GFS2-Tracepoints und die debugfs-Glocks-Datei*. Dieser Anhang beschreibt die Glock-**debugfs**-Schnittstelle und die GFS2-Tracepoints. Er richtet sich an fortgeschrittene Benutzer, die mit Dateisysteminterna vertraut sind und die gerne mehr über den Aufbau von GFS2 und das Debugging von GFS2-spezifischen Fehlern erfahren möchten.

1.1.4. Neue und veränderte Funktionen für Red Hat Enterprise Linux 6.3

Für die Red Hat Enterprise Linux 6.3 Release enthält dieses Dokument ein neues Kapitel, Kapitel 2, *Überlegungen zur Konfiguration und zum Betrieb von GFS2*. Dieses Kapitel enthält Empfehlungen zur Optimierung der Leistung von GFS2, einschließlich Empfehlungen für die Erstellung, Verwendung und Wartung eines GFS2-Dateisystems.

Zusätzlich wurden im gesamten Dokument kleinere Korrekturen vorgenommen und einige Sachverhalte verdeutlicht.

1.1.5. Neue und veränderte Funktionen für Red Hat Enterprise Linux 6.4

Für die Red Hat Enterprise Linux 6.4 Release wurde Kapitel 2, *Überlegungen zur Konfiguration und zum Betrieb von GFS2* aktualisiert, um einige Sachverhalte zu verdeutlichen.

1.1.6. Neue und veränderte Funktionen für Red Hat Enterprise Linux 6.6

Für die Red Hat Enterprise Linux 6.6 Release enthält dieses Dokument ein neues Kapitel, Kapitel 6, *Konfigurieren eines GFS2-Dateisystems in einem Pacemaker-Cluster*. Dieses Kapitel zeigt in groben Zügen die notwendigen Schritte auf, die zur Einrichtung eines Pacemaker-Clusters mit einem GFS2-Dateisystem erforderlich sind.

Zusätzlich wurden im gesamten Dokument kleinere Korrekturen vorgenommen und einige Sachverhalte verdeutlicht.

1.2. VOR DER EINRICHTUNG VON GFS2

Bevor Sie GFS2 installieren und einrichten, legen Sie bitte die folgenden grundlegenden Charakteristiken Ihres GFS2-Dateisystems fest:

GFS2-Knoten

Legen Sie fest, welche Knoten im Cluster die GFS2-Dateisysteme einhängen sollen.

Anzahl der Dateisysteme

Legen Sie fest, wie viele GFS2-Dateisysteme zu Beginn erstellt werden sollen. (Es können später mehr Dateisysteme hinzugefügt werden.)

Namen der Dateisysteme

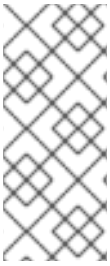
Legen Sie einen eindeutigen Namen für jedes Dateisystem fest. Jeder Name muss eindeutig für alle `lock_dlm`-Dateisysteme im Cluster sein. Jeder Dateisystemname muss in Form einer Parametervariable vorliegen. Für dieses Handbuch werden beispielsweise die Dateisystemnamen **`mydata1`** und **`mydata2`** für einige Beispielverfahren verwendet.

Journale

Legen Sie die Anzahl der Journale für Ihre GFS2-Dateisysteme fest. Für jeden Knoten, der ein GFS2-Dateisystem einhängt, ist ein Journal erforderlich. GFS2 erlaubt Ihnen, Journale später dynamisch hinzuzufügen, wenn nachträglich weitere Server das Dateisystem einhängen. Informationen über das Hinzufügen von Journalen zu einem GFS2-Dateisystem finden Sie in Abschnitt 4.7, »Hinzufügen von Journalen zu einem Dateisystem«.

Speichergeräte und Partitionen

Legen Sie die Speichergeräte und Partitionen fest, die zum Erstellen von logischen Datenträgern (via CLVM) in den Dateisystemen verwendet werden sollen.



ANMERKUNG

Mit GFS2 kann es unter Umständen zu Leistungseinbußen kommen, wenn gleichzeitig von mehr als einem Knoten in demselben Verzeichnis viele Operationen zum Erstellen und Löschen ausgeführt werden. Falls dies auf Ihrem System Probleme verursacht, sollten Sie die Dateierstellung und -löschung in einem Verzeichnis möglichst lokal auf einem Knoten durchführen.

Weitere Empfehlungen zur Erstellung, Verwendung und Wartung eines GFS2-Dateisystems finden Sie in Kapitel 2, *Überlegungen zur Konfiguration und zum Betrieb von GFS2*.

1.3. INSTALLATION VON GFS2

Zusätzlich zu den vom Red Hat High Availability Add-On benötigten Paketen müssen Sie das **gfs2-utils**-Paket für GFS2 und das **lvm2-cluster**-Paket für den Clustered Logical Volume Manager (CLVM) installieren. Die Pakete **lvm2-cluster** und **gfs2-utils** sind Teil des ResilientStorage-Channels, der vor der Installation dieser Pakete aktiviert sein muss.

Sie können den folgenden **yum install** Befehl verwenden, um die Software-Pakete für das Red Hat High Availability Add-On zu installieren:

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

Allgemeine Informationen über das Red Hat High Availability Add-On und über die Cluster-Verwaltung finden Sie im Handbuch *Cluster-Administration*.

1.4. UNTERSCHIEDE ZWISCHEN GFS UND GFS2

Dieser Abschnitt listet die Verbesserungen und Änderungen auf, die GFS2 gegenüber GFS bietet.

Die Migration von GFS auf GFS2 erfordert, dass Sie Ihre GFS-Dateisysteme mithilfe des **gfs2_convert**-Dienstprogramms auf GFS2 konvertieren. Informationen über das **gfs2_convert**-Dienstprogramm finden Sie in Anhang B, *Konvertieren eines Dateisystems von GFS auf GFS2*.

1.4.1. GFS2-Befehlsnamen

Im Allgemeinen ist die Funktionalität von GFS2 mit der von GFS identisch. Allerdings verweisen die Namen der Dateisystembefehle auf GFS2 anstelle von GFS. Tabelle 1.1, »GFS- und GFS2-Befehle« zeigt die GFS-Befehle samt entsprechender GFS2-Befehle.

Tabelle 1.1. GFS- und GFS2-Befehle

GFS-Befehl	GFS2-Befehl	Beschreibung
mount	**mount**	Einhängen eines Dateisystems. Das System ermittelt selbst, ob das Dateisystem ein GFS- oder ein GFS2-Dateisystemtyp ist. Für mehr Informationen zu den GFS2-Einhängeoptionen siehe die man-Seite für gfs2_mount(8).
umount	**umount**	Aushängen eines Dateisystems
fsck **gfs_fsck**	**fsck** **fsck.gfs2**	Überprüfen und Reparieren eines ausgehängten Dateisystems
gfs_grow	**gfs2_grow**	Erweitern eines eingehängten Dateisystems
gfs_jadd	**gfs2_jadd**	Hinzufügen eines Journals zu einem eingehängten Dateisystem

GFS-Befehl	GFS2-Befehl	Beschreibung
`gfs_mkfs` `mkfs -t gfs`	`mkfs.gfs2` `mkfs -t gfs2`	Erstellen eines Dateisystems auf einem Speichergerät
`gfs_quota`	`gfs2_quota`	Verwalten von Festplattenkontingenten auf einem eingehängten Dateisystem. Ab der Red Hat Enterprise Linux 6.1 Release unterstützt GFS2 die standardmäßigen Linux-Funktionen für Festplattenkontingente. Für weitere Informationen über GFS2-Kontingentverwaltung siehe Abschnitt 4.5, »Verwalten von GFS2-Festplattenkontingenten«.

`gfs_tool`	`tunegfs2`
	Einhängeparameter
	`dmsetup suspend`
		Konfigurieren, Einstellen oder Anzeigen von Informationen über ein Dateisystem. Der **`tunegfs2`**-Befehl wird ab der Red Hat Enterprise Linux 6.2 Release unterstützt. Es gibt auch einen **`gfs2_tool`**-Befehl.

`gfs_edit`	`gfs2_edit`	Anzeigen, Drucken oder Bearbeiten der internen Struktur eines Dateisystems. Der **`gfs2_edit`**-Befehl kann für GFS-Dateisysteme wie auch für GFS2-Dateisysteme verwendet werden
`gfs_tool setflag jdata/inherit_jdata`	`chattr +j` (bevorzugt)	Aktivieren des Journalings auf einer Datei oder einem Dateisystem
`setfacl/getfacl`	`setfacl/getfacl`	Anzeigen oder Einstellen der Zugriffskontrollliste einer Datei oder eines Verzeichnisses
`setfattr/getfattr`	`setfattr/getfattr`	Anzeigen oder Einstellen der erweiterten Parameter einer Datei

Eine vollständige Liste der unterstützten Optionen für die GFS2-Dateisystembefehle finden Sie auf den man-Seiten der jeweiligen Befehle.

1.4.2. Weitere Unterschiede zwischen GFS und GFS2

Dieser Abschnitt listet die zusätzlichen Unterschiede hinsichtlich der Administration von GFS und GFS2 auf, die nicht in Abschnitt 1.4.1, »GFS2-Befehlsnamen« beschrieben sind.

1.4.2.1. Kontextabhängige Pfade

GFS2-Dateisysteme unterstützen keine kontextabhängigen Pfade (Context-Dependent Path Names oder kurz CDPNs), die es Ihnen ermöglichen, symbolische Links zu erzeugen, die auf verschiedene

Zieldateien oder -verzeichnisse zeigen. Um dieselbe Funktionalität in GFS2-Dateisystemen zu erreichen, können Sie die **bind**-Option des **mount**-Befehls verwenden. Weitere Informationen über Bind Mounts und kontextabhängige Pfade in GFS2 finden Sie in Abschnitt 4.12, »Bind Mounts und kontextabhängige Pfade«.

1.4.2.2. gfs2.ko-Modul

Das Kernel-Modul, welches das GFS-Dateisystem implementiert, ist **gfs.ko**. Das Kernel-Modul, welches das GFS2-Dateisystem implementiert, ist **gfs2.ko**.

1.4.2.3. Erzwingen von Speicherkontingenten in GFS2

In GFS2-Dateisystemen werden Festplattenkontingente standardmäßig nicht erzwungen, sodass dieses Verhalten explizit aktiviert werden muss. Informationen über das Aktivieren und Deaktivieren von Festplattenkontingenten finden Sie in Abschnitt 4.5, »Verwalten von GFS2-Festplattenkontingenten«.

1.4.2.4. Daten-Journaling

GFS2-Dateisysteme unterstützen die Verwendung des **chattr**-Befehls, um das **j**-Flag auf einer Datei oder einem Verzeichnis zu setzen bzw. zu entfernen. Wird das Flag **+j** auf einer Datei gesetzt, so wird das Daten-Journaling auf dieser Datei aktiviert. Wird das Flag **+j** auf einem Verzeichnis gesetzt, bedeutet dies, dass für alle nachfolgend in diesem Verzeichnis erstellten Dateien oder Verzeichnisse ebenfalls das Daten-Journaling aktiviert ist („inherit jdata“). Die Verwendung des **chattr**-Befehls ist die bevorzugte Methode zum Aktivieren und Deaktivieren des Daten-Journalings auf einer Datei.

1.4.2.5. Journale dynamisch hinzufügen

In GFS-Dateisystemen sind Journale eingebettete Metadaten, die sich außerhalb des Dateisystems befinden, weshalb zunächst der logische Datenträger mit dem Dateisystem vergrößert werden muss, bevor Journale hinzugefügt werden können. In GFS2-Dateisystemen sind Journale einfache (versteckte) Dateien. Dies bedeutet, dass für GFS2-Dateisysteme Journale dynamisch hinzugefügt werden können, sobald weitere Server das Dateisystem einhängen, sofern noch genügend Platz auf dem Dateisystem für zusätzliche Journale vorhanden ist. Informationen über das Hinzufügen von Journalen zu einem GSF2-Dateisystem finden Sie unter Abschnitt 4.7, »Hinzufügen von Journalen zu einem Dateisystem« .

1.4.2.6. atime_quantum-Parameter entfernt

Das GFS2-Dateisystem unterstützt nicht den einstellbaren Parameter **atime_quantum**, der vom GFS-Dateisystem dazu verwendet werden kann, um festzulegen, wie oft **atime**-Aktualisierungen erfolgen. Stattdessen unterstützt GFS2 die Einhängeoptionen **relatime** und **noatime**. Die **relatime**-Einhängeoption wird empfohlen, um ein ähnliches Verhalten zu erreichen, wie mit dem **atime_quantum**-Parameter in GFS eingestellt werden kann.

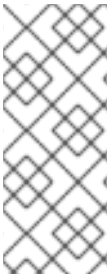
1.4.2.7. Die data= Option des mount-Befehls

Wenn Sie ein GFS2-Dateisystem einhängen, können Sie die Option **data=ordered** oder **data=writeback** des **mount**-Befehls angeben. Wenn **data=ordered** gesetzt ist, werden die Benutzerdaten, die von einer Transaktion verändert wurden, auf die Festplatte gespeichert, bevor die Transaktion auf die Festplatte festgeschrieben wird. Dies sollte verhindern, dass der Benutzer nach einem Absturz nicht initialisierte Blöcke in einer Datei sieht. Wenn **data=writeback** gesetzt ist, werden die Benutzerdaten zu einem beliebigen Zeitpunkt nach deren Änderung auf die Festplatte geschrieben. Dies garantiert zwar nicht dieselbe Konsistenz wie der **ordered**-Modus, sollte aber bei gewissen Workloads etwas schneller sein. Standardmäßig ist der **ordered**-Modus gesetzt.

1.4.2.8. Das gfs2_tool-Befehl

Der **gfs2_tool**-Befehl unterstützt eine andere Reihe von Optionen als der **gfs_tool**-Befehl für GFS:

- Der **gfs2_tool**-Befehl unterstützt einen **journals**-Parameter, der sämtliche Informationen über die derzeit konfigurierten Journale ausgibt, einschließlich der Anzahl an Journalen auf dem Dateisystem.
- Der **gfs2_tool**-Befehl unterstützt nicht das **counters**-Flag, das der **gfs_tool**-Befehl zum Anzeigen von GFS-Statistiken verwendet.
- Der **gfs2_tool**-Befehl unterstützt nicht das **inherit_jdata**-Flag. Um ein Verzeichnis als „inherit jdata“ zu kennzeichnen, können Sie das **jdata**-Flag auf dem Verzeichnis setzen oder Sie können den **chattr**-Befehl verwenden, um das Flag **+j** auf dem Verzeichnis zu setzen. Die Verwendung des **chattr**-Befehls ist die bevorzugte Methode, um das Daten-Journaling auf einer Datei zu aktivieren bzw. zu deaktivieren.



ANMERKUNG

Ab der Red Hat Enterprise Linux 6.2 Release unterstützt GFS2 den **tunegfs2**-Befehl, der einige der Funktionen des **gfs2_tool**-Befehls ersetzt. Weitere Informationen finden Sie auf der man-Seite von **tunegfs2**. Die Funktionen **settune** und **gettune** des **gfs2_tool**-Befehls wurden durch Befehlszeilenoptionen des **mount**-Befehls ersetzt, was es Ihnen ermöglicht, diese bei Bedarf mithilfe der **fstab**-Datei einzustellen.

1.4.2.9. Der gfs2_edit-Befehl

Der **gfs2_edit**-Befehl unterstützt eine andere Reihe von Optionen für GFS2 als der **gfs_edit**-Befehl für GFS. Weitere Informationen über die einzelnen Optionen, die jede Version des Befehls unterstützt, finden Sie auf den man-Seiten für **gfs2_edit** und **gfs_edit**.

1.4.3. Verbesserung der GFS2-Leistung

Viele Funktionen des GFS2-Dateisystems bringen zwar keine Veränderung an der Benutzerschnittstelle des GFS-Dateisystems mit sich, verbessern jedoch die Leistung des Dateisystems.

Ein GFS2-Dateisystem bietet in den folgenden Aspekten eine verbesserte Dateisystemleistung:

- Bessere Leistung bei intensiver Benutzung eines einzelnen Verzeichnisses
- Schnellere synchrone I/O-Operationen
- Schnelleres Lesen des Caches (kein Sperr-Mehraufwand)
- Schnellere direkte I/O mit bereits zugewiesenen Dateien (vorausgesetzt, der I/O-Umfang ist relativ groß, z. B. 4 M Blöcke)
- Allgemein schnellere I/O-Operationen
- Schnellere Ausführung des **df**-Befehls aufgrund von schnelleren **statfs**-Aufrufen
- Verbesserter **atime**-Modus, um die Anzahl der durch **atime** erzeugten Schreibvorgänge im Vergleich mit GFS zu verringern

GFS2-Dateisysteme bieten in den folgenden Punkten eine breitere und allgemeinere Unterstützung:

- GFS2 ist ein Teil des Upstream-Kernels (integriert in 2.16.19)
- GFS2 unterstützt die folgenden Funktionen:
  - erweiterte Dateiattribute (**`xattr`**)
  - die Attributeinstellungen **`lsattr`**() und **`chattr`**() über standardmäßige **`ioctl`**()-Aufrufe
  - Timestamps im Nanosekundenbereich

Ein GFS2-Dateisystem bietet folgende Verbesserungen für die Gesamteffizienz des Dateisystems:

- GFS2 verbraucht weniger Kernel-Speicher
- GFS2 braucht keine Metadaten-Generierungsnummern

  Das Zuweisen von GFS2-Metadaten erfordert keine Lesevorgänge. Kopien von Metadatenblöcken in mehreren Journalen werden gehandhabt, indem Blöcke vom Journal widerrufen werden, bevor die Sperrung aufgehoben wird.
- GFS2 beinhaltet eine deutlich einfachere Protokollverwaltung, die keine Kenntnis von nicht verlinkten Inodes oder Änderungen der Speicherkontingente hat.
- Die Befehle **`gfs2_grow`** und **`gfs2_jadd`** verwenden Sperren, um zu verhindern, dass mehrere Instanzen gleichzeitig ausgeführt werden.
- Der ACL-Code wurde für Aufrufe wie **`creat`**() und **`mkdir`**() vereinfacht.
- Nicht verlinkte Inodes, Änderungen der Festplattenkontingente und **`statfs`**-Änderungen werden wiederhergestellt, ohne dass das Journal neu eingehängt werden muss.

KAPITEL 2. ÜBERLEGUNGEN ZUR KONFIGURATION UND ZUM BETRIEB VON GFS2

Das GFS2-Dateisystem (Global File System 2) ermöglicht es mehreren Computern („Knoten“) in einem Cluster, kooperativ denselben Speicher zu verwenden. Um diese Zusammenarbeit zu erreichen und die Datenkonsistenz zwischen den Knoten aufrechtzuerhalten, verwenden die Knoten ein clusterweites Sperrschema für die Dateisystemressourcen. Dieses Sperrschema verwendet Kommunikationsprotokolle wie TCP/IP, um Sperrinformationen auszutauschen.

Sie können die Leistung verbessern, indem Sie den in diesem Kapitel beschriebenen Empfehlungen folgen, darunter Empfehlungen zur Erstellung, Verwendung und Pflege eines GFS2-Dateisystems.

WICHTIG

Stellen Sie sicher, dass Ihre Bereitstellung des Red Hat High Availability Add-Ons Ihren Anforderungen gerecht wird und unterstützt werden kann. Beratschlagen Sie sich dazu ggf. mit einem autorisierten Red Hat Vertreter, um Ihre Konfiguration vor der Bereitstellung zu prüfen.

2.1. ÜBERLEGUNGEN ZUR FORMATIERUNG

Dieser Abschnitt gibt Empfehlungen, wie Sie Ihr GFS2-Dateisystem formatieren sollten, um die Leistung zu optimieren.

2.1.1. Dateisystemgröße: Kleiner ist besser

GFS2 basiert auf einer 64-Bit-Architektur, die theoretisch ein 8 EB Dateisystem handhaben kann. Allerdings beträgt die derzeit maximal unterstützte Größe für ein GFS2-Dateisystem für 64-Bit-Hardware 100 TB. Die derzeit maximal unterstützte Größe für ein GFS2-Dateisystem für 32-Bit-Hardware beträgt 16 TB.

Beachten Sie, dass große Dateisysteme nicht unbedingt empfehlenswert sind, auch wenn diese mit GFS2 möglich sind. Als Faustregel gilt, dass kleine Dateisysteme mit GFS2 vorzuziehen sind: Es ist besser, 10 Dateisysteme mit 1 TB Kapazität zu haben als ein Dateisystem mit 10 TB Kapazität.

Es gibt mehrere Gründe, warum Sie Ihre GFS2-Dateisysteme klein halten sollten:

- Es wird weniger Zeit benötigt, um eine Sicherungskopie jedes Dateisystems zu erstellen.
- Es wird weniger Zeit benötigt, wenn Sie das Dateisystem mit dem Befehl **`fsck.gfs2`** überprüfen müssen.
- Es wird weniger Arbeitsspeicher benötigt, wenn Sie das Dateisystem mit dem Befehl **`fsck.gfs2`** überprüfen müssen.

Darüber hinaus bedeutet die Pflege von weniger Ressourcengruppen eine bessere Leistung.

Andererseits sollten Sie Ihr GFS2-Dateisystem nicht zu klein machen, da ungenügender Speicherplatz eine Reihe anderer Probleme mit sich bringt. Sie sollten sich über Ihre eigenen Anforderungen im Klaren sein, bevor Sie sich für eine bestimmte Größe entscheiden.

2.1.2. Blockgröße: Standardblöcke (4 K) werden bevorzugt

Ab der Red Hat Enterprise Linux 6 Release versucht der Befehl **`mkfs.gfs2`** eine optimale Blockgröße

basierend auf der Gerätetopologie zu schätzen. Im Allgemeinen ist 4 K die bevorzugte Blockgröße, da 4 K die standardmäßige Seitengröße (Arbeitsspeicher) für Linux ist. Im Gegensatz zu einigen anderen Dateisystemen führt GFS2 die meisten seiner Operationen mit 4-K-Kernel-Puffern durch. Wenn Ihre Blockgröße 4 K beträgt, hat der Kernel weniger Arbeit bei der Handhabung der Puffer.

Es wird empfohlen, dass Sie die Standardblockgröße verwenden, da diese die beste Leistung bringen sollte. Sie müssen Sie eine andere Blockgröße wahrscheinlich nur dann verwenden, wenn Sie eine effiziente Speicherung von vielen sehr kleinen Dateien benötigen.

2.1.3. Anzahl der Journale: Eines für jeden einhängenden Knoten

GFS2 erfordert ein Journal für jeden Knoten im Cluster, der das Dateisystem einhängen muss. Wenn Sie zum Beispiel einen 16-Knoten-Cluster haben, aber das Dateisystem von nur zwei Knoten eingehängt werden muss, benötigen Sie nur zwei Journale. Wenn Sie von einem dritten Knoten einhängen möchten, können Sie jederzeit ein Journal mit dem Befehl **`gfs2_jadd`** hinzufügen. Mit GFS2 können Sie jederzeit Journale im laufenden Betrieb hinzufügen.

2.1.4. Journalgröße: Standard (128 MB) ist in der Regel optimal

Wenn Sie den Befehl **`mkfs.gfs2`** ausführen, um ein GFS2-Dateisystem zu erstellen, können Sie die Größe der Journale festlegen. Wenn Sie keine Größe angeben, wird sie auf 128 MB festgelegt, was für die meisten Anwendungen optimal sein dürfte.

Einige Systemadministratoren könnten meinen, dass 128 MB zu groß ist, und versuchen, die Größe des Journals auf das Minimum von 8 MB oder auf konservativere 32 MB zu reduzieren. Dies könnte zwar funktionieren, kann sich jedoch stark auf den Durchsatz auswirken. Wie in vielen Journaling-Dateisystemen werden jedes Mal, wenn GFS2 Metadaten schreibt, die Metadaten in das Journal übergeben, bevor sie an ihren Platz geschrieben werden. Dadurch wird sichergestellt, dass im Falle eines Systemabsturzes oder Stromausfalls alle Metadaten wiederhergestellt werden, wenn das Journal automatisch beim Einhängen wieder eingespielt wird. Allerdings ist nicht viel Aktivität auf dem Dateisystem notwendig, um ein 8 MB kleines Journal zu füllen, und wenn das Journal voll ist, verlangsamt sich die Leistung, da GFS2 auf die Schreibvorgänge in den Speicher warten muss.

Es wird allgemein empfohlen, die standardmäßige Journalgröße von 128 MB zu verwenden. Wenn Ihr Dateisystem sehr klein ist (z. B. 5 GB), könnte ein 128 MB Journal unpraktisch sein. Wenn Sie ein größeres Dateisystem nutzen und den Platz dafür haben, könnten Journale mit 256 MB die Leistung verbessern.

2.1.5. Größe und Anzahl der Ressourcengruppen

Wenn ein GFS2-Dateisystem mit dem Befehl **`mkfs.gfs2`** erstellt wird, teilt es den Speicher in gleichmäßige Abschnitte, die Ressourcengruppen genannt werden. Dabei wird versucht, die optimale Größe für die Ressourcengruppen (zwischen 32 MB und 2 GB) abzuschätzen. Sie können diesen Standard mit der Option **`-r`** des Befehls **`mkfs.gfs2`** überschreiben.

Die optimale Größe für Ihre Ressourcengruppen hängt davon ab, wie Sie das Dateisystem verwenden werden. Bedenken Sie dabei, wie voll es sein wird und ob es stark fragmentiert sein wird oder nicht.

Sie sollten mit anderen Größen der Ressourcengruppen experimentieren, um zu sehen, welche Größe eine optimale Leistung erzielt. Eine bewährte Methode ist das Experimentieren mit einem Test-Cluster, bevor GFS2 für den Einsatz in der Produktionsumgebung bereitgestellt wird.

Wenn Ihr Dateisystem zu viele Ressourcengruppen hat (von denen jede zu klein ist), können die Blockzuweisungen zu viel Zeit mit der Suche von Zehntausenden (oder Hunderttausenden) von Ressourcengruppen für einen freien Block verschwenden. Je voller das Dateisystem, desto mehr

Ressourcengruppen müssen durchsucht werden, und jede von ihnen erfordert eine clusterweite Sperre. Dies führt zu einer verlangsamten Leistung.

Wenn Ihr Dateisystem jedoch zu wenige Ressourcengruppen hat (von denen jede zu groß ist), könnten Blockzuweisungen oft die Sperre der gleichen Ressourcengruppe verlangen, was ebenfalls Auswirkungen auf die Leistung hat. Wenn Sie beispielsweise ein 10 GB großes Dateisystem haben, das in fünf Ressourcengruppen zu je 2 GB aufgeteilt ist, werden die Knoten in Ihrem Cluster um diese fünf Ressourcengruppen öfter kämpfen, als wenn das gleiche Dateisystem in 320 Ressourcengruppen zu je 32 MB aufgeteilt wäre. Das Problem wird noch verschärft, wenn Ihr Dateisystem fast voll ist, da jede Blockzuweisung ggf. durch mehrere Ressourcengruppen suchen muss, bevor sie eine mit einem freien Block findet. GFS2 versucht dieses Problem auf zwei Arten zu verhindern:

- Erstens: Wenn eine Ressourcengruppe gänzlich voll ist, merkt sich GFS2 das und versucht, diese für spätere Zuweisungen nicht mehr zu prüfen (bis ein Block freigegeben wird). Wenn Sie niemals Dateien löschen, werden die Konflikte weniger schwerwiegend sein. Wenn Ihre Anwendung allerdings ständig auf einem fast vollen Dateisystem Blöcke löscht und neue Blöcke zuweist, werden die Konflikte sehr hoch sein, was sich stark auf die Leistung auswirken wird.

- Zweitens: Wenn neue Blöcke zu einer vorhandenen Datei hinzugefügt werden (z. B. durch Anhängen), wird GFS2 versuchen, die neuen Blöcke in derselben Ressourcengruppe wie die Datei zu gruppieren. Dies geschieht, um die Leistung zu erhöhen: Auf einer sich drehenden Festplatte wird die Suche weniger Zeit benötigen, wenn die Blöcke physisch nahe beisammen liegen.

Der schlimmste Fall ist ein zentrales Verzeichnis, in dem alle Knoten Dateien erstellen, da in diesem Fall alle Knoten ständig darum kämpfen werden, die gleiche Ressourcengruppe zu sperren.

2.2. DATEISYSTEMFRAGMENTIERUNG

Red Hat Enterprise Linux 6.4 führt Verbesserungen an der Fragmentierungsverwaltung in GFS2 ein. Unter Red Hat Enterprise Linux 6.4 führen gleichzeitige Schreibvorgänge zu weniger Fragmentierung und damit zu einer besseren Leistung für diese Workloads.

Zwar gibt es kein Defragmentierungstool für GFS2 auf Red Hat Enterprise Linux, aber Sie können einzelne Dateien defragmentieren, indem Sie diese mit dem filefrag-Dienstprogramm identifizieren, sie in temporäre Dateien kopieren und die temporären Dateien umbenennen, um die Originale zu ersetzen. (Dieses Verfahren kann auch in Versionen vor Red Hat Enterprise Linux 6.4 durchgeführt werden, sofern die Schreibvorgänge sequentiell erfolgen).

2.3. PROBLEME MIT DER BLOCKZUWEISUNG

Dieser Abschnitt enthält eine Übersicht über Probleme bei der Blockzuweisung in GFS2-Dateisystemen. Auch wenn Anwendungen, die lediglich Daten schreiben, in der Regel nicht beachten, wie oder wo ein Block zugewiesen wird, kann Ihnen ein wenig Wissen über die Funktionsweise der Blockzuweisung dabei helfen, die Leistung zu optimieren.

2.3.1. Freien Speicherplatz im Dateisystem belassen

Wenn ein GFS2-Dateisystem fast voll ist, wird es für die Blockzuweisung schwierig, Platz für die Zuweisung neuer Blöcke zu finden. Infolgedessen werden Blöcke oftmals an das Ende einer Ressourcengruppe gezwängt oder aber in sehr kleinen Scheiben, sodass eine Dateifragmentierung sehr viel wahrscheinlicher ist. Diese Dateifragmentierung kann Leistungsprobleme verursachen. Darüber hinaus verbringt die GFS2-Blockzuweisung auf einem fast vollen Dateisystem mehr Zeit damit, mehrere

Ressourcengruppen zu durchsuchen. Dies verursacht Sperrkonflikte, die auf einem Dateisystem mit ausreichend freiem Speicherplatz vermeidbar gewesen wären. Auch dies kann auch zu Leistungsproblemen führen.

Aus diesen Gründen wird empfohlen, kein Dateisystem auszuführen, das über 85 Prozent voll ist, wobei diese Zahl je nach Workload abweichen kann.

2.3.2. Dateizuweisung durch die jeweiligen Knoten, wenn möglich

Aufgrund der Funktionsweise des Distributed Lock Managers (DLM) treten mehr Sperrkonflikte auf, wenn alle Dateien von einem Knoten zugeordnet wurden und andere Knoten Blöcke zu diesen Dateien hinzufügen müssen.

In GFS (Version 1) wurden alle Sperren von einem zentralen Sperrmanager verwaltet, dessen Aufgabe es war, die Sperren im gesamten Cluster zu steuern. Dieser zentrale Sperrmanager (Grand Unified Lock Manager oder kurz GULM) war problematisch, da er einen Single-Point-of-Failure bildete. Das neue Sperrschema in GFS2 – DLM – verteilt die Sperren im gesamten Cluster. Wenn ein Knoten im Cluster ausfällt, werden die Sperren von den anderen Knoten wiederhergestellt.

Mit DLM wird der erste Knoten, der eine Ressource (z. B. eine Datei) sperrt, zum Besitzer („lock master“) der Sperre. Andere Knoten können diese Ressource zwar sperren, müssen jedoch zunächst den Besitzer der Sperre um Erlaubnis bitten. Jeder Knoten weiß, welche Sperren er besitzt, und jeder Knoten weiß, welchen anderen Knoten er eine Sperre geliehen hat. Das Erstellen einer Sperre auf dem Besitzerknoten ist viel schneller als Sperren auf einem anderen Knoten, der unterbrechen und den Besitzer der Sperre um Erlaubnis bitten muss.

Wie in vielen Dateisystemen versucht die GFS2-Zuweisung, Blöcke der selben Datei nahe beieinander zu platzieren, um die Bewegung der Festplattenköpfe zu reduzieren und die Leistung zu steigern. Ein Knoten, der Blöcke einer Datei zuweist, muss wahrscheinlich die gleichen Ressourcengruppen für die neuen Blöcke verwenden und sperren (es sei denn, alle Blöcke in der Ressourcengruppe werden bereits verwendet). Das Dateisystem wird schneller laufen, wenn der Besitzer der Sperre für die Ressourcengruppe, welche die Datei enthält, seine Datenblöcke zuordnet (das heißt, es ist schneller, wenn der Knoten, der zuerst die Datei öffnete, alle Schreibvorgänge von neuen Blöcken durchführt).

2.3.3. Vorabzuweisung, wenn möglich

Wenn Dateien vorab zugewiesen werden, können Blockzuweisungen vollständig vermieden werden und das Dateisystem kann effizienter ausgeführt werden. Neuere Versionen von GFS2 enthalten den Systemaufruf **`fallocate`** (1), mit dem Sie Datenblöcke vorab zuweisen.

2.4. CLUSTER-ÜBERLEGUNGEN

Beachten Sie bei der Festlegung der Anzahl der Knoten, die Ihr System enthalten wird, dass ein Kompromiss zwischen hoher Verfügbarkeit und Leistung geschlossen werden muss. Bei einer größeren Anzahl von Knoten wird es zunehmend schwierig, die Arbeitsbelastung zu skalieren. Aus diesem Grund unterstützt Red Hat die Verwendung von GFS2 nicht für Cluster-Dateisystembereitstellungen von mehr als 16 Knoten.

Die Implementierung eines Cluster-Dateisystems ist kein einfacher Ersatz für eine Bereitstellung mit einem einzelnen Knoten. Wir empfehlen, dass Sie einen Zeitraum von etwa 8–12 Wochen zum Testen auf neuen Installationen einplanen, um das System zu testen und sicherzustellen, dass es die erforderliche Leistung bringt. Während dieser Zeit können jegliche Leistungs- oder Funktionsprobleme gelöst werden, wobei Sie Fragen an das Red Hat Support-Team richten sollten.

Wir empfehlen, dass Kunden, die eine Bereitstellung von Clustern erwägen, Ihre Konfigurationen noch vor der Implementierung vom Red Hat Support untersuchen lassen, um mögliche spätere Support-Probleme zu vermeiden.

2.5. ÜBERLEGUNGEN ZUR VERWENDUNG

Dieser Abschnitt enthält generelle Empfehlungen zur Verwendung von GFS2

2.5.1. Einhängeoptionen: noatime und nodiratime

Es wird allgemein empfohlen, die GFS2-Dateisysteme mit den Optionen **`noatime`** und **`nodiratime`** einzuhängen. Dadurch wendet GFS2 weniger Zeit zur Aktualisierung der Festplatten-Inodes für jeden Zugriff auf.

2.5.2. DLM-Optimierungsoptionen: Erhöhen der DLM-Tabellengröße

DLM verwendet mehrere Tabellen, um Informationen über Sperren auf allen Knoten im Cluster zu verwalten, zu koordinieren und zu verteilen. Eine Erhöhung der DLM-Tabellengröße kann möglicherweise die Leistung steigern. In Red Hat Enterprise Linux 6.1 und höher wurden die Standardgrößen dieser Tabellen angehoben, Sie können diese jedoch auch manuell mit den folgenden Befehlen erhöhen:

```
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
```

Die durch diese Befehle vorgenommenen Änderungen sind nicht persistent und überdauern keinen Neustart. Deshalb müssen Sie diese Befehle zu einem der Startup-Skripte hinzufügen und vor dem Einhängen eines GFS2-Dateisystemes ausführen, andernfalls werden die Änderungen ohne jegliche Benachrichtigung ignoriert.

Detailliertere Informationen über GFS2-Knotensperren finden Sie in Abschnitt 2.9, »GFS2-Knotensperrung«.

2.5.3. VFS-Optimierungsoptionen: Recherchieren und Ausprobieren

Wie alle Linux-Dateisysteme befindet sich GFS2 auf einer Schicht namens virtuelles Dateisystem (virtual file system oder kurz VFS). Sie können die VFS-Ebene mithilfe des **`sysctl`**-Befehls optimieren, um die zugrunde liegende GFS2-Leistung zu verbessern. Zum Beispiel können die Werte für **`dirty_background_ratio`** und **`vfs_cache_pressure`** abhängig von Ihrer Situation angepasst werden. Um die aktuellen Werte abzurufen, verwenden Sie die folgenden Befehle:

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

Die folgenden Befehle passen die Werte an:

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

Sie können die Werte dieser Parameter dauerhaft durch Bearbeiten der Datei **`/etc/sysctl.conf`** ändern.

Um die optimalen Werte für Ihre Anwendungsfälle zu finden, recherchieren Sie die verschiedenen VFS-Optionen und probieren Sie diese vor der Bereitstellung in der Produktionsumgebung auf einem Test-Cluster aus.

2.5.4. SELinux: Vermeiden von SELinux auf GFS2

In den meisten Situationen wird aus Sicherheitsgründen dringend der Einsatz von Security Enhanced Linux (SELinux) empfohlen, dies wird jedoch für GFS2 nicht unterstützt. SELinux speichert Informationen zu jedem Objekt im Dateisystem mittels erweiterter Attribute. Das Lesen, Schreiben und die Pflege dieser erweiterten Attribute ist möglich, verlangsamt GFS2 allerdings erheblich. Sie müssen SELinux auf GFS2-Dateisystemen deaktivieren.

2.5.5. Einrichten von NFS auf GFS2

Durch die zusätzliche Komplexität des GFS2-Sperrsubsystems und dessen vernetzter Natur erfordert das Einrichten von NFS über GFS2 viele Vorsichtsmaßnahmen und sorgfältige Konfiguration. Dieser Abschnitt beschreibt die Einschränkungen, die Sie bei der Konfiguration eines NFS-Dienstes auf einem GFS2-Dateisystem berücksichtigen sollten.

WARNUNG

Falls das GFS2-Dateisystem per NFS exportiert wird und NFS-Client-Applikationen POSIX-Sperren verwenden, dann müssen Sie das Dateisystem mit der Option **`localflocks`** einhängen. Dadurch werden POSIX-Sperren von jedem Server dazu gezwungen, lokale sperren zu sein, d. h. nicht geclustert und unabhängig voneinander. (Es können eine Reihe von Problemen auftreten, falls GFS2 versuchen sollte, POSIX-Sperren von NFS auf den Knoten in einem Cluster zu implementieren.) Für Applikationen, die auf NFS-Clients laufen, bedeuten lokale POSIX-Sperren, dass zwei Clients gleichzeitig dieselbe Sperre halten können, falls die beiden Clients von verschiedenen Servern aus einhängen. Wenn alle Clients NFS nur von einem Server aus einhängen, dann stellt sich das Problem nicht, dass verschiedene Server unabhängig voneinander dieselbe Sperre vergeben. Wenn Sie nicht sicher sind, ob Sie Ihr Dateisystem mit der **`localflocks`**-Option einhängen sollen, dann sollten Sie die Option nicht verwenden. Es ist immer sicherer, die Sperren auf einer geclusterten Grundlage zu verwenden.

Zusätzlich zu den Überlegungen hinsichtlich der Sperren sollten Sie Folgendes bedenken, wenn Sie einen NFS-Dienst auf einem GFS2-Dateisystem konfigurieren.

- Red Hat unterstützt nur solche Konfigurationen des Red Hat High Availability Add-Ons, die NFSv3 mit Sperren in einer Aktiv/Passiv-Konfiguration mit den folgenden Eigenschaften verwenden:
  - Das zugrunde liegende Dateisystem ist ein GFS2-Dateisystem, das auf einem Cluster mit 2 bis 16 Knoten läuft.
  - Ein NFSv3-Server ist als ein Dienst definiert, der das gesamte GFS2-Dateisystem zu jeder Zeit von nur einem einzigen Cluster-Knoten exportiert.

- Der NFS-Server kann im Rahmen der Ausfallsicherung von einem Cluster-Knoten auf einen anderen wechseln (Aktiv/Passiv-Konfiguration).
- *Außer* über den NFS-Server ist keinerlei Zugriff auf das GFS2-Dateisystem gestattet. Das betrifft sowohl lokalen GFS2-Dateisystemzugriff als auch den Zugriff über Samba oder Clustered Samba.
- Es gibt keine NFS-Kontingentunterstützung auf dem System.

Diese Konfiguration bietet Hochverfügbarkeit für das Dateisystem und reduziert Ausfallzeiten, da ein ausgefallener Knoten nicht dazu führt, dass der **fsck**-Befehl ausgeführt werden muss, wenn der NFS-Server von einem Knoten auf einen anderen wechselt.

- Die NFS-Option **fsid =** ist zwingend erforderlich für NFS-Exporte auf GFS2.
- Falls Probleme mit Ihrem Cluster auftreten (falls z. B. der Cluster das Quorum verliert und das Fencing nicht erfolgreich ist), werden die geclusterten logischen Datenträger und das GFS2-Dateisystem eingefroren und der Zugriff ist erst wieder möglich, wenn der Cluster wieder ein Quorum erlangt. Sie sollten diese Möglichkeit bei der Entscheidung berücksichtigen, ob eine einfache Ausfallsicherungslösung, wie in diesem Verfahren beschrieben, für Ihr System geeignet ist.

2.5.6. Samba (SMB oder Windows) File Serving über GFS2

Ab der Red Hat Enterprise Linux 6.2 Release können Sie Samba (SMB oder Windows) File Serving von einem GFS2-Dateisystem mit CTDB verwenden, das Aktiv/Aktiv-Konfigurationen ermöglicht. Informationen über geclusterte Samba-Konfigurationen finden Sie im *Cluster-Administrationshandbuch*.

Ein zeitgleicher Zugriff von außerhalb Sambas auf die Daten auf der Samba-Freigabe wird nicht unterstützt. Es gibt derzeit keine Unterstützung für GFS2-Cluster-Leases, was das File Serving mit Samba verlangsamt.

2.6. DATENSICHERUNG

Es ist wichtig, regelmäßig Sicherungskopien Ihres GFS2-Dateisystems für den Notfall zu erstellen, unabhängig von der Größe des Dateisystems. Viele Systemadministratoren fühlen sich sicher, weil sie durch RAID, Multipath, Spiegelung, Snapshots und andere Formen der Redundanz geschützt sind, aber man kann nie sicher genug sein.

Das Anlegen einer Sicherungskopie eines Knotens oder einer Gruppe von Knoten kann problematisch sein, da dieser Vorgang für gewöhnlich das sequenzielle Lesen des gesamten Dateisystems erfordert. Wenn dies von einem einzelnen Knoten durchgeführt wird, behält dieser Knoten alle Daten im Cache, bis die anderen Knoten im Cluster beginnen, Sperren anzufordern. Die Durchführung dieser Art von Sicherungsprogramm im laufenden Betrieb des Clusters wirkt sich negativ auf die Leistung aus.

Durch ein Löschen der Caches, sobald die Datensicherung abgeschlossen ist, wird die Zeit reduziert, die von anderen Knoten benötigt wird, um wieder in den Besitz ihrer Cluster-Sperren/-Caches zu gelangen. Dies ist allerdings immer noch nicht optimal, da die anderen Knoten das Cachen der Daten, die sie vor Beginn der Datensicherung im Cache hatten, gestoppt haben. Sie können Caches mit dem folgenden Befehl löschen, nachdem die Datensicherung abgeschlossen ist:

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

Es ist schneller, wenn jeder Knoten im Cluster seine eigenen Dateien sichert, sodass diese Aufgabe auf die Knoten verteilt wird. Sie könnten dies auch mit einem Skript erreichen, das den **rsync**-Befehl auf knotenspezifische Verzeichnisse anwendet.

Der beste Weg zum Anlegen einer GFS2-Sicherungskopie ist das Erstellen eines Hardware-Snapshots auf dem SAN, um diesen Snapshot dann einem anderen System zur Verfügung zu stellen und dort zu sichern. Das Backup-System sollte den Snapshot mit **-o lockproto=lock_nolock** einhängen, da es nicht Teil eines Clusters sein wird.

2.7. HARDWARE-ÜBERLEGUNGEN

Sie sollten die folgenden Hardware-Aspekte beim Bereitstellen eines GFS2-Dateisystems berücksichtigen.

- Verwenden von qualitativ höherwertigen Speicherlösungen

  GFS2 kann auf billigeren Hardware-Lösungen für gemeinsam verwendeten Speicher arbeiten, wie iSCSI oder Fibre Channel over Ethernet (FCoE), aber Sie erreichen eine bessere Leistung, wenn Sie Speicher höherer Qualität mit größerer Caching-Kapazität anschaffen. Red Hat führt die meisten Qualitäts-, Fehler- und Leistungstests auf SAN-Speichersystemen mit Fibre-Channel-Verbindung durch. In der Regel ist es immer besser, etwas bereitstellen, das zuvor getestet wurde.

- Prüfen der Netzwerkausrüstung vor der Bereitstellung

  Eine qualitativ höherwertige und schnellere Netzwerkausrüstung führt dazu, dass auch die Cluster-Kommunikation und GFS2 schneller und zuverlässiger läuft. Allerdings müssen Sie nicht die teuerste Hardware kaufen. Einige der teuersten Netzwerk-Switches haben Probleme beim Transport von Multicast-Paketen, die für die Weitergabe von **fcntl**-Sperren (flocks) verwendet werden, während billigere Allerwelts-Netzwerk-Switches manchmal schneller und zuverlässiger sind. Im Allgemeinen empfiehlt es sich, die Hardware zu testen, bevor sie für den vollen Einsatz in einer Produktionsumgebung implementiert wird.

2.8. LEISTUNGSPROBLEME: MEHR INFORMATIONEN IM RED HAT KUNDENPORTAL

Informationen über bewährte Verfahren zur Bereitstellung und Aktualisierung von Red Hat Enterprise Linux Clustern unter Verwendung des High Availability Add-Ons und Red Hat Global File System 2 (GFS2) finden Sie im Artikel „Red Hat Enterprise Linux Cluster, High Availability, and GFS Deployment Best Practices“ im Red Hat Kundenportal unter https://access.redhat.com/site/articles/40051.

2.9. GFS2-KNOTENSPERRUNG

Um die beste Leistung aus einem GFS2-Dateisystem herauszuholen, ist es wichtig, einige Aspekte der zugrunde liegenden Theorie zu verstehen. Ein Ein-Knoten-Dateisystem wird zusammen mit einem Cache implementiert, um die Latenz der Plattenzugriffe für häufig aufgerufene Daten zu verringern. In Linux erfüllt der Seiten-Cache (und historisch der Puffer-Cache) diese Caching-Funktion.

Bei GFS2 besitzt jeder Knoten seinen eigenen Seiten-Cache, der einen Teil der Festplattendaten enthalten kann. GFS2 verwendet ein Sperrverfahren namens *Glocks* (ausgesprochen „Dschie-Locks“), um die Integrität der Caches auf allen Knoten zu bewahren. Das Glock-Untersystem liefert eine Cache-Verwaltungsfunktion, die unter Verwendung des *Distributed Lock Managers* (DLM) als zugrunde liegende Kommunikationsschicht implementiert ist.

Glocks schützen den Cache auf Inode-Basis, das heißt es gibt eine Sperre pro Inode, die zur Steuerung der Caching-Schicht verwendet wird. Wird das Glock im gemeinsamen Modus (DLM-Sperrmodus: PR) zugewiesen, so dürfen die Daten unter diesem Glock auf einem oder mehreren Knoten gleichzeitig gecacht werden, sodass alle Knoten lokalen Zugriff auf diese Daten haben.

Wird das Glock im exklusiven Modus (DLM-Sperrmodus: EX) zugewiesen, so darf nur ein einziger Knoten die Daten unter diesem Glock cachen. Dieser Modus wird von allen Operationen genutzt, die Daten verändern (z. B. der **`write`**-Systemaufruf).

Falls ein anderer Knoten ein Glock anfordert, das jedoch nicht sofort zugewiesen werden kann, dann sendet der DLM eine Nachricht an die Knoten, die derzeit das Glock halten und dadurch die Anfrage blockieren, damit diese ihre Sperren verwerfen. Das Verwerfen von Glocks kann (gemessen an den Maßstäben der meisten Dateisystemoperationen) ein langwieriger Vorgang sein. Das Verwerfen eines gemeinsam verwendeten Glocks erfordert lediglich, dass der Cache für ungültig erklärt wird (Invalidierung), was vergleichsweise schnell und proportional zur Menge der gecachten Daten ist.

Das Verwerfen eines exklusiven Glocks erfordert die Bereinigung der Protokolle und das Schreiben sämtlicher veränderter Daten auf die Festplatte, gefolgt von der Invalidierung des Caches wie beim gemeinsam verwendeten Glock.

Der Unterschied zwischen einem Ein-Knoten-Dateisystem und GFS2 besteht darin, dass ein Ein-Knoten-Dateisystem über einen einzigen Cache verfügt, während GFS2 über separate Caches auf jedem Knoten verfügt. In beiden Fällen ist die Latenz beim Zugriff auf gecachte Daten ähnlich groß, allerdings ist die Latenz beim Zugriff auf nicht gecachte Daten deutlich größer mit GFS2, wenn ein anderer Knoten bereits dieselben Daten gecacht hatte.

ANMERKUNG

Aufgrund der Art und Weise, wie Caching in GFS2 implementiert ist, erreichen Sie die beste Leistung in einer dieser beiden Fälle:

- Ein Inode wird auf allen Knoten nur zum Lesezugriff verwendet.
- Schreibzugriffe auf einen Inode erfolgen nur von einem einzigen Knoten aus.

Beachten Sie, dass das Einfügen und Entfernen von Einträgen aus einem Verzeichnis bei der Dateierstellung oder -löschung als Schreibvorgang in den Verzeichnis-Inode gilt.

Es ist möglich, diese Regel zu ignorieren, sofern dies nur selten geschieht. Wird diese Regel zu oft missachtet, sind erhebliche Leistungseinbußen die Folge.

Falls Sie **`mmap`**() für eine Datei auf einem GFS2-Dateisystem mit einem Lese/Schreib-Mapping anwenden, diese Datei jedoch nur lesen, zählt dies als Lesevorgang. Auf GFS dagegen zählt dies als Schreibvorgang, sodass GFS2 sehr viel skalierbarer ist mit **`mmap`**() I/O.

Falls Sie den Parameter **`noatime`** **`mount`** nicht setzen, führen Lesevorgänge auch zu Schreibvorgängen, um die Timestamps der Datei zu aktualisieren. Wir empfehlen im Allgemeinen, dass alle GFS2-Benutzer mit **`noatime`** einhängen sollten, sofern kein besonderer Grund für die Verwendung von **`atime`** vorliegt.

2.9.1. Probleme mit Posix-Sperren

Bei der Verwendung von Posix-Sperren sollten Sie die folgenden Aspekte berücksichtigen:

- Die Verwendung von Flocks ermöglicht eine schnellere Verarbeitung als die Verwendung von POSIX-Sperren.
- Programme, die POSIX-Sperren in GFS2 verwenden, sollten die **GETLK**-Funktion vermeiden, da sich in einer Cluster-Umgebung die Prozess-ID auf einen anderen Knoten im Cluster beziehen kann.

2.9.2. Leistungsoptimierung mit GFS2

Normalerweise ist es möglich, für eine problematische Applikation die Art und Weise anzupassen, wie diese ihre Daten speichert, um dadurch eine erheblich bessere Leistung zu erreichen.

Ein typisches Beispiel für eine problematische Applikation ist ein E-Mail Server. Diese verfügen oft über ein Spool-Verzeichnis, das Dateien für jeden Benutzer enthält (**mbox**) oder über ein Verzeichnis für jeden Benutzer, das eine Datei für jede Nachricht enthält (**maildir**). Wenn Anfragen über IMAP eingehen, wird idealerweise jedem Benutzer eine Affinität zu einem bestimmten Knoten zugewiesen. Auf diese Weise werden deren Anfragen zum Ansehen und Löschen von E-Mails tendenziell vom Cache auf diesem Knoten bedient. Falls dieser Knoten ausfällt, kann die Sitzung natürlich auf einem anderen Knoten neu gestartet werden.

Wenn E-Mail über SMTP eingeht, können die einzelnen Knoten so eingerichtet werden, dass sie die E-Mails eines bestimmten Benutzers standardmäßig an einen bestimmten Knoten weiterleiten. Falls dieser Knoten nicht läuft, kann der empfangende Knoten die Nachricht direkt im Mail-Spool-Verzeichnis des Benutzers speichern. Dieser Aufbau dient dazu, eine bestimmte Gruppen von Dateien im Normalfall vorrangig auf einem einzigen Knoten zwischenzuspeichern, erlaubt jedoch direkten Zugriff im Falle eines Knotenausfalls.

Dieser Aufbau nutzt den Seiten-Cache von GFS2 optimal aus und macht darüber hinaus Ausfälle für die Applikation (**imap** oder **smtp**) transparent.

Die Datensicherung (Backup) ist ebenfalls oft problematisch. Falls möglich, ist es sehr von Vorteil, das Working Set eines jeden Knotens direkt von dem Knoten zu sichern, der diese bestimmte Gruppe von Inodes cacht. Falls Sie zur Datensicherung ein Skript nutzen, das regelmäßig zu einem bestimmten Zeitpunkt ausgeführt wird, und dieser Zeitpunkt mit einer Spitze in der Reaktionszeit einer auf GFS2 laufenden Applikation zusammenfällt, dann deutet dies mit ziemlicher Sicherheit darauf hin, dass der Cluster den Seiten-Cache nicht effizient genug verwendet.

Falls Sie sich in der glücklichen Position befinden, die Applikation zur Datensicherung stoppen zu können, stellt dies natürlich kein Problem dar. Wird die Datensicherung nur von einem Knoten durchgeführt, wird andererseits nach Abschluss der Sicherung ein großer Teil des Dateisystems auf diesem Knoten gecacht, was Leistungseinbußen für nachfolgende Zugriffe von anderen Knoten zur Folge hat. Bis zu einem gewissen Grad kann dies vermieden werden, indem Sie nach Abschluss der Datensicherung den VFS-Seiten-Cache auf dem Backup-Knoten mit dem folgenden Befehl bereinigen:

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

Eine bessere Lösung besteht jedoch darin, sicherzustellen, dass das Working Set auf jedem Knoten gemeinsam hauptsächlich für Lesezugriffe im Cluster verwendet wird oder dass weitgehend nur von einem einzigen Knoten darauf zugegriffen wird.

2.9.3. Suche und Bereinigung von Problemen bei der GFS2-Leistung mit GFS2 Lock Dump

Falls die Leistung Ihres Clusters unter ineffizienter Nutzung des GFS2-Cachings leidet, bemerken Sie möglicherweise zunehmend große I/O-Wartezeiten. Sie können die Informationen des GFS2 Lock Dump nutzen, um der Ursache des Problems auf den Grund zu gehen.

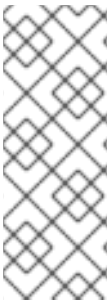
Dieser Abschnitt enthält einen Überblick über den GFS2 Lock Dump. Eine vollständige Beschreibung des GFS2 Lock Dumps finden Sie in Anhang C, *GFS2-Tracepoints und die debugfs-Glocks-Datei*.

Die Informationen des GFS2 Lock Dumps können aus der **debugfs**-Datei entnommen werden, die sich im folgenden Pfad befindet, sofern **debugfs** unter **/sys/kernel/debug/** eingehängt ist:

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

Die Datei enthält eine Reihe von Zeilen. Jede Zeile, die mit einem G: beginnt, steht für einen Glock und die folgenden Zeilen – um ein Leerzeichen eingerückt – stehen für eine Information, die sich auf das in der Datei darüberstehende Glock bezieht.

Erstellen Sie am besten mithilfe des **cat**-Befehls eine Kopie des gesamten Inhalts der **debugfs**-Datei (dies kann eine längere Zeit dauern, falls Sie eine große Menge RAM und zahlreiche gecachte Inodes haben), während bei der Applikation Probleme auftreten, und schauen Sie sich die so erstellten Daten zu einem späteren Zeitpunkt an.



ANMERKUNG

Es kann hilfreich sein, zwei Kopien der **debugfs**-Datei zu erstellen, im Abstand von ein paar Sekunden oder ein bis zwei Minuten. Vergleichen Sie in den zwei Kopien nun die Halterinformationen derselben Glock-Nummer. Anhand dessen sollten Sie feststellen können, ob die Verarbeitung Fortschritte macht (also nur langsam ist) oder ob sie hängen geblieben ist (was immer auf einen Fehler hindeutet und umgehend dem Red Hat Support gemeldet werden sollte).

Jede Zeile in der **debugfs**-Datei, die mit H: (Halter) beginnt, steht für eine Sperranfrage, die entweder bereits gewährt wurde oder darauf wartet, gewährt zu werden. Das Flags-Feld auf der Halterzeile f: zeigt an, was von beidem der Fall ist: Das „W“-Flag kennzeichnet eine wartende Anfrage, das „H“-Flag kennzeichnet eine gewährte Anfrage. Diejenigen Glocks mit einer hohen Anzahl wartender Anfragen sind wahrscheinlich diejenigen mit Konflikten.

Tabelle 2.1, »Glock-Flags« zeigt die Bedeutungen der verschiedenen Glock-Flags und Tabelle 2.2, »Glock-Halter-Flags« zeigt die Bedeutungen der verschiedenen Glock-Halter-Flags in der Reihenfolge, in der Sie in den Glock-Dumps auftreten.

Tabelle 2.1. Glock-Flags

Flag	Name	Bedeutung
b	Blocking	Gültig, wenn das „locked“-Flag gesetzt ist. Zeigt an, dass die Operation, die vom DLM angefordert wurde, sperren könnte. Dieses Flag wird für „demote“-Operationen und für „try“-Sperren entfernt. Der Zweck dieses Flags ist es, das Sammeln von Statistiken über die DLM-Antwortzeiten zu ermöglichen, unabhängig von der Zeit, die andere Knoten zum Herabstufen von Sperren benötigen.
d	Pending demote	Eine wartende Anfrage zum Herabstufen (remote)

Flag	Name	Bedeutung
D	Demote	Eine Anfrage zum Herabstufen (lokal oder remote)
f	Log flush	Das Protokoll muss festgeschrieben werden, bevor dieses Glock freigegeben werden kann
F	Frozen	Antworten von Remote-Knoten werden ignoriert, eine Wiederherstellung läuft. Dieses Flag hat nichts mit dem Dateisystem-Freeze zu tun, das einen anderen Mechanismus verwendet, sondern wird nur zur Wiederherstellung verwendet.
i	Invalidate in progress	Seiten unter diesem Glock werden derzeit ungültig gemacht (invalidiert)
I	Initial	Gesetzt, wenn eine DLM-Sperre mit diesem Glock verknüpft ist
l	Locked	Das Glock ändert derzeit seinen Status
L	LRU	Gesetzt, wenn das Glock auf der LRU-Liste ist
o	Object	Gesetzt, wenn das Glock einem Objekt zugeordnet ist (d. h. einem Inode für Typ-2-Glocks und einer Ressourcengruppe für Typ-3-Glocks)
p	Demote in progress	Das Glock antwortet derzeit auf eine Anfrage zum Herabstufen
q	Queued	Gesetzt, wenn ein Halter der Warteschlange eines Glocks hinzugefügt wird, und gelöscht, wenn das Glock noch gehalten wird, es jedoch keine verbleibenden Halter gibt. Verwendet als Teil des Algorithmus, der die minimale Haltezeit für ein Glock berechnet.
r	Reply pending	Von Remote-Knoten erhaltene Antwort wartet auf Verarbeitung
y	Dirty	Daten müssen auf die Festplatte überschrieben werden, bevor dieses Glock freigegeben werden kann

Tabelle 2.2. Glock-Halter-Flags

Flag	Name	Bedeutung
a	Async	Nicht auf das Glock-Ergebnis warten (Ergebnis wird später abgerufen)
A	Any	Jeder kompatible Sperrmodus ist zulässig
c	No cache	Wenn nicht gesperrt, sofort DLM-Sperre herabstufen

Flag	Name	Bedeutung
e	No expire	Nachfolgende Anfragen zur Aufhebung der Sperre ignorieren
E	exact	Muss den exakten Sperrmodus haben
F	First	Gesetzt, wenn der Halter der Erste ist, dem diese Sperre gewährt wird
H	Holder	Zeigt an, dass die angeforderte Sperre gewährt wird
p	Priority	Reiht Halter an der Spitze der Warteschlange ein
t	Try	Eine „try“-Sperre
T	Try 1CB	Eine „try“-Sperre, die einen Callback sendet
W	Wait	Gesetzt, während auf den Abschluss einer Anfrage gewartet wird

Nachdem Sie herausgefunden haben, welches Glock das Problem verursacht, müssen Sie nun feststellen, auf welchen Inode es sich bezieht. Dies wird angezeigt durch die Glock-Nummer (n: auf der Zeile G:) im Format *type*/*number*. Falls *type* 2 ist, dann ist das Glock ein Inode-Glock und die *number* ist eine Inode-Nummer. Um den Inode zu finden, können Sie den Befehl **`find -inum number`** ausführen, wobei *number* die Inode-Nummer ist, die aus dem Hexadezimalformat in der Glocks-Datei in ein Dezimalformat konvertiert wurde.

ANMERKUNG

Falls Sie den **`find`**-Befehl auf einem Dateisystem durchführen, während dort Sperrkonflikte auftreten, werden Sie wahrscheinlich das Problem dadurch noch verschlimmern. Wenn Sie nach Inode-Konflikten suchen, ist es ratsam, die Applikation vor der Ausführung des **`find`**-Befehls zu stoppen.

Tabelle 2.3, »Glock-Typen« zeigt die Bedeutungen der verschiedenen Glock-Typen.

Tabelle 2.3. Glock-Typen

Typnummer	Sperrtyp	Verwendung
1	Trans	Transaktionssperre
2	Inode	Inode-Metadaten und -Daten
3	Rgrp	Ressourcengruppen-Metadaten
4	Meta	Der Superblock

Typnummer	Sperrtyp	Verwendung
5	Iopen	Feststellung des letzten Schließers des Inodes
6	Flock	**flock**(2)-Systemaufruf
8	Quota	Kontingentoperationen
9	Journal	Journal-Mutex

Falls das identifizierte Glock einem anderen Typ angehört, dann am ehesten Typ 3: (Ressourcengruppe). Falls Sie unter normaler Auslastung eine erhebliche Anzahl von Prozessen sehen, die auf andere Glock-Typen warten, dann melden Sie dies bitte dem Red Hat Support.

Falls Sie eine Reihe von Anfragen in der Warteschlange sehen, die auf eine Ressourcengruppensperre warten, könnte dies eine Reihe von Ursachen haben. Zum einen gibt es unter Umständen eine hohe Anzahl von Knoten im Vergleich zur Anzahl von Ressourcengruppen im Dateisystem. Zum anderen könnte das Dateisystem unter Umständen beinahe voll sein (wodurch die Suche nach freien Blöcken durchschnittlich länger braucht). In beiden Fällen kann die Situation verbessert werden, indem mehr Speicher hinzugefügt wird und das Dateisystem mithilfe des Befehls **gfs2_grow** vergrößert wird.

KAPITEL 3. ERSTE SCHRITTE

Dieses Kapitel beschreibt, wie Sie bei der Erstinstallation von GFS2 vorgehen müssen und beinhaltet folgende Abschnitte:

- Abschnitt 3.1, »Grundlegende Vorbereitungen«
- Abschnitt 3.2, »Schritte zur erstmaligen Einrichtung«

3.1. GRUNDLEGENDE VORBEREITUNGEN

Bevor Sie Red Hat GFS2 einrichten, sollten Sie die folgenden Schritte durchführen:

- Stellen Sie sicher, dass Sie sich die wesentlichen Charakteristiken der GFS2-Knoten notiert haben (siehe Abschnitt 1.2, »Vor der Einrichtung von GFS2«).
- Stellen Sie sicher, dass die Systemuhren auf den GFS2-Knoten synchronisiert sind. Es wird empfohlen, dass Sie die Network Time Protocol (NTP) Software verwenden, die in Ihrer Red Hat Enterprise Linux Distribution integriert ist.

  ANMERKUNG

  Die Systemuhren der GFS2-Knoten dürfen nur ein paar Minuten voneinander abweichen, um unnötiges Aktualisieren der Inode-Timestamps zu vermeiden. Unnötige Aktualisierungen der Inode-Timestamps haben schwerwiegende Auswirkungen auf die Leistung des Clusters.

- Um GFS2 in einer Cluster-Umgebung zu verwenden, müssen Sie Ihr System zur Verwendung des Clustered Logical Volume Manager (CLVM) konfigurieren, der eine Reihe von Clustering-Erweiterungen für den LVM Logical Volume Manager bereitstellt. Um CLVM verwenden zu können, muss die Red Hat Cluster Suite Software laufen, einschließlich dem **`clvmd`**-Daemon. Informationen zur Verwendung von CLVM finden Sie im Handbuch *Administration des Logical Volume Manager*. Informationen zur Installation und Verwaltung der Red Hat Cluster Suite finden Sie im Handbuch *Cluster-Administration*.

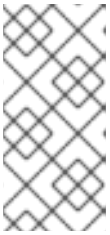
3.2. SCHRITTE ZUR ERSTMALIGEN EINRICHTUNG

Die erstmalige Einrichtung von GFS2 umfasst die folgenden Schritte:

1. Einrichten von logischen Datenträgern
2. Erstellen der GFS2-Dateisysteme
3. Einhängen der Dateisysteme

Führen Sie die folgenden Schritte aus, um GFS2 zum ersten Mal einzurichten.

1. Verwenden Sie LVM, um einen logischen Datenträger für jedes Red Hat GFS2-Dateisystem zu erstellen.



ANMERKUNG

Sie können die in der Red Hat Cluster Suite enthaltenen **`init.d`**-Skripte verwenden, um das Aktivieren und Deaktivieren der logischen Datenträger zu automatisieren. Weitere Informationen über **`init.d`**-Skripte finden Sie unter *Konfiguration und Verwaltung eines Red Hat Clusters*.

2. Erzeugen Sie auf den in Schritt 1 erstellten logischen Datenträgern GFS2-Dateisysteme. Wählen Sie einen eindeutigen Namen für jedes Dateisystem. Weitere Informationen über das Erstellen von GFS2-Dateisystemen finden Sie in Abschnitt 4.1, »Erstellen eines Dateisystems«.

   Sie können eines der folgenden Befehlsformate verwenden, um ein geclustertes GFS2-Dateisystem zu erstellen:

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

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

   Weitere Informationen über das Erstellen von GFS2-Dateisystemen finden Sie in Abschnitt 4.1, »Erstellen eines Dateisystems«.

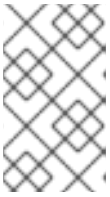
3. Hängen Sie auf jedem Knoten die GFS2-Dateisysteme ein. Weitere Informationen über das Einhängen von GFS2-Dateisystemen finden Sie in Abschnitt 4.2, »Einhängen eines Dateisystems«.

   Befehlsverwendung:

   `mount BlockDevice MountPoint`

   `mount -o acl BlockDevice MountPoint`

   Die Einhängeoption **`-o acl`** erlaubt die Veränderung der Datei-ACLs. Wird ein Dateisystem ohne die Einhängeoption **`-o acl`** eingehängt, können Benutzer die ACLs zwar einsehen (mittels **`getfacl`**), dürfen diese jedoch nicht verändern (mittels **`setfacl`**).



ANMERKUNG

Sie können die im Red Hat High Availability Add-On enthaltenen **`init.d`**-Skripte verwenden, um das Ein- und Aushängen von GFS2-Dateisystemen zu automatisieren.

KAPITEL 4. VERWALTUNG VON GFS2

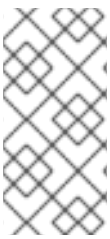
Dieses Kapitel beschreibt die Aufgaben und Befehle zur Verwaltung von GFS2 und umfasst die folgenden Abschnitte:

4.1. ERSTELLEN EINES DATEISYSTEMS

Mit dem **mkfs.gfs2**-Befehl erstellen Sie ein GFS2-Dateisystem. Sie können dazu auch den **mkfs**-Befehl mit der Option **-t gfs2** verwenden. Ein Dateisystem wird auf einem aktivierten LVM-Datenträger erstellt. Folgende Informationen sind erforderlich, um den **mkfs.gfs2**-Befehl auszuführen:

- Sperrprotokoll/Modulname (das Sperrprotokoll für einen Cluster ist **lock_dlm**)
- Clustername (falls Teil einer Cluster-Konfiguration)
- Anzahl der Journale (jeweils ein Journal ist erforderlich für jeden Knoten, der das Dateisystem einhängen soll)

Beim Erstellen eines GFS2-Dateisystems können Sie den **mkfs.gfs2**-Befehl direkt verwenden oder Sie können den **mkfs**-Befehl mit dem Parameter **-t** verwenden, um den Dateisystemtyp **gfs2** anzugeben, gefolgt von den GFS2-Dateisystemoptionen.



ANMERKUNG

Sobald Sie mit dem **mkfs.gfs2**-Befehl ein GFS2-Dateisystem erstellt haben, können Sie die Größe dieses Dateisystems nicht mehr nach unten korrigieren. Sie können ein vorhandenes Dateisystem jedoch mithilfe des Befehls **gfs2_grow** vergrößern, wie in Abschnitt 4.6, »Vergrößern eines Dateisystems« beschrieben.

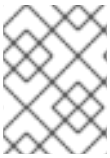
4.1.1. Verwendung

Beim Erstellen eines Cluster-GFS2-Dateisystems können Sie eines der folgenden Formate verwenden:

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

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

Beim Erstellen eines lokalen GFS2-Dateisystems können Sie eines der folgenden Formate verwenden:



ANMERKUNG

Für die Red Hat Enterprise Linux 6 Release unterstützt Red Hat nicht länger den Einsatz von GFS2 als Ein-Knoten-Dateisystem.

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

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

WARNUNG

Sie sollten sich unbedingt mit der Verwendung der Parameter ***LockProtoName*** und ***LockTableName*** vertraut machen. Eine unsachgemäße Verwendung der Parameter ***LockProtoName*** und ***LockTableName*** kann zur Beschädigung des Dateisystems oder des Lock Space führen.

LockProtoName

Gibt den Namen des zu verwendenden Sperrprotokolls an. Das Sperrprotokoll für einen Cluster ist **lock_dlm**.

LockTableName

Dieser Parameter wird für ein GFS2-Dateisystem in einer Cluster-Konfiguration spezifiziert. Er besteht aus zwei Teilen, durch einen Doppelpunkt voneinander getrennt (ohne Leerstellen), also: ***ClusterName:FSName***

- ***ClusterName***, der Name des Clusters, für den das GFS2-Dateisystem erstellt wird.
- ***FSName***, der Dateisystemname, darf zwischen 1 und 16 Zeichen lang sein. Der Name muss eindeutig für alle **lock_dlm**-Dateisysteme im Cluster sein und für alle Dateisysteme (**lock_dlm** und **lock_nolock**) auf jedem lokalen Knoten.

Number

Gibt die Anzahl der Journale an, die vom **mkfs.gfs2**-Befehl erzeugt werden sollen. Es wird jeweils ein Journal für jeden Knoten benötigt, der das Dateisystem einhängt. Für GFS2-Dateisysteme

können später mehr Journale hinzugefügt werden, ohne dass das Dateisystem vergrößert werden muss, wie im Abschnitt 4.7, »Hinzufügen von Journalen zu einem Dateisystem« beschrieben.

BlockDevice

Gibt einen logischen oder physischen Datenträger an.

4.1.2. Beispiele

In diesen Beispielen ist **lock_dlm** das vom Dateisystem verwendete Sperrprotokoll, da es sich hierbei um ein Cluster-Dateisystem handelt. Der Cluster-Name ist **alpha** und der Dateisystemname ist **mydata1**. Das Dateisystem beinhaltet 8 Journale und wird auf **/dev/vg01/lvol0** erstellt.

```
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
```

In diesen Beispielen wird ein zweites **lock_dlm**-Dateisystem erstellt, das im Cluster **alpha** verwendet werden kann. Der Dateisystemname lautet **mydata2**. Das Dateisystem beinhaltet 8 Journale und wird auf **/dev/vg01/lvol1** erstellt.

```
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
```

4.1.3. Vollständige Optionen

Tabelle 4.1, »Befehlsoptionen: **mkfs.gfs2**« beschreibt die **mkfs.gfs2**-Befehlsoptionen (Flags und Parameter).

Tabelle 4.1. Befehlsoptionen: mkfs.gfs2

Flag	Parameter	Beschreibung
-c	***Megabytes***	Legt die anfängliche Größe der Kontingentdatei aller Journal auf ***Megabytes*** fest.
-D		Aktiviert Ddebugging-Ausgabe.
-h		Hilfe. Zeigt verfügbare Optionen an.
-J	***MegaBytes***	Gibt die Größe des Journals in Megabytes an. Die standardmäßige Journalgröße beträgt 128 Megabytes, die minimale Größe ist 8 Megabytes. Größere Journale verbessern die Leistung, allerdings verbrauchen sie mehr Speicher als kleinere Journale.

Flag	Parameter	Beschreibung
-j	***Number***	Gibt die Anzahl der Journale an, die vom **mkfs.gfs2**-Befehl erzeugt werden. Es wird jeweils ein Journal für jeden Knoten benötigt, der das Dateisystem einhängt. Falls diese Option nicht spezifiziert ist, wird 1 Journal angelegt. Im GFS2-Dateisystem können Sie später Journale hinzufügen, ohne dass das Dateisystem vergrößert werden muss.
-O		Verhindert, dass der **mkfs.gfs2**-Befehl nach einer Bestätigung fragt, bevor das Dateisystem geschrieben wird.
-p	***LockProtoName***	Gibt den Namen des zu verwendenden Sperrprotokolls an. Zulässige Sperrprotokolle sind: **lock_dlm** – das standardmäßige Sperrmodul, erforderlich für ein geclustertes Dateisystem. **lock_nolock** – wird benutzt, wenn GFS2 als lokales Dateisystem fungiert (auf nur einem Knoten).
-q		Es wird keinerlei Ausgabe angezeigt.
-r	***MegaBytes***	Gibt die Größe der Ressourcengruppen in Megabytes an. Die minimale Ressourcengruppengröße beträgt 32 MB, die maximale Ressourcengruppengröße 2048 MB. Eine große Ressourcengruppengröße kann unter Umständen die Leistung von sehr großen Dateisystemen steigern. Falls dies nicht spezifiziert wird, wählt mkfs.gfs2 die Ressourcengruppengröße auf Basis der Dateisystemgröße aus: Durchschnittlich große Dateisysteme erhalten 256 MB große Ressourcengruppen und größere Dateisysteme erhalten für eine bessere Leistung größere Ressourcengruppen.

Flag	Parameter	Beschreibung
-t	***LockTableName***	Eine eindeutige Kennung, die das Sperrtabellenfeld spezifiziert, wenn Sie das **lock_dlm**-Protokoll verwenden; das **lock_nolock**-Protokoll verwendet diesen Parameter nicht. Dieser Parameter besteht aus zwei Teilen, durch einen Doppelpunkt (keine Leertaste) voneinander getrennt, wie z. B. ***ClusterName:FSName***. ***ClusterName*** ist der Name des Clusters, für den das GFS2-Dateisystem erzeugt wird; nur Mitgliedern des Clusters ist es erlaubt, dieses Dateisystem zu benutzen. Der Clustername wird in der **/etc/cluster/cluster.conf** Datei mithilfe des **Cluster Configuration Tool** festgelegt, und wird im **Cluster Status Tool** in der Cluster-Verwaltungsoberfläche der Red Hat Cluster Suite angezeigt. ***FSName***, der Dateisystemsname kann 1 bis 16 Zeichen lang sein. Der Name muss eindeutig im gesamten Dateisystem des Cluster sein.
-u	***MegaBytes***	Gibt die anfängliche Größe der nicht verlinkten Tag-Datei jedes Journals an.
-V		Zeigt Informationen zur Befehlsversion an.

4.2. EINHÄNGEN EINES DATEISYSTEMS

Bevor Sie ein GFS2-Dateisystem einhängen können, muss das Dateisystem existieren (siehe Abschnitt 4.1, »Erstellen eines Dateisystems«), der Datenträger, auf dem das Dateisystem existiert, muss aktiviert sein und die unterstützenden Clustering- und Sperrsysteme müssen gestartet sein (siehe *Konfiguration und Verwaltung eines Red Hat Clusters*). Nachdem dies sichergestellt wurde, können Sie das GFS2-Dateisystem genauso einhängen, wie Sie es von jedem anderen Linux-Dateisystem gewohnt sind.

ANMERKUNG

Sollten Sie versuchen, ein GFS2-Dateisystem einzuhängen, wenn der Cluster Manager (**cman**) nicht gestartet wurde, wird die folgende Fehlermeldung ausgegeben:

```
[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
```

Um die Datei-ACLs zu verändern, müssen Sie das Dateisystem mit der Einhängeoption **-o acl** einhängen. Wird ein Dateisystem ohne die Einhängeoption **-o acl** eingehängt, können Benutzer die ACLs zwar einsehen (mittels **getfacl**), dürfen diese jedoch nicht verändern (mittels **setfacl**).

4.2.1. Verwendung

Einhängen ohne ACL-Veränderung

```
mount BlockDevice MountPoint
```

Einhängen mit ACL-Veränderung

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

-o acl

GFS2-spezifische Option, um Veränderungen an Datei-ACLs zu erlauben.

BlockDevice

Gibt das Blockgerät an, auf dem sich das GFS2-Dateisystem befindet.

MountPoint

Gibt das Verzeichnis an, in dem das GFS2-Dateisystem eingehängt werden soll.

4.2.2. Beispiel

In diesem Beispiel ist das GFS2-Dateisystem auf **/dev/vg01/lvol0** im **/mygfs2**-Verzeichnis eingehängt.

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

4.2.3. Vollständige Verwendung

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

Der Parameter **-o option** enthält GFS2-spezifische Optionen (siehe Tabelle 4.2, »GFS2-spezifische Einhängeoptionen«) oder zulässige, standardmäßige Linux-Optionen für **mount -o** oder einer Kombination aus beiden. Mehrere ***option***-Parameter werden durch ein Komma getrennt, ohne Leerzeichen.

ANMERKUNG

Der **mount**-Befehl ist ein Linux-Systembefehl. Zusätzlich zu den in diesem Abschnitt beschriebenen GFS2-spezifischen Optionen können Sie weitere standardmäßige **mount**-Befehlsoptionen (z. B. **-r**) verwenden. Weitere Informationen über andere **mount**-Befehlsoptionen für Linux finden Sie auf der man-Seite für **mount**.

Tabelle 4.2, »GFS2-spezifische Einhängeoptionen« beschreibt die verfügbaren GFS2-spezifischen Werte für **-o option**, die zum Zeitpunkt des Einhängens an GFS2 übergeben werden können.

ANMERKUNG

Diese Tabelle enthält Beschreibungen der Optionen, die ausschließlich mit lokalen Dateisystemen verwendet werden. Beachten Sie jedoch, dass Red Hat für die Red Hat Enterprise Linux 6 Release den Einsatz von GFS2 nicht für ein Ein-Knoten-System unterstützt. Red Hat unterstützt auch weiterhin Ein-Knoten-GFS2-Dateisysteme zum Einhängen von Snapshots von Cluster-Dateisystemen (z. B. zwecks Datensicherung).

Tabelle 4.2. GFS2-spezifische Einhängeoptionen

Optionen	Beschreibung
acl	Erlaubt das Verändern von Datei-ACLs. Wird ein Dateisystem ohne die Einhängeoption **acl** eingehängt, können Benutzer die ACLs zwar einsehen (mittels **getfacl**), dürfen diese jedoch nicht verändern (mittels **setfacl**).
data=[ordered|writeback]	Wenn **data=ordered** gesetzt ist, werden die Benutzerdaten, die von einer Transaktion verändert wurden, auf die Festplatte gespeichert, bevor die Transaktion auf die Festplatte festgeschrieben wird. Dies sollte verhindern, dass der Benutzer nach einem Absturz in einer Datei nicht initialisierte Blöcke sieht. Wenn dagegen **data=writeback** gesetzt ist, werden die Benutzerdaten zu einem beliebigen Zeitpunkt nach deren Änderung auf die Festplatte geschrieben. Dies garantiert zwar nicht dieselbe Konsistenz wie der **ordered**-Modus, sollte aber bei einer gewissen Auslastung etwas schneller sein. Standardmäßig ist der **ordered**-Modus gesetzt.
ignore_local_fs **Achtung:** Diese Option sollte *nicht* eingesetzt werden, wenn GFS2-Dateisysteme gemeinsam verwendet werden.	Zwingt GFS2 dazu, das Dateisystem als Multihost-Dateisystem zu behandeln. Standardmäßig schaltet **lock_nolock** automatisch das **localflocks**-Flag ein.
localflocks **Achtung:** Diese Option sollte nicht eingesetzt werden, wenn GFS2-Dateisysteme gemeinsam verwendet werden.	Teilt GFS2 mit, dass die VFS-Schicht (Virtual File System) alle flocks und fcntl ausführen soll. Das **localflocks**-Flag wird automatisch durch **lock_nolock** eingeschaltet.
lockproto=*LockModuleName*	Erlaubt dem Benutzer die Angabe, welches Sperrprotokoll mit dem Dateisystem verwendet werden soll. Falls ***LockModuleName*** nicht spezifiziert wurde, wird der Name des Sperrprotokolls vom Dateisystem-Superblock gelesen.
locktable=*LockTableName*	Erlaubt dem Benutzer die Angabe, welche Sperrtabelle vom Dateisystem verwendet werden soll.

Optionen	Beschreibung
quota=[off/account/on]	Aktiviert oder deaktiviert Festplattenkontingente für das Dateisystem. Werden die Festplattenkontingente in den **account**-Status gesetzt, führt dies dazu, dass die Verbrauchsstatistiken pro UID/GID korrekt vom Dateisystem gepflegt werden; Grenzen und Warnwerte werden ignoriert. Der Standardwert ist **off**.
errors=panic|withdraw	Wenn **errors=panic** spezifiziert ist, führen Fehler im Dateisystem zu einer Kernel-Panik. Standardmäßig (was der Einstellung von **errors=withdraw** entspricht) zieht sich das System aus dem Dateisystem zurück und macht es bis zum nächsten Neustart nicht erreichbar; in einigen Fällen kann das System weiterhin ausgeführt werden. Weitere Informationen über die GFS2-Rückzugsfunktion finden Sie in Abschnitt 4.14, »Die GFS2-Rückzugsfunktion«.
discard/nodiscard	Veranlasst GFS2 dazu, „discard“ I/O-Anfragen für Blöcke zu generieren, die freigegeben wurden. Von geeigneter Hardware kann dies dazu eingesetzt werden, um „Thin Provisioning“ und ähnliche Schemata zu implementieren.
barrier/nobarrier	Veranlasst GFS2 dazu, I/O-Barrieren zu senden, wenn das Journal auf die Festplatte übertragen wird. Der Standardwert lautet **on**. Diese Option wird automatisch auf **off** gesetzt, wenn das zugrunde liegende Gerät keine I/O-Barrieren unterstützt. Die Verwendung von I/O-Barrieren mit GFS2 wird grundsätzlich dringend empfohlen, es sei denn, das Blockgerät ist derart aufgebaut, dass es die Inhalte seines Schreib-Caches nicht verlieren kann (z. B. falls es an eine unterbrechungsfreie Stromversorgung angeschlossen ist oder über keinen Schreib-Cache verfügt).
quota_quantum=*secs*	Legt die Anzahl an Sekunden fest, für die eine Änderung an den Informationen der Festplattenkontingente auf einem Knoten verbleiben darf, bevor diese in die Kontingentdatei übertragen wird. Dies ist die bevorzugte Methode zum Einstellen dieses Parameters. Der Parameter ist ein ganzzahliger Wert größer als Null. Der Standard beträgt 60 Sekunden. Kleinere Werte führen zu schnelleren Aktualisierungen der Kontingentinformationen und einer geringeren Wahrscheinlichkeit, dass Benutzer oder Gruppen ihre Kontingente überschreiten. Größere Werte machen Dateisystemoperationen mit aktiven Kontingenten schneller und effizienter.

Optionen	Beschreibung
statfs_quantum=*secs*	Das Einstellen von **statfs_quantum** auf 0 ist die bevorzugte Methode zum Festlegen der langsamen Version von **statfs**. Der Standardwert beträgt 30 Sekunden, was die maximale Zeit festlegt, bevor **statfs**-Änderungen mit der Master-**statfs**-Datei synchronisiert werden. Dies kann angepasst werden, um schnellere, weniger genaue **statfs**-Werte oder langsamere, genauere Werte zu ermöglichen. Wenn diese Option auf 0 gesetzt ist, wird **statfs** immer die echten Werte berichten.
statfs_percent=*value*	Implementiert eine maximale Grenze prozentualer Änderungen in den **statfs**-Informationen auf lokaler Basis, bevor diese mit der Master-**statfs**-Datei synchronisiert werden, selbst wenn die festgelegte Zeitspanne noch nicht überschritten wurde. Falls die **statfs_quantum**-Einstellung 0 lautet, wird diese Einstellung ignoriert.

4.3. AUSHÄNGEN EINES DATEISYSTEMS

Das GFS2-Dateisystem kann auf dieselbe Art und Weise wie jedes andere Linux-Dateisystem ausgehängt werden – mithilfe des **umount**-Befehls.

ANMERKUNG

Der **umount**-Befehl ist ein Linux-Systembefehl. Informationen zu diesem Befehl finden Sie auf der man-Seite des **umount**-Linux-Befehls.

4.3.1. Verwendung

```
umount MountPoint
```

MountPoint

Gibt das Verzeichnis an, in dem das GFS2-Dateisystem derzeit eingehängt ist.

4.4. SPEZIELLE ÜBERLEGUNGEN ZUM EINHÄNGEN VON GFS2-DATEISYSTEMEN

GFS2-Dateisysteme, die manuell eingehängt wurden, statt automatisch durch einen Eintrag in der **fstab**-Datei, sind dem System nicht bekannt, wenn dieses beim Herunterfahren die Dateisysteme aushängt. Infolgedessen wird das GFS2-Skript das GFS2-Dateisystem nicht aushängen. Nachdem das GFS2-Shutdown-Skript ausgeführt wurde, beendet der standardmäßige Shutdown-Prozess alle verbleibenden Benutzerprozesse, einschließlich der Cluster-Infrastruktur, und versucht das Dateisystem auszuhängen. Dieses Aushängen schlägt jedoch ohne die Cluster-Infrastruktur fehl, sodass sich das System aufhängt.

Um ein Aufhängen des Systems zu verhindern, wenn die GFS2-Dateisysteme ausgehängt werden, haben Sie zwei Möglichkeiten:

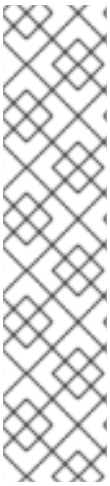
- Verwenden Sie grundsätzlich einen Eintrag in der **fstab**-Datei, um das GFS2-Dateisystem einzuhängen.
- Falls ein GFS2-Dateisystem manuell mit dem **mount**-Befehl eingehängt wurde, hängen Sie dies auch manuell wieder mit dem **umount**-Befehl aus, bevor Sie einen Neustart durchführen oder das System herunterfahren.

Falls sich Ihr Dateisystem beim Aushängen während des System-Shutdowns unter diesen Umständen aufhängt, führen Sie einen Hardware-Neustart aus. Es ist unwahrscheinlich, dass Daten verloren gehen, da das Dateisystem zu einem früheren Zeitpunkt im Shutdown-Vorgang synchronisiert wurde.

4.5. VERWALTEN VON GFS2-FESTPLATTENKONTINGENTEN

Dateisystemkontingente werden dazu verwendet, um die Menge des Dateisystemspeichers zu begrenzen, den ein Benutzer oder eine Gruppe verbrauchen darf. Ein Benutzer oder eine Gruppe hat standardmäßig kein begrenztes Kontingent, bevor nicht eines gesetzt wird. Wenn ein GFS2-Dateisystem mit der Option **quota=on** oder der **quota=account** eingehängt wird, verfolgt GFS2 den Speicherverbrauch von jedem Benutzer und jeder Gruppe nach, selbst wenn keine Grenzen gesetzt wurden. GFS2 aktualisiert die Kontingentinformationen auf transaktionale Weise, sodass nach einem Systemabsturz der Kontingentverbrauch nicht rekonstruiert werden muss.

Um nachteilige Auswirkungen auf die Leistung zu vermeiden, synchronisiert ein GFS2-Knoten die Aktualisierungen an der Kontingentdatei nur periodisch. Die demzufolge etwas ungenaue Nachverfolgung der Kontingente ermöglicht es Benutzern oder Gruppen unter Umständen, ihre gesetzten Grenzen leicht zu überschreiten. Um diesen Effekt zu minimieren, verkleinert GFS2 die zeitlichen Abstände der Synchronisation dynamisch, wenn sich die „harte“ Kontingentgrenze nähert.



ANMERKUNG

Ab der Red Hat Enterprise Linux 6.1 Release unterstützt GFS2 die standardmäßigen Linux-Funktionen für Festplattenkontingente. Um diese nutzen zu können, müssen Sie das **quota**-RPM installieren. Dies ist die bevorzugte Methode zur Verwaltung von Festplattenkontingenten auf GFS2 und sollte für alle neuen GFS2-Bereitstellungen, die Kontingente nutzen, verwendet werden. Dieser Abschnitt dokumentiert die GFS2-Kontingentverwaltung mithilfe dieser Funktionen.

In früheren Releases von Red Hat Enterprise Linux erforderte GFS2 den **gfs2_quota**-Befehl zur Verwaltung von Festplattenkontingenten. Informationen über die Verwendung des **gfs2_quota**-Befehls finden Sie in Anhang A, *GFS2-Kontingentverwaltung mit dem Befehl **gfs2_quota***.

4.5.1. Konfigurieren von Festplattenkontingenten

Um Festplattenkontingente zu implementieren, führen Sie die folgenden Schritte aus:

1. Richten Sie Kontingente im Erzwingen- oder Berechnen-Modus ein.
2. Initialisieren Sie die Kontingentdatenbankdatei mit den aktuellen Informationen zum Blockverbrauch.
3. Weisen Sie Kontingentrichtlinien zu. (Wurde lediglich der Berechnen-Modus eingestellt, werden diese Richtlinien jedoch nicht erzwungen.)

Jeder dieser Schritte wird in den nachfolgenden Abschnitten detailliert behandelt.

4.5.1.1. Einrichten von Kontingenten im Erzwingen- oder Berechnen-Modus

In GFS2-Dateisystemen sind Kontingente standardmäßig deaktiviert. Um Kontingente für ein Dateisystem zu aktivieren, hängen Sie das Dateisystem mit der Option **quota=on** ein.

Es ist möglich, den Festplattenverbrauch nachzuverfolgen und Kontingente für alle Benutzer und Gruppen zu berechnen, ohne die Grenz- oder Warnwerte zu erzwingen. Hängen Sie dazu das Dateisystem mit der Option **quota=account** ein.

4.5.1.1.1. Verwendung

Um ein Dateisystem mit aktivierten Kontingenten einzuhängen, hängen Sie das Dateisystem mit der Option **quota=on** ein.

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

Um ein Dateisystem mit aktivierter Kontingentberechnung einzuhängen, wobei die Kontingentgrenzen jedoch nicht erzwungen werden, hängen Sie das Dateisystem mit der Option **quota=account** ein.

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

Um ein Dateisystem mit deaktivierten Kontingenten einzuhängen, hängen Sie das Dateisystem mit der Option **quota=off** ein. Dies ist die Standardeinstellung.

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

quota={on|off|account}

on – Gibt an, dass Kontingente aktiviert sind, wenn das Dateisystem eingehängt wird.

off – Gibt an, dass Kontingente deaktiviert sind, wenn das Dateisystem eingehängt wird.

account – Gibt an, dass die Benutzer- und Gruppenverbrauchsstatistik im Dateisystem gepflegt werden soll, selbst wenn die Kontingentgrenzen nicht erzwungen werden.

BlockDevice

Gibt das Blockgerät an, auf dem sich das GFS2-Dateisystem befindet.

MountPoint

Gibt das Verzeichnis an, in dem das GFS2-Dateisystem eingehängt werden soll.

4.5.1.1.2. Beispiele

In diesem Beispiel wird das GFS2-Dateisystem **/dev/vg01/lvol0** mit aktivierten Kontingenten im **/mygfs2**-Verzeichnis eingehängt.

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

In diesem Beispiel wird das GFS2-Dateisystem **/dev/vg01/lvol0** mit Kontingentberechnung, jedoch ohne Erzwingen von Kontingentgrenzen, im **/mygfs2** Verzeichnis eingehängt.

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

-

4.5.1.2. Erstellen der Kontingent-Datenbankdateien

Nachdem jedes Dateisystem mit aktivierten Kontingenten eingehängt wurde, kann das System nun mit Kontingenten arbeiten. Allerdings ist das System selbst noch nicht dazu eingerichtet, Kontingente zu unterstützen. Der nächste Schritt ist daher das Ausführen des **quotacheck**-Befehls.

Der **quotacheck**-Befehl untersucht die Dateisysteme mit aktivierten Kontingenten und erzeugt eine Tabelle des aktuellen Festplattenverbrauchs pro Dateisystem. Diese Tabelle wird dann verwendet, um für das Betriebssystem eine laufend aktualisierte Kopie des Verbrauchs zu pflegen. Zudem werden die Kontingentdateien des Dateisystems aktualisiert.

Um die Kontingentdateien auf dem Dateisystem zu erstellen, verwenden Sie die Optionen **-u** und **-g** des **quotacheck**-Befehls; beide Optionen müssen angegeben werden, damit sowohl Benutzer- als auch Gruppenkontingente initialisiert werden. Wenn Kontingente beispielsweise auf dem **/home**-Dateisystem aktiviert sind, erstellen Sie die Dateien im **/home**-Verzeichnis:

```
quotacheck -ug /home
```

4.5.1.3. Kontingente pro Benutzer zuweisen

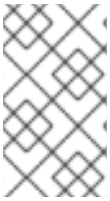
Weisen Sie abschließend die Festplattenkontingente mit dem **edquota**-Befehl zu. Beachten Sie, dass Kontingente nicht erzwungen werden, wenn Sie Ihr Dateisystem nur im Berechnen-Modus eingehängt haben (durch Angabe der Option **quota=account**).

Um ein Festplattenkontingent für einen Benutzer zu konfigurieren, führen Sie als Root in einer Shell den folgenden Befehl aus:

```
edquota username
```

Führen Sie diesen Schritt für jeden Benutzer durch, dem ein Kontingent zugewiesen werden soll. Wenn beispielsweise in der **/etc/fstab**-Datei Kontingente für die **/home**-Partition (**/dev/VolGroup00/LogVol02** im nachfolgenden Beispiel) aktiviert sind und Sie den Befehl **edquota testuser** ausführen, sehen Sie Folgendes im Standard-Editor des Systems:

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



ANMERKUNG

Der **edquota**-Befehl nutzt den Texteditor, der durch die **EDITOR**-Umgebungsvariable definiert wurde. Um den Editor zu ändern, setzen Sie die **EDITOR**-Umgebungsvariable in Ihrer **~/.bash_profile**-Datei auf den vollständigen Pfad zum Editor Ihrer Wahl.

Die erste Spalte zeigt den Namen des Dateisystems, auf dem Kontingente aktiviert wurden. Die zweite Spalte zeigt, wie viele Blöcke der Benutzer derzeit verbraucht. Die nächsten zwei Spalten legen die weichen und harten Blockgrenzen für den Benutzer auf dem Dateisystem fest.

Die weiche Blockgrenze definiert den maximalen Speicherplatz, der verbraucht werden kann.

Die harte Blockgrenze definiert den absolut maximalen Speicherplatz, den ein Benutzer oder eine Gruppe verbrauchen kann. Sobald diese Grenze erreicht wurde, kann kein weiterer Festplattenplatz verbraucht werden.

Das GFS2-Dateisystem pflegt keine Kontingente für Inodes, diese Spalten sind für GFS2-Dateisysteme demnach irrelevant und bleiben leer.

Falls einer der Werte auf 0 gesetzt ist, wurde diese Grenze nicht gesetzt. Ändern Sie im Texteditor die Grenzen auf die gewünschten Werte. Zum Beispiel:

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

Um zu überprüfen, ob das Kontingent wie gewünscht für den Benutzer festgelegt wurde, führen Sie den folgenden Befehl aus:

quota testuser

4.5.1.4. Kontingente pro Gruppe zuweisen

Kontingente können auch pro Gruppe zugewiesen werden. Beachten Sie, dass Kontingente nicht erzwungen werden, wenn Sie Ihr Dateisystem nur im Berechnen-Modus eingehängt haben (durch Angabe der Option **account=on**).

Um ein Gruppenkontingent für die **devel**-Gruppe festzulegen (die Gruppe muss bereits existieren, ehe das Gruppenkontingent festgelegt werden kann), verwenden Sie den folgenden Befehl:

```
edquota -g devel
```

Dieser Befehl zeigt die vorhandenen Kontingente für die Gruppe im Texteditor:

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

Das GFS2-Dateisystem pflegt keine Kontingente für Inodes, diese Spalten sind für GFS2-Dateisysteme demnach irrelevant und bleiben leer. Verändern Sie die Grenzen wie gewünscht und speichern Sie die Datei.

Um zu überprüfen, ob das Kontingent wie gewünscht für die Gruppe festgelegt wurde, führen Sie den folgenden Befehl aus:

```
quota -g devel
```

4.5.2. Verwalten von Festplattenkontingenten

Wenn Kontingente implementiert sind, erfordern diese ein gewisses Maß an Pflege – hauptsächlich sollte überwacht werden, ob Kontingente überschritten werden und ob die gesetzten Grenzen noch angemessen sind.

Sollten Benutzer wiederholt ihre Kontingentgrenzen überschreiten oder dauerhaft die weichen Grenzen erreichen, muss der Systemadministrator, abhängig von der Art des Benutzers und der Auswirkungen

des begrenzten Speicherplatzes auf ihre Arbeit, die Entscheidung treffen, wie vorzugehen ist. Der Administrator kann dem Benutzer entweder dabei helfen, seinen Speicherplatzverbrauch zu verringern, oder er kann das Festplattenkontingent des Benutzers erhöhen.

Sie können mithilfe des **repquota**-Befehls einen Bericht des Festplattenverbrauchs generieren. Beispielsweise erzeugt der Befehl **repquota /home** die folgende Ausgabe:

```
*** 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
```

Um einen Bericht zum Festplattenverbrauch für alle (Option **-a**) Dateisysteme mit aktivierten Kontingenten zu erhalten, führen Sie den folgenden Befehl aus:

```
repquota -a
```

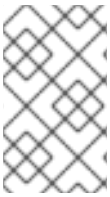
Der Bericht ist zwar recht einfach verständlich, aber einige Punkte sollten dennoch erläutert werden. Die Zeichen **--** nach jedem Benutzer bieten einen schnellen Weg, um festzustellen, ob die Blockgrenzen überschritten wurden. Falls die weiche Blockgrenze überschritten wurde, erscheint ein **+** anstelle des ersten **-** in der Ausgabe. Das zweite **-** repräsentiert die Inode-Grenze; da GFS2-Dateisysteme jedoch keine Inode-Grenzen unterstützen, bleibt an dieser Stelle stets das Zeichen **-**. GFS2-Dateisysteme unterstützen keine Schonfristen, die **grace**-Spalte bleibt demzufolge leer.

Beachten Sie, dass der **repquota**-Befehl nicht über NFS unterstützt wird, unabhängig vom zugrunde liegenden Dateisystem.

4.5.3. Pflegen der Genauigkeit von Kontingenten

Falls Sie Kontingente auf Ihrem Dateisystem aktivieren, nachdem Sie einige Zeit mit deaktivierten Kontingenten gearbeitet haben, sollten Sie den **quotacheck**-Befehl ausführen, um die Kontingentdateien zu erstellen, zu überprüfen und zu reparieren. Außerdem sollten Sie den **quotacheck**-Befehl ausführen, wenn Sie den Eindruck haben, dass Ihre Kontingentdateien nicht genau sind, was in Situationen der Fall sein kann, in denen ein Dateisystem nach einem Systemabsturz nicht ordnungsgemäß ausgehängt wurde.

Weitere Informationen über den **quotacheck**-Befehl finden Sie auf der man-Seite für **quotacheck**.



ANMERKUNG

Führen Sie den **quotacheck**-Befehl zu einem Zeitpunkt durch, an dem das Dateisystem auf allen Knoten relativ untätig ist, da Festplattenaktivität die Berechnung der Kontingentwerte stören kann.

4.5.4. Synchronisieren von Kontingenten mit dem `quotasync`-Befehl

GFS2 speichert alle Kontingentinformationen in seiner eigenen, internen Datei auf der Festplatte. Ein GFS2-Knoten aktualisiert diese Kontingentdatei nicht bei jedem Schreibvorgang im Dateisystem, sondern aktualisiert sie standardmäßig nur alle 60 Sekunden. Dies ist notwendig, um Konflikte zwischen

Knoten zu vermeiden, die in die Kontingentdatei schreiben, was sich andernfalls nachteilig auf die Leistung auswirken würde.

Wenn ein Benutzer oder eine Gruppe sich seiner bzw. ihrer Kontingentgrenze nähert, verkürzt GFS2 dynamisch die zeitlichen Abstände zwischen den Aktualisierungen der Kontingentdatei, um das Überschreiten der Grenze zu vermeiden. Der normale Zeitabstand zwischen Kontingentsynchronisationen ist ein einstellbarer Parameter, **quota_quantum**. Sie können dessen Standardwert von 60 Sekunden mithilfe der Einhängeoption **quota_quantum=** ändern, wie in Tabelle 4.2, »GFS2-spezifische Einhängeoptionen« beschrieben. Der **quota_quantum**-Parameter muss auf jedem Knoten und jedes Mal, wenn das Dateisystem eingehängt wird, gesetzt werden. Änderungen am **quota_quantum**-Parameter sind über Aushängevorgänge hinweg nicht persistent. Sie können den **quota_quantum**-Wert mit dem Befehl **mount -o remount** aktualisieren.

Sie können den **quotasync**-Befehl verwenden, um zwischen den automatischen Aktualisierungen durch GFS2 die Kontingentinformationen manuell von einem Knoten auf die Kontingentdatei auf der Festplatte zu synchronisieren.

4.5.4.1. Verwendung

Synchronisation der Kontingentinformationen

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

u

Synchronisiert die Benutzer-Kontingentdateien.

g

Synchronisiert die Gruppen-Kontingentdateien.

a

Synchronisiert alle Dateisysteme, die derzeit Kontingente aktiviert haben und die Synchronisation unterstützen. Wenn -a nicht angegeben ist, sollte ein Einhängepunkt im Dateisystem angegeben werden.

mntpnt

Gibt das GFS2-Dateisystem an, auf dem diese Aktion durchzuführen ist.

Anpassen der Zeitabstände zwischen Synchronisationen

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

MountPoint

Gibt das GFS2-Dateisystem an, auf dem diese Aktion durchzuführen ist.

secs

Gibt den Zeitabstand zwischen regulären Synchronisationen der Kontingentdatei durch GFS2 an. Kleinere Werte können zu mehr Konflikten führen und sich nachteilig auf die Leistung auswirken.

4.5.4.2. Beispiele

Dieses Beispiel synchronisiert alle gecachten, veränderten Kontingente von dem Knoten, auf dem der Befehl ausgeführt wird, auf die Kontingentdatei des Datenträgers für das Dateisystem **/mnt/mygfs2**.

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

In diesem Beispiel wird der standardmäßige Zeitraum zwischen den regelmäßigen Aktualisierungen der Kontingentdatei auf eine Stunde (3600 Sekunden) für das Dateisystem **/mnt/mygfs2** beim Wiedereinhängen dieses Dateisystems auf dem logischen Datenträger **/dev/volgroup/logical_volume** geändert.

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

4.5.5. Referenzen

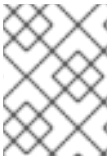
Weitere Informationen über Festplattenkontingente finden Sie auf den **man**-Seiten der folgenden Befehle:

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

4.6. VERGRÖSSERN EINES DATEISYSTEMS

Der **gfs2_grow**-Befehl wird zum Erweitern des GFS2-Dateisystems verwendet, nachdem der Speicher, auf dem das Dateisystem liegt, erweitert wurde. Startet man den **gfs2_grow**-Befehl auf einem existierenden GFS2-Dateisystem, wird sämtlicher freier Platz zwischen dem derzeitigen Ende des Dateisystems und dem Ende des Geräts mit einer neu initialisierten GFS2-Dateisystemerweiterung gefüllt. Wenn der Füllvorgang abgeschlossen ist wird der Ressourcenindex des Dateisystems aktualisiert. Alle Knoten im Cluster können dann den zusätzlich hinzugefügten Speicherplatz verwendet.

Der **gfs2_grow**-Befehl muss auf einem eingehängten Dateisystem laufen, braucht jedoch nur auf einem Knoten im Cluster ausgeführt zu werden. Die anderen Knoten merken, dass eine Erweiterung stattgefunden hat, und benutzen den neuen Speicherplatz automatisch.



ANMERKUNG

Nachdem Sie ein GFS2-Dateisystem mit dem **mkfs.gfs2**-Befehl erstellt haben, können Sie es nicht mehr verkleinern.

4.6.1. Verwendung

```
gfs2_grow MountPoint
```

MountPoint

Gibt das GFS2-Dateisystem an, auf dem diese Aktion durchzuführen ist.

4.6.2. Anmerkungen

Bevor Sie den **gfs2_grow**-Befehl ausführen:

- Sichern Sie alle wichtigen Daten auf dem Dateisystem.
- Ermitteln Sie den Datenträger, der von dem zu erweiternden Dateisystem verwendet wird, indem Sie den Befehl **df *MountPoint*** ausführen.
- Erweitern Sie den zugrunde liegenden Cluster-Datenträger mit LVM. Mehr Informationen über die Verwaltung von LVM-Datenträgern finden Sie im Handbuch *Administration des Logical Volume Manager*.

Führen Sie nach dem **gfs2_grow**-Befehl nun den **df**-Befehl aus, um zu überprüfen, ob der neue Speicher jetzt im Dateisystem verfügbar ist.

4.6.3. Beispiele

In diesem Beispiel wird das Dateisystem auf dem **/mygfs2fs**-Verzeichnis erweitert.

```
[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.
```

4.6.4. Vollständige Verwendung

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

MountPoint

Gibt das Verzeichnis an, in dem das GFS2-Dateisystem eingehängt ist.

Device

Gibt den Geräteknoten des Dateisystems an.

Tabelle 4.3, »Verfügbare GFS2-spezifische Optionen zum Erweitern eines Dateisystems« beschreibt die GFS2-spezifischen Optionen, die Sie beim Erweitern eines GFS2-Dateisystems verwenden können.

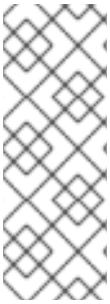
Tabelle 4.3. Verfügbare GFS2-spezifische Optionen zum Erweitern eines Dateisystems

Optionen	Beschreibung
-h	Hilfe. Zeigt kurze Informationen zur Verwendung an.
-q	Verringert die Ausführlichkeit der Ausgabe.

Optionen	Beschreibung
-r MegaBytes	Gibt die Größe der neuen Ressourcengruppe an. Die standardmäßige Größe ist 256 MB.
-T	Test. Führt sämtliche Berechnungen durch, schreibt jedoch keine Daten auf den Datenträger und erweitert das Dateisystem nicht.
-V	Zeigt Informationen zur Befehlsversion an.

4.7. HINZUFÜGEN VON JOURNALEN ZU EINEM DATEISYSTEM

Der **gfs2_jadd**-Befehl wird dazu verwendet, um Journale zu einem GFS2-Dateisystem hinzuzufügen. Sie können jederzeit Journale zu einem GFS2-Dateisystem dynamisch hinzufügen, ohne den zugrunde liegenden logischen Datenträger zu erweitern. Der **gfs2_jadd**-Befehl muss auf einem eingehängten Dateisystem ausgeführt werden, allerdings braucht er nur auf einem Knoten im Cluster ausgeführt zu werden. Alle anderen Knoten merken automatisch, dass eine Erweiterung stattgefunden hat.



ANMERKUNG

Falls ein GFS2-Dateisystem voll ist, schlägt der **gfs2_jadd**-Befehl fehl, selbst wenn der logische Datenträger, der das Dateisystem enthält, erweitert wurde und größer als das Dateisystem ist. Der Grund hierfür liegt in den Journalen des GFS2-Dateisystems, denn bei den Journalen handelt es sich um einfache Dateien statt um eingebettete Metadaten, sodass ein einfaches Erweitern des zugrunde liegenden logischen Datenträgers nicht mehr Platz für die Journale verfügbar macht.

Bevor Sie Journale zu einem GFS-Dateisystem hinzufügen, können Sie die **journals**-Option des **gfs2_tool**-Befehls verwenden, um herauszufinden, wie viele Journale das GFS2-Dateisystem derzeit enthält. Das folgende Beispiel zeigt die Anzahl und die Größe der Journale im unter **/mnt/gfs2** eingehängten Dateisystem an.

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

4.7.1. Verwendung

```
gfs2_jadd -j Number MountPoint
```

Number

Gibt die Anzahl der neu hinzuzufügenden Journale an.

MountPoint

Gibt das Verzeichnis an, in dem das GFS2-Dateisystem eingehängt ist.

4.7.2. Beispiele

In diesem Beispiel wird ein Journal zum Dateisystem im **/mygfs2**-Verzeichnis hinzugefügt.

```
gfs2_jadd -j1 /mygfs2
```

In diesem Beispiel werden zwei Journale zum Dateisystem im **/mygfs2**-Verzeichnis hinzugefügt.

```
gfs2_jadd -j2 /mygfs2
```

4.7.3. Vollständige Verwendung

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

MountPoint

Gibt das Verzeichnis an, in dem das GFS2-Dateisystem eingehängt ist.

Device

Gibt den Geräteknoten des Dateisystems an.

Tabelle 4.4, »GFS2 spezifische Optionen zum Hinzufügen von Journalen« beschreibt die GFS2-spezifischen Optionen, die Sie beim Hinzufügen von Journalen zu einem GFS2-Dateisystem verwenden können.

Tabelle 4.4. GFS2 spezifische Optionen zum Hinzufügen von Journalen

Flag	Parameter	Beschreibung
-h		Hilfe. Zeigt kurze Informationen zur Verwendung an.
-J	***MegaBytes***	Gibt die Größe der neuen Journale in Megabytes an. Die standardmäßige Journalgröße ist 128 Megabytes. Die minimale Größe ist 32 Megabytes. Um Journale mit verschiedenen Größen zum Dateisystem hinzuzufügen, muss der **gfs2_jadd**-Befehl für jede gewünschte Journalgröße ausgeführt werden. Die angegebene Größe wird abgerundet, sodass diese einem Vielfachen der Journal-Segmentgröße entspricht, die bei der Erstellung des Dateisystems festgelegt wurde.
-j	***Number***	Gibt die Anzahl der neuen Journale an, die vom **gfs2_jadd**-Befehl hinzugefügt werden. Der standardmäßige Wert ist 1.
-q		Verringert die Ausführlichkeit der Ausgabe.
-V		Zeigt Informationen zur Befehlsversion an.

4.8. DATENJOURNALE

Gewöhnlich schreibt GFS2 nur Metadaten in die Journale. Dateiinhalte werden nachträglich auf die Festplatte geschrieben, wenn der Kernel seine regelmäßige Synchronisation durchführt, die alle Dateisystempuffer auf die Platte schreibt. Ein **fsync()**-Aufruf auf einer Datei verursacht, dass die Daten der Datei sofort auf die Platte geschrieben werden. Der Aufruf kehrt zurück, wenn die Platte meldet, dass sämtliche Daten sicher geschrieben wurden.

Das Pflegen von Datenjournale kann die benötigte Zeit für **fsync()**-Operationen für sehr kleine Dateien verkürzen, da zusätzlich zu den Metadaten auch die Dateidaten in das Journal geschrieben werden. Dieser Vorteil verringert sich jedoch rapide bei größeren Dateien. Das Schreiben in mittlere und große Dateien ist sehr viel langsamer mit aktivierten Datenjournalen.

Applikationen, die auf **fsync()** zur Synchronisation von Dateidaten angewiesen sind, erzielen eine bessere Leistung durch die Verwendung von Datenjournalen. Das Datenjournaling kann automatisch für jede GFS2-Datei aktiviert werden, die in einem entsprechend gekennzeichneten Verzeichnis (und all seinen Unterverzeichnissen) erstellt wird. Vorhandene Dateien mit Null-Länge können ebenfalls das Datenjournaling aktiviert oder deaktiviert haben.

Ist das Datenjournaling in einem Verzeichnis aktiviert, wird das Verzeichnis auf „inherit jdata“ gesetzt, was bedeutet, dass alle in diesem Verzeichnis erstellten Dateien und Unterverzeichnisse journalisiert werden. Sie können das Datenjournaling auf einer Datei mithilfe des **chattr**-Befehls aktivieren bzw. deaktivieren.

Die folgenden Befehle aktivieren das Datenjournaling auf der Datei **/mnt/gfs2/gfs2_dir/newfile** und überprüfen anschließend, ob das Flag richtig gesetzt wurde.

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

Die folgenden Befehle deaktivieren das Datenjournaling auf der Datei **/mnt/gfs2/gfs2_dir/newfile** und überprüfen anschließend, ob das Flag richtig gesetzt wurde.

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

Sie können den **chattr**-Befehl auch dazu verwenden, um das **j**-Flag auf einem Verzeichnis zu setzen. Wenn Sie dieses Flag auf einem Verzeichnis setzen, werden alle nachfolgend in diesem Verzeichnis erstellten Dateien und Unterverzeichnisse journalisiert. Die folgenden Befehle setzen das **j**-Flag auf dem **gfs2_dir**-Verzeichnis und überprüfen danach, ob das Flag richtig gesetzt wurde. Anschließend erstellen diese Befehle eine neue Datei namens **newfile** im Verzeichnis **/mnt/gfs2/gfs2_dir** und überprüfen danach, ob das **j**-Flag für die Datei gesetzt wurde. Da das **j**-Flag für das Verzeichnis gesetzt wurde, sollte auch für die **newfile**-Datei das Journaling aktiviert sein.

```
[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. KONFIGURIEREN DER ATIME-AKTUALISIERUNGEN

Jeder Datei-Inode und Verzeichniss-Inode hat drei Timestamps:

- **ctime** – Der Zeitpunkt, an dem zum letzten Mal der Inode-Status verändert wurde
- **mtime** – Der Zeitpunkt, an dem zum letzten Mal die Datei (oder das Verzeichnis) verändert wurde
- **atime** – Der Zeitpunkt, an dem zum letzten Mal auf die Datei (oder das Verzeichnis) zugegriffen wurde

Falls die **atime**-Aktualisierung aktiviert ist, was standardmäßig bei GFS2 und anderen Linux-Dateisystemen der Fall ist, dann muss jedes Mal, wenn die Datei gelesen wird, dessen Inode aktualisiert werden.

Da nur wenige Applikationen die **atime**-Informationen nutzen, können diese Aktualisierungen eine große Menge unnötiger Schreibvorgänge und Dateisperrvorgänge verursachen. Diese Vorgänge können die Leistung beeinträchtigen, daher kann es ggf. empfehlenswert sein, die **atime**-Aktualisierung zu deaktivieren oder deren Häufigkeit einzuschränken.

Zwei Methoden stehen zur Verfügung, um die Auswirkungen von **atime**-Aktualisierungen zu reduzieren:

- Hängen Sie das Dateisystem mit **relatime** (relative atime) ein, wodurch **atime** aktualisiert wird, falls die vorherige **atime**-Aktualisierung älter als die **mtime**- oder die **ctime**-Aktualisierung ist.
- Hängen Sie das Dateisystem mit **noatime** ein, wodurch **atime**-Aktualisierungen für dieses Dateisystem deaktiviert werden.

4.9.1. Einhängen mit `relatime`

Die Linux-Einhängeoption **relatime** (relative atime) kann beim Einhängen des Dateisystems angegeben werden. Dadurch wird **atime** aktualisiert, falls die vorherige **atime**-Aktualisierung älter als die **mtime**- oder die **ctime**-Aktualisierung ist.

4.9.1.1. Verwendung

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

BlockDevice

Gibt das Blockgerät an, auf dem sich das GFS2-Dateisystem befindet.

MountPoint

Gibt das Verzeichnis an, in dem das GFS2-Dateisystem eingehängt werden soll.

4.9.1.2. Beispiel

In diesem Beispiel befindet sich das GFS2-Dateisystem auf **/dev/vg01/lvol0** und wird unter dem Verzeichnis **/mygfs2** eingehängt. Die **atime**-Aktualisierung wird nur dann durchgeführt, wenn die vorherige **atime**-Aktualisierung älter als die **mtime**- oder die **ctime**-Aktualisierung ist.

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

4.9.2. Einhängen mit `noatime`

Die Linux-Einhängeoption **noatime** kann beim Einhängen des Dateisystems angegeben werden, wodurch die **atime**-Aktualisierung auf diesem Dateisystem deaktiviert wird.

4.9.2.1. Verwendung

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

BlockDevice

Gibt das Blockgerät an, auf dem sich das GFS2-Dateisystem befindet.

MountPoint

Gibt das Verzeichnis an, in dem das GFS2-Dateisystem eingehängt werden soll.

4.9.2.2. Beispiel

In diesem Beispiel befindet sich das GFS2-Dateisystem auf **/dev/vg01/lvol0** und wird unter dem Verzeichnis **/mygfs2** mit deaktivierter **atime**-Aktualisierung eingehängt.

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

4.10. UNTERBRECHEN DER AKTIVITÄT AUF EINEM DATEISYSTEM

Sie können die Schreibaktivität auf einem Dateisystem unterbrechen, indem Sie den Befehl **dmsetupsuspend** benutzen. Das Unterbrechen der Schreibaktivität erlaubt die Verwendung von hardwarebasierten Geräte-Snapshots, um das Dateisystem in einem konsistenten Zustand zu erfassen. Der Befehl **smsetupresume** beendet die Unterbrechung.

4.10.1. Verwendung

Unterbrechung starten

```
dmsetup suspend MountPoint
```

Unterbrechung beenden

```
dmsetup resume MountPoint
```

MountPoint

Gibt das Dateisystem an

4.10.2. Beispiele

Dieses Beispiel unterbricht die Schreibaktivität auf dem Dateisystem **/mygfs2**.

```
# dmsetup suspend /mygfs2
```

Dieses Beispiel beendet die Unterbrechung der Schreibaktivität auf dem Dateisystem **/mygfs2**.

```
# dmsetup resume /mygfs2
```

4.11. REPARIEREN EINES DATEISYSTEMS

Wenn Knoten mit eingehängtem Dateisystem ausfallen, ermöglicht das Dateisystem-Journal eine schnelle Wiederherstellung. Wenn auf einem Speichergerät jedoch ein Stromausfall auftritt oder die Verbindung physisch unterbrochen wird, kann das Dateisystem Schaden nehmen. (Journaling kann nicht dazu verwendet werden, um das Dateisystem nach Ausfällen oder Fehlern im Speichersubsystem wiederherzustellen.) Falls diese Art von Fehler auftritt, können Sie das GFS2-Dateisystem mithilfe des **fsck.gfs2**-Befehls wiederherstellen.

WICHTIG

Der **fsck.gfs2**-Befehl darf nur auf einem Dateisystem ausgeführt werden, das auf allen Knoten ausgehängt ist.

WICHTIG

Sie sollten ein GFS2-Dateisystem nicht beim Systemstart mit dem **fsck.gfs2**-Befehl überprüfen. Der **fsck.gfs2**-Befehl kann beim Start nicht feststellen, ob das Dateisystem auf einem anderen Knoten im Cluster eingehängt ist. Sie sollten den **fsck.gfs2**-Befehl erst nach dem Systemstart manuell durchführen.

Um sicherzustellen, dass der **fsck.gfs2**-Befehl nicht auf einem GFS2-Dateisystem beim Systemstart ausgeführt werden kann, ändern Sie die **/etc/fstab**-Datei, sodass die beiden letzten Spalten für den Einhängepunkt des GFS2-Dateisystems „0 0“ statt „1 1“ anzeigen (oder beliebige andere Zahlen), wie im folgenden Beispiel:

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

ANMERKUNG

Falls Sie bereits Erfahrungen mit dem gfs_fsck-Befehl auf GFS-Dateisystemen haben, beachten Sie bitte, dass sich der **fsck.gfs2**-Befehl von einigen älteren **gfs_fsck**-Versionen wie folgt unterscheidet:

- Wenn Sie **Strg**+**C** während der Ausführung von **fsck.gfs2** drücken, wird die Verarbeitung unterbrochen und eine Eingabeaufforderung angezeigt, die Sie fragt, ob Sie den Befehl abbrechen möchten, den Rest des aktuellen Durchlaufs überspringen möchten oder die Verarbeitung fortsetzen möchten.
- Sie können den Grad der Ausführlichkeit der Ausgabe erhöhen, indem Sie das **-v**-Flag verwenden. Ein weiteres **-v** erhöht die Ausführlichkeit nochmals.
- Sie können den Grad der Ausführlichkeit der Ausgabe verringern, indem Sie das **-q**-Flag verwenden. Ein weiteres **-q** verringert die Ausführlichkeit nochmals.
- Die Option **-n** öffnet ein Dateisystem schreibgeschützt und beantwortet alle Anfragen automatisch mit **no**. Diese Option bietet eine Möglichkeit, den Befehl auszuprobieren, um Fehler aufzudecken, ohne dass der **fsck.gfs2**-Befehl jedoch tatsächlich wirksam ist.

Weitere Informationen über andere Befehlsoptionen finden Sie auf der man-Seite für **fsck.gfs2**.

Das Ausführen des **fsck.gfs2**-Befehls erfordert Systemspeicher noch über den Speicher für das Betriebssystem und den Kernel hinaus. Jeder Speicherblock im GFS2-Dateisystem selbst erfordert ungefähr fünf Bits zusätzlichen Speicher oder 5/8 eines Bytes. Um abzuschätzen, wie viele Bytes an Speicher Sie zur Ausführung des **fsck.gfs2**-Befehls auf Ihrem Dateisystem benötigen, bestimmen Sie, wie viele Blöcke das Dateisystem umfasst und multiplizieren Sie diese Zahl mit 5/8.

Um beispielsweise abzuschätzen, wie viel Speicher erforderlich ist, um den **fsck.gfs2**-Befehl auf einem GFS2-Dateisystem auszuführen, das 16 TB groß ist und eine Blockgröße von 4 K aufweist, bestimmen Sie zunächst, wie viele Speicherblöcke das Dateisystem umfasst, indem Sie 16 TB durch 4 K teilen:

```
17592186044416 / 4096 = 4294967296
```

Dieses Dateisystem umfasst 4294967296 Blöcke; multiplizieren Sie diese Zahl also mit 5/8, um zu bestimmen, wie viele Bytes an Speicher erforderlich sind:

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

Dieses Dateisystem erfordert ungefähr 2,6 GB an freiem Speicher, um den **fsck.gfs2**-Befehl auszuführen. Beachten Sie, dass bei einer Blockgröße von 1 K zur Ausführung von **fsck.gfs2** die vierfache Menge an Speicher notwendig wäre, nämlich etwa 11 GB.

4.11.1. Verwendung

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

-y

Mit dem Flag **-y** werden alle Fragen mit **yes** beantwortet. Wenn Sie das Flag **-y** angeben, fordert Sie der **fsck.gfs2**-Befehl nicht zur Eingabe einer Antwort auf, bevor Änderungen vorgenommen werden.

BlockDevice

Gibt das Blockgerät an, auf dem sich das GFS2-Dateisystem befindet.

4.11.2. Beispiel

In diesem Beispiel wird das GFS2-Dateisystem repariert, das auf dem Blockgerät **/dev/testvol/testlv** liegt. Alle Fragen zur Reparatur werden automatisch mit **yes** beantwortet.

```
[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. BIND MOUNTS UND KONTEXTABHÄNGIGE PFADE

GFS2-Dateisysteme unterstützen keine kontextabhängigen Pfade (Context-Depentent Path Names oder kurz CDPNs), die es Ihnen ermöglichen, symbolische Links zu erzeugen, die auf verschiedene Zieldateien oder -verzeichnisse zeigen. Um dieselbe Funktionalität In GFS2-Dateisystemen zu erreichen, können Sie die **bind**-Option des **mount**-Befehls verwenden.

Die **bind**-Option des **mount**-Befehls ermöglicht es Ihnen, einen Teil einer Dateihierachie an anderer Stelle einzuhängen, während dieser gleichzeitig weiterhin an der ursprünglichen Stelle erreichbar bleibt. Das Format des Befehls lautet wie folgt:

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

Nach der Ausführung des Befehls ist der Inhalt des ***olddir***-Verzeichnisses von zwei Orten aus erreichbar: ***olddir*** und ***newdir***. Sie können diese Option auch dazu verwenden, um eine einzelne Datei von zwei Orten aus erreichbar zu machen.

Beispielsweise ist der Inhalt des **/root/tmp**-Verzeichnisses nach Ausführung des folgenden Befehls identisch mit dem Inhalt des vorher eingehängten **/var/log**-Verzeichnisses.

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

Alternativ können Sie auch einen Eintrag in die **/etc/fstab**-Datei einfügen, um beim Einhängen das gleiche Ergebnis zu erreichen. Der folgende **/etc/fstab**-Eintrag führt dazu, dass die Inhalte von **/root/tmp** mit den Inhalten des **/var/log**-Verzeichnisses identisch sind.

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

Nachdem das Dateisystem eingehängt wurde, können Sie wie im folgenden Beispiel dargestellt den **mount**-Befehl verwenden, um zu überprüfen, ob das Dateisystem eingehängt worden ist.

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

Angenommen, Sie haben mit einem Dateisystem, das kontextabhängige Pfade unterstützt, das **/bin**-Verzeichnis als kontextabhängigen Pfad definiert, der abhängig von der Systemarchitektur in einen der folgenden Pfade aufgelöst wird.

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

Sie können das gleiche Ergebnis erreichen, indem Sie ein leeres **/bin**-Verzeichnis erstellen. Anschließend können Sie entweder mithilfe eines Skripts oder mithilfe eines Eintrags in der **/etc/fstab**-Datei jedes der einzelnen Architekturverzeichnisse unter dem **/bin**-Verzeichnis mit dem Befehl **mount -bind** einhängen. Beispielsweise könnten Sie den folgenden Befehl als eine Zeile in einem Skript verwenden.

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

Alternativ können Sie den folgenden Eintrag in der **/etc/fstab**-Datei verwenden.

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

Ein Bind Mount bietet mehr Flexibilität als ein kontextabhängiger Pfad, da Sie mithilfe dieser Funktion verschiedene Verzeichnisse abhängig von verschiedenen, von Ihnen definierten Kriterien einhängen können (z. B. abhängig vom **%fill**-Wert für das Dateisystem). Kontextabhängige Pfade sind dagegen eher eingeschränkt in ihren Möglichkeiten. Beachten Sie jedoch, dass Sie Ihr eigenes Skript schreiben müssen, um Dateisysteme abhängig von Kriterien wie dem **%fill**-Wert einhängen zu können.

WARNUNG

Wenn Sie ein Dateisystem mit der **bind**-Option einhängen und das Originaldateisystem als **rw** eingehängt war, dann wird das neue Dateisystem genauso eingehängt (**rw**), selbst wenn Sie das **ro**-Flag verwenden. Das **ro**-Flag wird ohne jegliche Meldung ignoriert. In diesem Fall kann das neue Dateisystem im **/proc/mounts**-Verzeichnis unter Umständen dennoch als **ro** gekennzeichnet sein, was irreführend sein kann.

4.13. EINHÄNGEREIHENFOLGE FÜR BIND MOUNTS UND DATEISYSTEME

Wenn Sie die **bind**-Option des **mount**-Befehls verwenden, müssen Sie sicherstellen, dass die Dateisysteme in der richtigen Reihenfolge eingehängt werden. In dem folgenden Beispiel muss das **/var/log**-Verzeichnis eingehängt werden, bevor der Bind Mount auf dem **/tmp**-Verzeichnis ausgeführt wird:

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

Die Reihenfolge, in der Dateisysteme eingehängt werden, wird wie folgt bestimmt:

- Üblicherweise folgt die Reihenfolge, in der Dateisysteme eingehängt werden, der Reihenfolge, in der diese in der **fstab**-Datei aufgeführt sind. Ausnahmen davon betreffen Dateisysteme, die mit dem **_netdev**-Flag eingehängt werden, oder Dateisysteme, die über eigene **init**-Skripte verfügen.
- Ein Dateisystem mit eigenem **init**-Skript wird später im Initialisierungsprozess eingehängt, nach den Dateisystemen in der **fstab**-Datei.
- Dateisysteme mit dem **_netdev**-Flag werden eingehängt, sobald das Netzwerk auf dem System aktiviert wurde.

Falls Ihre Konfiguration erfordert, dass Sie ein Bind Mount erstellen, auf dem ein GFS2-Dateisystem eingehängt wird, können Sie Ihre **fstab**-Datei wie folgt ordnen:

1. Einhängen der lokalen Dateisysteme, die für den Bind Mount notwendig sind.
2. Bind Mount des Verzeichnisses, auf dem das GFS2-Dateisystem eingehängt werden soll.
3. Einhängen des GFS2-Dateisystems.

Falls Ihre Konfiguration erfordert, dass Sie ein lokales Verzeichnis oder Dateisystem mittels Bind Mount auf einem GFS2-Dateisystem einhängen, würde auch die richtige Reihenfolge der Dateisysteme in der **fstab**-Datei die Dateisysteme nicht richtig einhängen, da das GFS2-Dateisystem erst eingehängt wird, wenn das GFS2-**init**-Skript ausgeführt wird. In diesem Fall sollten Sie ein **init**-Skript schreiben, um den Bind Mount derart zu konfigurieren, dass der Bind Mount erst durchgeführt wird, nachdem das GFS2-Dateisystem eingehängt wurde.

Das folgende Skript ist ein Beispiel für ein benutzerdefiniertes **init**-Skript. Dieses Skript führt einen Bind Mount von zwei Verzeichnissen auf zwei Verzeichnisse eines GFS2-Dateisystems durch. In diesem

Beispiel gibt es einen vorhandenen GFS2-Einhängepunkt unter **/mnt/gfs2a**, der eingehängt wird, wenn das GFS2-**init**-Skript nach dem Cluster-Start ausgeführt wird.

In diesem Beispielskript bedeuten die Werte der **chkconfig**-Anweisung Folgendes:

- 345 steht für die Runlevels, in denen das Skript gestartet wird
- 29 ist die Startpriorität, was in diesem Fall bedeutet, dass das Skript zum Startzeitpunkt nach dem GFS2-**init**-Skript ausgeführt wird, welches eine Startpriorität von 26 hat
- 73 ist die Stoppprioritä t, was in diesem Fall bedeutet, dass das Skript während des Herunterfahrens vor dem GFS2-Skript gestoppt wird, welches eine Stoppprioritä t von 74 hat

Die Start- und Stoppwerte zeigen, dass Sie die angegebene Aktion manuell durch Ausführen der Befehle **service start** und **service stop** vornehmen können. Falls das Skript beispielsweise **fredwilma** heißt, können Sie **service fredwilma start** ausführen.

Dieses Skript sollte im **/etc/init.d**-Verzeichnis abgelegt werden, mit denselben Berechtigungen wie die anderen Skripte in dem Verzeichnis. Sie können anschließend den Befehl **chkconfig on** ausführen, um das Skript mit den angegebenen Runlevels zu verknüpfen. Falls das Skript beispielsweise **fredwilma** heißt, können Sie **chkconfig fredwilma on** ausführen.

```
#!/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. DIE GFS2-RÜCKZUGSFUNKTION

Die GFS2-*Rückzugsfunktion* (auch engl. *Withdraw-Funktion*) ist ein Feature zur Sicherung der Datenintegrität auf GFS2-Dateisystemen in einem Cluster. Falls das GFS2-Kernel-Modul nach einer I/O-Operation eine Inkonsistenz in einem GFS2-Dateisystem entdeckt, wird dem Cluster der Zugriff auf das Dateisystem verwehrt. Die I/O-Operation stoppt und das System wartet, bis weitere I/O-Operationen ebenfalls mit einem Fehler fehlschlagen, um weiteren Schaden zu verhindern. Sie können in dieser Situation andere Dienste oder Applikationen manuell stoppen, anschließend neu starten und das GFS2-Dateisystem wieder einhängen, um anhand der Journale wiederherzustellen. Wenn das Problem weiterhin besteht, können Sie das Dateisystem auf allen Knoten im Cluster aushängen und mit dem **`fsck.gfs2`**-Befehl eine Wiederherstellung des Dateisystems durchführen. Die GFS-Rückzugsfunktion ist weniger schwerwiegend als eine Kernel-Panic, bei der ein anderer Knoten diesen Knoten abgrenzen würde.

Falls auf Ihrem System das **`gfs2`**-Startup-Skript aktiviert ist und das GFS2-Dateisystem in der **`/etc/fstab`**-Datei enthalten ist, wird das GFS2-Dateisystem bei einem Neustart neu eingehängt. Falls sich das GFS2-Dateisystem aufgrund eines beschädigten Dateisystems zurückgezogen hat, ist es empfehlenswert, den **`fsck.gfs2`**-Befehl auszuführen, bevor Sie das Dateisystem wieder einhängen. Um in diesem Fall zu verhindern, dass Ihr Dateisystem beim Neustart wieder eingehängt wird, führen Sie die folgenden Schritte aus:

1. Deaktivieren Sie vorübergehend das Startup-Skript auf dem betroffenen Knoten mit dem folgenden Befehl:

   ```
   # chkconfig gfs2 off
   ```

2. Starten Sie den betroffenen Knoten neu und starten Sie die Cluster-Software. Das GFS2-Dateisystem wird nicht eingehängt werden.

3. Hängen Sie das Dateisystem auf allen Knoten im Cluster aus.

4. Führen Sie auf nur einem Knoten den **`fsck.gfs2`**-Befehl auf dem Dateisystem aus, um sicherzustellen, dass das Dateisystem nicht beschädigt ist.

5. Aktivieren Sie wieder das Startup-Skript auf dem betroffenen Knoten mit dem folgenden Befehl:

   ```
   # chkconfig gfs2 on
   ```

6. Hängen Sie das GFS-Dateisystem auf allen Knoten im Cluster neu ein.

Ein Beispiel für eine Inkonsistenz, die zu einem GFS2-Rückzug führt, ist eine fehlerhafte Blockanzahl.

Wenn der GFS-Kernel eine Datei aus einem Dateisystem löscht, entfernt er systematisch sämtliche zu dieser Datei gehörende Daten- und Metadaten-Blöcke. Sobald er damit fertig ist, überprüft er die Blockanzahl. Falls die Blockanzahl nicht eins ist (was bedeutet, dass nur der Festplatten-Inode selbst übrig ist), weist dies auf eine Dateisysteminkonsistenz hin, da die Blockanzahl nicht mit der Liste der gefundenen Blöcke übereinstimmt.

Sie können die GFS2-Rückzugsfunktion außer Kraft setzen, indem Sie das Dateisystem mit der Option **-o errors=panic** einhängen. Wenn diese Option angegeben wird, führen Fehler, die normalerweise zu einem Rückzug des Systems führen, stattdessen zu einer Panic des Systems. Dadurch stoppt die Cluster-Kommunikation des Knotens, woraufhin der Knoten abgegrenzt wird.

Intern funktioniert die GFS2-Rückzugsfunktion, indem der Kernel eine Nachricht an den Daemon **gfs_controld** sendet und den Rückzug fordert. Der Daemon **gfs_controld** führt das **dmsetup**-Programm aus, um das Device-Mapper Error-Ziel unter das Dateisystem zu platzieren, um weitere Zugriffe auf das Blockgerät zu verhindern. Es teilt dem Kernel mit, wenn dieser Vorgang abgeschlossen wurde. Aus diesem Grund wird zur Unterstützung von GFS2 vorausgesetzt, dass immer ein CLVM-Gerät unter GFS2 verwendet wird, denn andernfalls ist es nicht möglich, ein Device-Mapper-Ziel einzufügen.

Der Zweck des Device-Mapper Error-Ziels besteht darin sicherzustellen, dass alle nachfolgenden I/O-Operationen zu einem I/O-Fehler führen, wodurch es dem Dateisystem ermöglicht wird, ordnungsgemäß ausgehängt zu werden. Deshalb ist es im Falle eines Rückzugs völlig normal, eine Reihe von I/O-Fehlern vom Device-Mapper-Gerät in den Systemprotokollen vorzufinden.

Gelegentlich schlägt der Rückzug fehl, falls es dem **dmsetup**-Programm nicht möglich war, das Error-Ziel wie angefordert einzufügen. Dies kann auftreten, wenn zum Zeitpunkt des Rückzugs der Speicher knapp ist und dieser Speicher aufgrund des Fehlers, der den Rückzug überhaupt erst ausgelöst hat, auch nicht neu zugewiesen werden kann.

Ein Rückzug bedeutet nicht immer, dass ein Fehler in GFS2 vorliegt. Manchmal wird die Rückzugsfunktion durch I/O-Fehler im zugrunde liegenden Blockgerät verursacht. Wir empfehlen Ihnen dringend, die Protokolle einzusehen, um zu überprüfen, ob dies der Fall ist.

KAPITEL 5. DIAGNOSE UND BEHEBUNG VON PROBLEMEN MIT GFS2-DATEISYSTEMEN

Dieses Kapitel liefert Informationen über einige häufig auftretende Probleme mit GFS2 sowie deren Lösungen.

5.1. GFS2-DATEISYSTEM ZEIGT NUR GERINGE LEISTUNG

Unter Umständen stellen Sie fest, dass Ihr GFS2-Dateisystem eine geringere Leistung zeigt als ein ext3-Dateisystem. Die GFS2-Leistung kann abhängig von einer Reihe von Faktoren und in bestimmten Anwendungsfällen variieren. Informationen über Probleme mit der GFS2-Leistung finden Sie an zahlreichen Stellen in diesem Handbuch.

5.2. GFS2-DATEISYSTEM HÄNGT SICH AUF UND ERFORDERT NEUSTART EINES KNOTENS

Falls sich Ihr GFS2-Dateisystem aufhängt und darauf ausgeführte Befehle nicht zurückkehren, das System nach einem Neustart eines bestimmten Knotens jedoch wieder einwandfrei läuft, so kann dies ggf. auf ein Sperrproblem oder einen Bug hindeuten. Sollte dieser Fall bei Ihnen eintreten, sollten Sie die folgenden Daten sammeln:

- Den GFS2-Sperr-Dump für das Dateisystem auf jedem Knoten:

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

- Den DLM-Sperr-Dump für das Dateisystem auf jedem Knoten. Sie erhalten diese Informationen mit **dlm_tool**:

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

  In diesem Befehl ist *lsname* der Lockspace-Name, der von DLM für das fragliche Dateisystem genutzt wird. Sie finden diesen Wert in der Ausgabe des **group_tool**-Befehls.

- Die Ausgabe des Befehls **sysrq -t**.

- Die Inhalte der **/var/log/messages**-Datei.

Sobald Sie diese Daten gesammelt haben, können Sie ein Ticket beim Red Hat Support erstellen und die gesammelten Daten einreichen.

5.3. GFS2-DATEISYSTEM HÄNGT SICH AUF UND ERFORDERT NEUSTART ALLER KNOTEN

Falls sich Ihr GFS2-Dateisystem aufhängt und darauf ausgeführte Befehle nicht zurückkehren, das System nach dem Neustart aller Knoten im Cluster jedoch wieder einwandfrei läuft, sollten Sie die folgenden Dinge überprüfen.

- Unter Umständen ist das Fencing fehlgeschlagen. GFS2-Dateisysteme frieren im Falle einer fehlgeschlagenen Abgrenzung ein, um die Datenintegrität zu gewährleisten. Überprüfen Sie die Protokolldateien auf Nachrichten bezüglich einer fehlgeschlagener Abgrenzung zum Zeitpunkt des Ausfalls. Vergewissern Sie sich, dass das Fencing einwandfrei konfiguriert ist.

- Das GFS2-Dateisystem hat sich unter Umständen zurückgezogen. Suchen Sie in den Nachrichtenprotokollen das Wort **`withdraw`** und prüfen Sie, ob Sie Nachrichten und Calltraces von GFS2 finden, die darauf hinweisen, dass das Dateisystem zurückgezogen wurde. Ein Rückzug kann auf ein schadhaftes Dateisystem, einen Speicherausfall oder einem Bug hindeuten. Hängen Sie das Dateisystem aus, aktualisieren Sie das **`gfs2-utils`**-Paket und führen Sie den **`fsck`**-Befehl auf dem Dateisystem durch, um es wieder funktionsfähig zu machen. Reichen Sie beim Red Hat Support ein Support-Ticket ein. Informieren Sie den Support über den GFS2-Rückzug und stellen Sie sosreport-Daten und Protokolle zur Verfügung.

  Weitere Informationen über die GFS2-Rückzugsfunktion finden Sie in Abschnitt 4.14, »Die GFS2-Rückzugsfunktion«.

- Dieser Fehler kann auf ein Sperrproblem oder einen Bug hindeuten. Sammeln Sie Daten, während dieser Fehler das nächste Mal auftritt, und reichen Sie beim Red Hat Support ein Ticket ein, wie in Abschnitt 5.2, »GFS2-Dateisystem hängt sich auf und erfordert Neustart eines Knotens« beschrieben.

5.4. GFS2-DATEISYSTEM WIRD AUF NEU ERSTELLTEM CLUSTER-KNOTEN NICHT EINGEHÄNGT

Falls Sie einen neuen Knoten zum Cluster hinzufügen und feststellen, dass Sie Ihr GFS2-Dateisystem auf diesem Knoten nicht einhängen können, dann haben Sie ggf. weniger Journale auf dem GFS2-Dateisystem als Knoten, die auf das GFS2-Dateisystem zuzugreifen versuchen. Sie benötigen ein Journal pro GFS2-Host, auf dem das Dateisystem eingehängt werden soll (ausgenommen sind GFS2-Dateisysteme, die mit der Einhängeoption **`spectator`** eingehängt wurden, da diese kein Journal erfordern). Sie können mithilfe des **`gfs2_jadd`**-Befehls Journale zu einem GFS2-Dateisystem hinzufügen, wie in Abschnitt 4.7, »Hinzufügen von Journalen zu einem Dateisystem« beschrieben.

5.5. VERBRAUCHTER PLATZ IN LEEREM DATEISYSTEM

Wenn Sie ein leeres GFS2-Dateisystem haben, zeigt der **`df`**-Befehl dennoch an, dass etwas Platz verbraucht ist. Der Grund hierfür liegt in den GFS2-Dateisystem-Journalen, die eine gewisse Menge Speicherplatz verbrauchen (Anzahl der Journale * Journalgröße). Falls Sie ein GFS2-Dateisystem mit einer großen Anzahl an Journalen erstellt haben oder eine große Journalgröße festgelegt haben, dann sehen Sie den Speicherplatz (Anzahl der Journale * Journalgröße) bereits verbraucht, wenn Sie den **`df`**-Befehl ausführen. Selbst wenn Sie weder eine große Anzahl an Journalen noch große Journale konfiguriert haben, werden kleine GFS2-Dateisysteme (in der Größenordnung von 1 GB oder kleiner) mit der standardmäßigen GFS2-Journalgröße dennoch einen großen Teil des Speicherplatzes als verbraucht ausweisen.

KAPITEL 6. KONFIGURIEREN EINES GFS2-DATEISYSTEMS IN EINEM PACEMAKER-CLUSTER

Das folgende Verfahren zeigt eine Überblick über die notwendigen Schritte zur Einrichtung eines Pacemaker-Clusters, der ein GFS2-Dateisystem verwendet.

Nachdem Sie die Cluster-Software, GFS2 und die geclusterten LVM-Pakete auf jedem Knoten installiert haben, starten Sie die Dienste **cman**, **clvmd** und **pacemaker** auf jedem Knoten und erstellen Sie den Pacemaker-Cluster. Sie müssen zudem das Fencing für den Cluster konfigurieren. Informationen über die Konfiguration eines Pacemaker-Clusters finden Sie im Handbuch *Konfiguration des Red Hat High Availability Add-Ons mit Pacemaker*.

1. Setzen Sie den globalen Pacemaker-Parameter **no_quorum_policy** auf **freeze**.

ANMERKUNG

Standardmäßig ist **no-quorum-policy** auf den Wert **stop** festgelegt, was bewirkt, dass bei Verlust des Quorums alle Ressourcen auf der verbleibenden Partition sofort gestoppt werden. In der Regel ist diese Standardeinstellung die sicherste und beste Option, allerdings benötigt GFS2 im Gegensatz zu den meisten Ressourcen ein Quorum, um zu funktionieren. Beim Verlust des Quorums können weder die Applikationen, die die GFS2-Mounts verwenden, noch der GFS2-Mount selbst richtig gestoppt werden. Jegliche Versuche, diese Ressourcen ohne Quorum zu stoppen, werden fehlschlagen und letztlich dazu führen, dass bei jedem Quorum-Verlust der gesamte Cluster abgegrenzt wird.

Um dieses Problem zu umgehen, können Sie bei der Verwendung von GFS2 **no-quorum-policy=freeze** einstellen. Das bedeutet, dass im Falle eines Quorum-Verlusts die verbleibende Partition nichts tun wird, bis das Quorum wieder hergestellt ist.

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

2. Vergewissern Sie sich zunächst, dass der Sperrtyp in der Datei **/etc/lvm/lvm.conf** auf 3 eingestellt ist, um geclusterte Sperren zu unterstützen. Erstellen Sie dann den geclusterten logischen Datenträger und formatieren Sie den Datenträger mit einem GFS2-Dateisystem. Stellen Sie dabei sicher, dass Sie genügend Journale für jeden Knoten in Ihrem Cluster anlegen.

```
# 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. Konfigurieren Sie eine **clusterfs**-Ressource.

   Sie sollten das Dateisystem nicht zur Datei **/etc/fstab** hinzufügen, da es als Pacemaker-Cluster-Ressource verwaltet wird. Einhängepunkte können als Teil der Ressourcenkonfiguration mit **options=*options*** festgelegt werden. Führen Sie den Befehl **pcs resource describe Filesystem** aus, um alle Konfigurationsoptionen zu sehen.

   Dieser Befehl zur Cluster-Ressourcenerstellung gibt die Einhängeoption **noatime** an.

```
# 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. Überprüfen Sie, ob das GFS2-Dateisystem wie erwartet eingehängt wurde.

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

5. (Optional) Starten Sie alle Cluster-Knoten neu, um die GFS2-Persistenz und -Wiederherstellung zu testen.

ANHANG A. GFS2-KONTINGENTVERWALTUNG MIT DEM BEFEHL GFS2_QUOTA

Ab der Red Hat Enterprise Linux 6.1 Release unterstützt GFS2 die standardmäßigen Linux-Funktionen für Festplattenkontingente. Um diese nutzen zu können, müssen Sie das **quota**-RPM installieren. Dies ist die bevorzugte Methode zur Verwaltung von Festplattenkontingenten auf GFS2 und sollte für alle neuen GFS2-Implementierungen, die Kontingente nutzen, verwendet werden. Informationen über die Verwendung der standardmäßigen Linux-Funktionen für Festplattenkontingente finden Sie in Abschnitt 4.5, »Verwalten von GFS2-Festplattenkontingenten«.

In früheren Releases von Red Hat Enterprise Linux erforderte GFS2 den **gfs2_quota**-Befehl zur Verwaltung von Festplattenkontingenten. Dieser Anhang dokumentiert die Verwendung des **gfs2_quota**-Befehls zur Verwaltung von GFS2-Dateisystemkontingenten.

A.1. KONTINGENTE FESTLEGEN MIT DEM BEFEHL GFS2_QUOTA

Zwei Kontingenteinstellungen stehen für jede Benutzer-ID (UID) oder Gruppen-ID (GID) zur Verfügung: eine *harte Grenze* und eine *weiche Grenze* (letztere auch „Warngrenze“ genannt).

Eine harte Grenze ist der Speicherplatz, der maximal in Anspruch genommen werden darf. Das Dateisystem lässt den Benutzer oder die Gruppe nicht mehr als diese festgelegte Menge an Speicherplatz belegen. Ein Wert von *Null* für eine harte Grenze bedeutet, dass keine Grenze eingehalten werden muss.

Der Wert der weichen Grenze liegt in der Regel unter dem der harten Grenze. Das Dateisystem benachrichtigt den Benutzer oder die Gruppe, wenn die weiche Grenze erreicht ist, um sie über die Menge an Speicherplatz, die sie verwenden, zu warnen. Ein Wert von *Null* für eine weiche Grenze bedeutet, dass keine Grenze eingehalten werden muss.

Sie können diese Grenzen mithilfe des Befehls **gfs2_quota** festlegen. Der Befehl muss nur auf einem einzigen Knoten ausgeführt werden, auf dem GFS2 eingehängt ist.

In GFS2-Dateisystemen werden Festplattenkontingente standardmäßig nicht erzwungen. Um die Berechnung von Kontingenten zu aktivieren, verwenden Sie beim Einhängen des GFS2-Dateisystems die Option **quota=** zum **mount**-Befehl, wie in Abschnitt A.4, »Erzwingen von Kontingenten aktivieren/deaktivieren« beschrieben.

A.1.1. Verwendung

Festlegen von Kontingenten, harte Grenze

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

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

Festlegen von Kontingenten, weiche Grenze

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

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

User

Die Benutzer-ID, für die eine Grenze festgelegt werden soll. Dabei kann es sich entweder um einen Benutzernamen aus der Passwortdatei oder um die UID-Nummer handeln.

Group

Die Gruppen-ID, für die eine Grenze festgelegt werden soll. Dabei kann es sich entweder um einen Gruppennamen aus der Gruppendatei oder um die GID-Nummer handeln.

Size

Legt den Wert für die harte oder weiche Grenze fest. Standardmäßig wird dieser Wert in Megabytes angegeben. Die zusätzlichen Flags **-k**, **-s** oder **-b** ändern die Einheit auf Kilobytes, Sektoren bzw. Dateisystemblöcke.

MountPoint

Gibt das GFS2-Dateisystem an, auf dem diese Aktion ausgeführt werden soll.

A.1.2. Beispiele

Dieses Beispiel legt die harte Grenze für den Benutzer *Bert* auf 1024 Megabytes (1 Gigabyte) auf dem Dateisystem **/mygfs2** fest.

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

Dieses Beispiel legt die weiche Grenze für die Gruppen-ID 21 auf 50 Kilobytes auf dem Dateisystem **/mygfs2** fest.

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

A.2. ANZEIGEN VON KONTINGENTGRENZEN UND -VERBRAUCH MIT DEM GFS2_QUOTA-BEFEHL

Sie können Kontingentgrenzen und den derzeitigen Verbrauch für einen bestimmten Benutzer oder eine bestimmte Gruppe mithilfe des Befehls **gfs2_quota get** anzeigen. Sie können sich auch den gesamten Inhalt der Kontingentdatei mithilfe des Befehls **gfs2_quota list** anzeigen lassen, wobei alle IDs angezeigt werden, die eine harte Grenze oder weiche Grenze von ungleich Null aufweisen.

A.2.1. Verwendung

Anzeigen von Kontingentgrenzen für einen Benutzer

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

Anzeigen von Kontingentgrenzen für eine Gruppe

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

Anzeigen der gesamten Kontingentdatei

```
gfs2_quota list -f MountPoint
```

User

Die Benutzer-ID eines bestimmten Benutzers, über den Informationen angezeigt werden sollen. Dabei kann es sich entweder um einen Benutzernamen aus der Passwortdatei oder um die UID-Nummer handeln.

Group

Die Gruppen-ID einer bestimmten Gruppe, über die Informationen angezeigt werden sollen. Dabei kann es sich entweder um einen Gruppennamen aus der Gruppendatei oder um die GID-Nummer handeln.

MountPoint

Gibt das GFS2-Dateisystem an, auf dem diese Aktion ausgeführt werden soll.

A.2.2. Befehlsausgabe

Die GFS2-Kontingentinformationen vom **gfs2_quota**-Befehl wird folgendermaßen dargestellt:

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

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

Die Werte ***LimitSize***, ***WarnSize*** und ***Value*** werden standardmäßig in Megabyte-Einheiten angegeben. Wenn Sie auf der Befehlszeile die Flags **-k**, **-s** oder **-b** hinzufügen, ändert dies die Einheiten auf Kilobytes, Sektoren bzw. Dateisystemblöcke.

User

Ein Benutzername oder eine Benutzer-ID, auf den/die sich diese Daten beziehen.

Group

Ein Gruppenname oder eine Gruppen-ID, auf den/die sich diese Daten beziehen.

LimitSize

Die für den Benutzer oder die Gruppe festgelegte harte Grenze. Dieser Wert ist Null, falls keine Grenze festgelegt wurde.

Value

Der tatsächlich vom Benutzer oder von der Gruppe verwendete Festplattenplatz.

A.2.3. Anmerkungen

Beim Anzeigen der Kontingentinformationen löst der Befehl **gfs2_quota** die UIDs und GIDs nicht in Namen auf, wenn die Option **-n** auf der Befehlszeile hinzugefügt wird.

Der Platz, der den versteckten Dateien von GFS2 zugewiesen ist, kann für die Root-UID und -GID aus der Anzeige herausgelassen werden, indem Sie die Option **-d** auf der Befehlszeile hinzufügen. Dies ist hilfreich, wenn Sie die Werte von **gfs2_quota** mit den Ergebnissen des **du**-Befehls abgleichen möchten.

A.2.4. Beispiele

Dieses Beispiel zeigt Kontingentinformationen für alle Benutzer und Gruppen, für die Grenzen gesetzt wurden oder die Festplattenplatz auf dem Dateisystem **/mygfs2** verwenden.

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

Dieses Beispiel zeigt Kontingentinformationen in Sektoren für die Gruppe **users** auf dem Dateisystem **/mygfs2**.

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

A.3. SYNCHRONISIEREN VON KONTINGENTEN MIT DEM GFS2_QUOTA-BEFEHL

GFS2 speichert sämtliche Kontingentinformationen auf seiner eigenen, internen Datei auf der Festplatte. Ein GFS2-Knoten aktualisiert diese Datei jedoch nicht nach jedem Schreibvorgang auf dem Dateisystem, sondern standardmäßig nur alle 60 Sekunden. Dies ist notwendig, um Konflikte zwischen Knoten zu vermeiden, die andernfalls gleichzeitig in die Kontingentdatei zu schreiben versuchten, was zu Leistungseinbußen führen würde.

Wenn sich ein Benutzer oder eine Gruppe der Kontingentgrenze nähert, reduziert GFS2 die Zeit zwischen den Aktualisierungen der Quotendatei dynamisch, um einer Überschreitung der Grenze vorzubeugen. Der normale Zeitabstand zwischen den Kontingentsynchronisationen ist ein einstellbarer Parameter, **quota_quantum**. Sie können den Standardwert von 60 Sekunden mit der Einhängeoption **quota_quantum=** ändern, wie in Tabelle 4.2, »GFS2-spezifische Einhängeoptionen« beschrieben. Der **quota_quantum**-Parameter muss auf jedem Knoten und jedes Mal, wenn das Dateisystem eingehängt wird, gesetzt werden. Änderungen am **quota_quantum**-Parameter gehen beim Aushängen verloren. Sie können den **quota_quantum**-Wert mit dem Befehl **mount -o remount** aktualisieren.

Sie können den **gfs2_quota sync**-Befehl verwenden, um die Kontingentinformationen zwischen den automatischen Aktualisierungen durch GFS2 von einem Knoten auf die Kontingentdatei auf der Festplatte zu synchronisieren.

A.3.1. Verwendung

Synchronisieren von Kontingentinformationen

```
gfs2_quota sync -f MountPoint
```

MountPoint

Gibt das GFS2-Dateisystem an, auf dem diese Aktion ausgeführt werden soll.

Anpassen der Zeitabstände zwischen Synchronisationen

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

MountPoint

Gibt das GFS2-Dateisystem an, auf dem diese Aktion ausgeführt werden soll.

secs

Gibt den neuen Zeitabstand zwischen den regulären Synchronisationen der Kontingentdatei durch GFS2 an. Kleinere Werte können vermehrt zu Konflikten und zu Leistungseinbußen führen.

A.3.2. Beispiele

Dieses Beispiel synchronisiert die Kontingentinformationen von dem Knoten, auf dem der Befehl ausgeführt wird, auf das Dateisystem **/mygfs2**.

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

In diesem Beispiel wird der standardmäßige Zeitabstand zwischen regelmäßigen Aktualisierungen der Kontingentdatei auf eine Stunde (3600 Sekunden) für das Dateisystem **/mnt/mygfs2** beim Wiedereinhängen dieses Dateisystems auf dem logischen Datenträger **/dev/volgroup/logical_volume** geändert.

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

A.4. ERZWINGEN VON KONTINGENTEN AKTIVIEREN/DEAKTIVIEREN

In GFS2-Dateisystemen werden Festplattenkontingente standardmäßig nicht erzwungen. Um Kontingente für ein Dateisystem zu erzwingen, geben Sie beim Einhängen des Dateisystems die Option **quota=on** an.

A.4.1. Verwendung

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

Um ein Dateisystem ohne das Erzwingen von Kontingenten einzuhängen, geben Sie beim Einhängen des Dateisystems die Option **quota=off** an. Dies ist die Standardeinstellung.

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

-o quota={on|off}

Gibt an, ob das Erzwingen von Kontingenten beim Einhängen des Dateisystems aktiviert oder deaktiviert werden soll.

BlockDevice

Gibt das Blockgerät an, auf dem sich das GFS2-Dateisystem befindet.

MountPoint

Gibt das Verzeichnis an, unter dem das GFS2-Dateisystem eingehängt werden soll.

A.4.2. Beispiele

In diesem Beispiel wird das GFS2-Dateisystem auf **/dev/vg01/lvol0** unter dem Verzeichnis **/mygfs2** eingehängt und das Erzwingen von Kontingenten ist aktiviert.

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

A.5. AKTIVIEREN DER KONTINGENTBERECHNUNG

Es ist möglich, den Verbrauch von Festplattenplatz nachzuverfolgen und für jeden Benutzer und jede Gruppe Kontingentberechnungen anzustellen, ohne dabei die weichen und harten Grenzen zu erzwingen. Hängen Sie dafür das Dateisystem mit der Option **quota=account** ein.

A.5.1. Verwendung

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

-o quota=account

Gibt an, dass Verbrauchsstatistiken für Benutzer- und Gruppen vom Dateisystem gepflegt werden, ohne jedoch die Kontingentgrenzen zu erzwingen.

BlockDevice

Gibt das Blockgerät an, auf dem sich das GFS2-Dateisystem befindet.

MountPoint

Gibt das Verzeichnis an, unter dem das GFS2-Dateisystem eingehängt werden soll.

A.5.2. Beispiel

In diesem Beispiel wird das GFS2-Dateisystem auf **/dev/vg01/lvol0** unter dem Verzeichnis **/mygfs2** eingehängt und die Berechnung von Kontingenten ist aktiviert.

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

ANHANG B. KONVERTIEREN EINES DATEISYSTEMS VON GFS AUF GFS2

Da die Red Hat Enterprise Linux 6 Release keine GFS-Dateisysteme unterstützt, müssen Sie jegliche vorhandenen GFS-Dateisysteme mithilfe des **gfs2_convert**-Befehls in GFS2-Dateisysteme konvertieren. Beachten Sie, dass Sie diesen Konvertierungsprozess auf dem Red Hat Enterprise Linux 5 System ausführen müssen, noch bevor Sie auf Red Hat Enterprise Linux 6 aktualisieren.

WARNUNG

Bevor Sie das GFS-Dateisystem konvertieren, müssen Sie eine Sicherungskopie des Dateisystems anlegen, da der Konvertierungsprozess nicht umkehrbar ist und möglicherweise auftretende Fehler während der Konvertierung zu einem plötzlichen Abbruch des Programms und infolgedessen zu einem irreparabel beschädigten Dateisystem führen können.

Bevor Sie das GFS-Dateisystem konvertieren, verwenden Sie den **gfs_fsck**-Befehl, um das Dateisystem zu überprüfen und eventuelle Fehler zu beseitigen.

Falls die Konvertierung von GFS zu GFS2 durch einen Stromausfall oder ähnliches unterbrochen wird, starten Sie das Konvertierungstool neu. Versuchen Sie nicht, den **fsck.gfs2**-Befehl auf dem Dateisystem auszuführen, ehe der Konvertierungsvorgang abgeschlossen wurde.

Wenn Sie ein volles oder beinahe volles Dateisystem konvertieren, bietet dies unter Umständen nicht mehr ausreichend Platz für die GFS2-Dateisystemstrukturen. In diesem Fall wird die Größe aller Journale gleichmäßig reduziert, sodass alles in den verfügbaren Platz passt.

B.1. KONVERTIERUNG KONTEXTABHÄNGIGER PFADE

GFS2-Dateisysteme unterstützen keine kontextabhängigen Pfade (Context-Dependent Path Names oder kurz CDPNs), die es Ihnen ermöglichen, symbolische Links zu erzeugen, die auf verschiedene Zieldateien oder -verzeichnisse zeigen. Um dieselbe Funktionalität wie kontextabhängige Pfade in GFS2-Dateisystemen zu erreichen, können Sie die **bind**-Option des **mount**-Befehls verwenden.

Der **gfs2_convert**-Befehl identifiziert kontextabhängige Pfade und ersetzt sie durch leere Verzeichnisse desselben Namens. Um Bind Mounts als Ersatz für kontextabhängige Pfade zu konfigurieren, benötigen Sie die vollständigen Pfade der Link-Ziele aller kontextabhängigen Pfade, die Sie ersetzen möchten. Bevor Sie Ihr Dateisystem konvertieren, können Sie mithilfe des **find**-Befehls die Links identifizieren.

Der folgende Befehl listet die symbolischen Links auf, die auf den kontextabhängigen Pfad **hostname** verweisen:

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

Auf die gleiche Weise können Sie den **find**-Befehl für andere kontextabhängige Pfade ausführen

(**mach**, **os**, **sys**, **uid**, **gid**, **jid**). Beachten Sie, dass kontextabhängige Pfade entweder in der Form **@hostname** oder **{hostname}** vorliegen können, Sie müssen den **find**-Befehl also für jeweils beide Varianten ausführen.

Weitere Informationen über Bind Mounts und kontextabhängige Pfade in GFS2 finden Sie in Abschnitt 4.12, »Bind Mounts und kontextabhängige Pfade«.

B.2. VERFAHREN ZUR KONVERTIERUNG VON GFS IN GFS2

Verwenden Sie folgendes Verfahren, um ein GFS-Dateisystem in ein GFS2-Dateisystem zu konvertieren.

1. Erstellen Sie auf einem Red Hat Enterprise Linux System eine Datensicherung Ihres vorhandenen GFS-Dateisystems.
2. Hängen Sie das GFS-Dateisystem auf allen Knoten im Cluster aus.
3. Führen Sie den **gfs_fsck**-Befehl auf dem GFS-Dateisystem aus, um sicherzustellen, dass das Dateisystem fehlerfrei ist.
4. Führen Sie den Befehl **gfs2_convert *gfsfilesystem*** aus. Das System zeigt daraufhin Warnmeldungen und fordert zur Bestätigung auf, bevor das ***gfsfilesystem*** nach GFS2 konvertiert wird.
5. Aktualisieren Sie auf Red Hat Enterprise Linux 6.

Das folgende Beispiel konvertiert ein GFS-Dateisystem auf dem Blockgerät **/dev/shell_vg/500g** in ein GFS2-Dateisystem.

```
[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.
```

ANHANG C. GFS2-TRACEPOINTS UND DIE DEBUGFS-GLOCKS-DATEI

Dieser Anhang beschreibt die Glock-**debugfs**-Schnittstelle und die GFS2-Tracepoints. Er richtet sich an fortgeschrittene Benutzer, die mit Dateisysteminterna vertraut sind und die gerne mehr über den Aufbau von GFS2 und das Debugging von GFS2-spezifischen Fehlern erfahren möchten.

C.1. ARTEN VON GFS2-TRACEPOINTS

Derzeit gibt es drei Arten von GFS2-Tracepoints: *Glock*-Tracepoints (ausgesprochen: „Dschie-Lock“), *Bmap*-Tracepoints (Block Map) und *Log*-Tracepoints. Diese können verwendet werden, um ein laufendes GFS2-Dateisystem zu überwachen und zusätzliche Informationen zu jenen zu erhalten, die mit den Debugging-Optionen, die in früheren Versionen von Red Hat Enterprise Linux unterstützt wurden, erhalten werden können. Tracepoints (Punkte zur Ablaufverfolgung) sind besonders nützlich, wenn ein Problem – wie ein Aufhänger oder ein Leistungsproblem – reproduzierbar ist und daher eine Tracepoint-Ausgabe während der problematischen Operation erhalten werden kann. In GFS2 sind Glocks der primäre Mechanismus zur Cache-Steuerung und der Schlüssel zum Verständnis der Kernleistung von GFS2. Die Bmap-Tracepoints können verwendet werden, um Blockzuweisungen und Block-Mapping (Lookup von bereits zugewiesenen Blöcken in der Metadatenstruktur der Festplatte) zu überwachen und eventuelle Probleme in Bezug auf den Ort des Zugangs zu kontrollieren. Die Log-Tracepoints verfolgen die Daten, die in das Journal geschrieben und von ihm freigegeben werden, und können nützliche Informationen über diesen Aspekt von GFS2 liefern.

Die Tracepoints sind darauf ausgelegt, so allgemein wie möglich zu sein. Das bedeutet, dass es nicht nötig sein sollte, die API im Laufe von Red Hat Enterprise Linux 6 zu ändern. Auf der anderen Seite sollten sich die Nutzer dieser Schnittstelle bewusst sein, dass sie eine Debugging-Schnittstelle und nicht Teil des normalen Red Hat Enterprise Linux 6 API-Sets ist, weshalb Red Hat nicht garantieren kann, dass keinerlei Veränderungen in der GFS2-Tracepoints-Schnittstelle vorkommen.

Tracepoints sind eine generische Funktion von Red Hat Enterprise Linux 6 und ihr Anwendungsbereich geht weit über GFS2 hinaus. Sie werden insbesondere zur Implementierung der **blktrace**-Infrastruktur verwendet und die **blktrace**-Tracepoints können zusammen mit jenen von GFS2 verwendet werden, um ein umfassenderes Bild der Systemleistung zu erhalten. Aufgrund ihrer Funktionsebene können Tracepoints innerhalb von kürzester Zeit große Datenmengen produzieren. Zwar wurden sie so konzipiert, dass die Belastung des Systems möglichst gering ist, doch eine gewisse Auswirkung ist unvermeidlich. Es kann hilfreich sein, die Ereignisse mithilfe einer Vielzahl von Instrumenten zu filtern und so das Volumen der Daten zu verringern, um das Hauptaugenmerk auf jene Informationen zu richten, die nützlich für das Verständnis der jeweiligen Situation sind.

C.2. TRACEPOINTS

Sie finden die Tracepoints im Verzeichnis **/sys/kernel/debug/tracing/**, sofern **debugfs** im standardmäßigen Verzeichnis **/sys/kernel/debug** eingehängt ist. Das **events**-Unterverzeichnis enthält alle Ereignisse, die nachverfolgt werden können. Sofern das **gfs2**-Modul geladen ist, gibt es ebenfalls ein **gfs2**-Unterverzeichnis mit weiteren Unterverzeichnissen, eins für jedes GFS2-Ereignis. Der Inhalt des Verzeichnisses **/sys/kernel/debug/tracing/events/gfs2** sieht etwa wie folgt aus:

```
[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
```

Um alle GFS2-Tracepoints zu aktivieren, führen Sie den folgenden Befehl aus:

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

Um einen bestimmten Tracepoint zu aktivieren, gibt es eine **enable**-Datei in jedem der Ereignis-Unterverzeichnisse. Das gleiche gilt für die **filter**-Datei, die verwendet werden kann, um eine Filterfunktion für jedes Ereignis oder eine Reihe von Ereignissen einzustellen. Die Bedeutung der einzelnen Ereignisse wird im Folgenden näher erläutert.

Die Ausgabe der Tracepoints ist in ASCII-oder Binär-Format verfügbar. Dieser Anhang behandelt derzeit nicht die binäre Schnittstelle. Die ASCII-Schnittstelle ist auf zwei Arten verfügbar. Um den aktuellen Inhalt des Ringpuffers anzuzeigen, führen Sie den folgenden Befehl aus:

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

Diese Schnittstelle ist nützlich in Fällen, in denen Sie einen lang andauernden Prozess für einen bestimmten Zeitraum verwenden und nach einem bestimmten Ereignis einen Rückblick auf die letzten erfassten Informationen im Puffer wollen. Eine alternative Schnittstelle, **/sys/kernel/debug/tracing/trace_pipe**, kann verwendet werden, wenn die gesamte Ausgabe benötigt wird. Die Ereignisse werden aus dieser Datei gelesen, wie sie auftreten, über diese Schnittstelle sind keine historischen Informationen verfügbar. Das Format der Ausgabe ist dasselbe für beide Schnittstellen und wird für jedes der GFS2-Ereignisse in den späteren Abschnitten dieses Anhangs beschrieben.

Zum Lesen der Tracepoint-Daten steht ein Dienstprogramm namens **trace-cmd** zur Verfügung. Weitere Informationen zu diesem Dienstprogramm finden Sie unter dem Link in Abschnitt C.10, »Verweise«. Das **trace-cmd**-Dienstprogramm kann in ähnlicher Weise wie das **strace**-Dienstprogramm verwendet werden, zum Beispiel zur Ausführung eines Befehls, während Trace-Daten aus verschiedenen Quellen gesammelt werden.

C.3. GLOCKS

Für das Verständnis von GFS2 ist es notwendig, das wichtigste Konzept zu verstehen, das es von anderen Dateisystemen unterscheidet: Das Konzept der Glocks. In Quellcode-Begriffen ist ein Glock eine Datenstruktur, die DLM und Caching in einem Single-State-Rechner vereinigt. Jedes Glock hat eine 1:1-Beziehung mit einer einzigen DLM-Sperre und bietet Caching für diesen Sperrstatus, sodass wiederholt durchgeführte Operationen von einem einzelnen Knoten des Dateisystems nicht immer wieder den DLM aufrufen müssen, was unnötigen Netzwerkverkehr vermeidet. Es gibt zwei wesentliche Kategorien von Glocks: Solche, die Metadaten cachen, und solche, die dies nicht tun. Die Inode-Glocks und die Ressourcengruppen-Glocks cachen beide Metadaten, andere Arten von Glocks cachen Metadaten nicht. Der Inode-Glock cacht darüberhinaus auch Daten und hat von allen Glocks die aufwendigste Logik.

Tabelle C.1. Glock-Modus und DLM-Sperrmodus

Glock-Modus	DLM-Sperrmodus	Anmerkungen
UN	IV/NL	Nicht gesperrt (keine DLM-Sperre zu Glock oder NL-Sperre zugewiesen, abhängig vom I-Flag)
SH	PR	Gemeinsame Sperre (geschütztes Lesen)

Glock-Modus	DLM-Sperrmodus	Anmerkungen
EX	EX	Exklusive Sperre
DF	CW	Zeitversetzt (gleichzeitiges Schreiben), verwendet für Direct I/O und Dateisystem-Freeze

Glocks bleiben im Speicher, bis sie entsperrt werden (auf Anforderung eines anderen Knotens oder auf Anforderung der VM) und es keine lokalen Benutzer gibt. Zu diesem Zeitpunkt werden sie aus der Glock-Hash-Tabelle entfernt und freigegeben. Beim Erstellen eines Glocks wird die DLM-Sperre nicht sofort mit dem Glock verknüpft. Die DLM-Sperre wird bei der ersten Anforderung an den DLM mit dem Glock verknüpft und wenn diese Anforderung erfolgreich ist, dann wird das Flag „I“ (Initial) auf dem Glock gesetzt. Tabelle C.4, »Glock-Flags« zeigt die Bedeutungen der verschiedenen Glock-Flags. Sobald der DLM mit dem Glock verknüpft wurde, verbleibt die DLM-Sperre immer zumindest im Sperrmodus NL (Null), bis das Glock freigegeben wird. Eine Herabstufung der DLM-Sperre von NL auf entsperrt ist immer die letzte Operation im Leben eines Glocks.

ANMERKUNG

Dieser besondere Aspekt des DLM-Sperrverhaltens hat sich seit Red Hat Enterprise Linux 5 verändert, in dem manchmal die mit Glocks verknüpften DLM-Sperren völlig entsperrt werden. Daher hat Red Hat Enterprise Linux 5 einen anderen Mechanismus, um sicherzustellen, dass LVbs (Lock Value Blocks) bewahrt werden, wo erforderlich. Die neue Regelung, die in Red Hat Enterprise Linux 6 verwendet wird, wurde erst durch die Integration des **`lock_dlm`**-Sperrmoduls (nicht mit dem DLM selbst zu verwechseln) in GFS2 ermöglicht.

Jedem Glock kann eine Reihe von „Haltern“ zugeordnet sein, von denen jeder eine Sperranforderung von den höheren Schichten darstellt. Die Systemaufrufe für GFS2 fügen Halter in die Warteschlange für dieses Glock ein oder entfernen sie daraus, um den kritischen Abschnitt des Codes zu schützen.

Die Glock State Machine basiert auf einer Arbeitswarteschlange. Aus Effizienzgründen wären Tasklets vorzuziehen, aber in der aktuellen Implementierung muss I/O aus diesem Kontext gesendet werden, was deren Verwendung verhindert.

ANMERKUNG

Arbeitswarteschlangen haben ihre eigenen Tracepoints, die bei Bedarf in Kombination mit den GFS2-Tracepoints verwendet werden können.

Tabelle C.2, »Glock-Modi und Datentypen« zeigt, welcher Status bei jedem Glock-Modus gecacht werden kann und ob dieser gecachte Status verändert („dirty“) sein darf. Dies gilt sowohl für Inode- als auch für Ressourcengruppen-Sperren, obwohl es keine Datenkomponente für die Ressourcengruppen-Sperren gibt, nur Metadaten.

Tabelle C.2. Glock-Modi und Datentypen

Glock-Modus	Cache-Daten	Cache-Metadaten	Veränderte Daten	Veränderte Metadaten
UN	Nein	Nein	Nein	Nein
SH	Ja	Ja	Nein	Nein
DF	Nein	Ja	Nein	Nein
EX	Ja	Ja	Ja	Ja

C.4. DIE GLOCK-DEBUGFS-SCHNITTSTELLE

Die glock-**debugfs**-Schnittstelle ermöglicht die Visualisierung des internen Status der Glocks und der Halter; sie enthält in einigen Fällen auch einige zusammenfassende Daten über die Objekte, die gesperrt sind. Jede Zeile der Datei beginnt entweder mit G: ohne Einrückung und bezieht sich auf das Glock selbst, oder aber mit einem anderen Buchstaben, um ein Leerzeichen eingerückt, und bezieht sich auf die Strukturen, die mit dem Glock in der Datei unmittelbar darüber verknüpft sind (H: ist ein Halter, I: ein Inode und R: eine Ressourcengruppe). Nachfolgend sehen Sie ein Beispiel dafür, wie der Inhalt dieser Datei aussehen könnte:

```
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]
```

Das obige Beispiel zeigt eine Reihe von Auszügen (aus einer etwa 18 MB Datei), die durch den Befehl **cat /sys/kernel/debug/gfs2/unity:myfs/glocks >my.lock** in einem Durchlauf der Postmark Benchmark auf einem Ein-Knoten-GFS2-Dateisystem erstellt wurden. Die Glocks in der Abbildung wurden ausgewählt, um einige der interessantesten Merkmale der Glock-Dumps zu zeigen.

Der Glock-Status ist entweder EX (exklusiv), DF (zurückgestellt), SH (gemeinsam verwendet) oder UN (nicht gesperrt). Diese Status entsprechen direkt den DLM-Sperrmodi mit Ausnahme von UN, was entweder den DLM-Null-Sperrstatus darstellen kann oder dass GFS2 nicht im Besitz einer DLM-Sperre ist (abhängig von dem I-Flag, wie oben erläutert). Das s: Feld des Glocks zeigt den aktuellen Status der Sperre an; das gleiche Feld im Halter zeigt den angeforderten Modus an. Wenn die Sperre gewährt ist, ist beim Halter das H-Bit in den Flags (f: Feld) gesetzt, andernfalls ist das W-Bit (warten) gesetzt.

Das n: Feld (eine Zahl) gibt die Nummer an, die jedem Element zugeordnet ist. Für Glocks ist das die Typnummer gefolgt von der Glock-Nummer, sodass im obigen Beispiel das erste Glock n:5/75320 ist, das heißt, ein **iopen**-Glock, das den Inode 75320 betrifft. Im Fall von Inode und **iopen**-Glocks entspricht die Glock-Nummer immer der Festplatten-Blocknummer des Inodes.

ANMERKUNG

Die Glock-Nummern (n: Feld) in der debugfs-Glocks-Datei sind in Hexadezimalwerten angegeben, während die Tracepoints-Ausgabe sie in Dezimalwerten aufführt. Dies hat historische Gründe; Glock-Nummern wurden schon immer in Hexadezimalwerten angegeben, aber für die Tracepoints wurden Dezimalwerte gewählt, sodass die Nummern leicht mit anderen Tracepoint-Ausgaben (zum Beispiel von **blktrace**) und mit der Ausgabe von **stat**(1) verglichen werden können.

Eine vollständige Liste aller Flags sowohl für den Halter als auch den Glock sehen Sie in Tabelle C.4, »Glock-Flags« und Tabelle C.5, »Glock-Halter-Flags«. Der Inhalt der Sperrwertblöcke steht derzeit nicht über die Glock-**debugfs**-Schnittstelle zur Verfügung.

Tabelle C.3, »Glock-Typen« zeigt die Bedeutung der verschiedenen Glock-Typen.

Tabelle C.3. Glock-Typen

Typnummer	Sperrtyp	Verwendung
1	trans	Transaktionssperre
2	Inode	Inode-Metadaten und -Daten
3	rgrp	Ressourcengruppen-Metadaten
4	meta	Der Superblock
5	iopen	Feststellung der letzten Schließers des Inodes
6	flock	**flock**(2)-Systemaufruf
8	quota	Kontingentoperationen
9	journal	Journal-Mutex

Einer der wichtigsten Glock-Flags ist das l-Flag (gesperrt). Dies ist die Bit-Sperre, mit der der Zugriff auf den Glock-Status vermittelt wird, wenn ein Statuswechsel durchgeführt werden soll. Es wird gesetzt, wenn die State Machine eine externe Sperranforderung über den DLM zu senden bereit ist, und wird erst gelöscht, wenn der komplette Vorgang abgeschlossen wurde. Manchmal kann dies dazu führen, dass mehr als eine Sperranforderung gesendet wurde mit verschiedenen Invalidierungen dazwischen.

Tabelle C.4, »Glock-Flags« zeigt die Bedeutung der verschiedenen Glock-Flags.

Tabelle C.4. Glock-Flags

Flag	Name	Bedeutung
d	Pending demote	Eine wartende Anfrage zum Herabstufen (remote)
D	Demote	Eine Anfrage zum Herabstufen (lokal oder remote)
f	Log flush	Das Protokoll muss festgeschrieben werden, bevor dieses Glock freigegeben werden kann
F	Frozen	Antworten von Remote-Knoten werden ignoriert, eine Wiederherstellung läuft.
i	Invalidate in progress	Seiten unter diesem Glock werden derzeit ungültig gemacht (invalidiert)
I	Initial	Gesetzt, wenn eine DLM-Sperre mit diesem Glock verknüpft ist
l	Locked	Das Glock ändert derzeit seinen Status
L	LRU	Gesetzt, wenn das Glock auf der LRU-Liste ist
o	Object	Gesetzt, wenn das Glock einem Objekt zugeordnet ist (d. h. einem Inode für Typ-2-Glocks und einer Ressourcengruppe für Typ-3-Glocks)
p	Demote in progress	Das Glock antwortet derzeit auf eine Anfrage zum Herabstufen
q	Queued	Gesetzt, wenn ein Halter der Warteschlange eines Glocks hinzugefügt wird, und gelöscht, wenn das Glock noch gehalten wird, es jedoch keine verbleibenden Halter gibt. Verwendet als Teil des Algorithmus, der die minimale Haltezeit für ein Glock berechnet.
r	Reply pending	Von Remote-Knoten erhaltene Antwort wartet auf Verarbeitung
y	Dirty	Daten müssen auf die Festplatte überschrieben werden, bevor dieses Glock freigegeben werden kann

Wenn ein Remote-Callback von einem Knoten empfangen wird, der eine Sperre in einem Modus erhalten will, der mit dem Modus des lokalen Knoten kollidiert, dann wird entweder das Flag D (demote) oder d (demote pending) gesetzt. Um bei diesen Konflikten um eine Sperre das „Verhungern“ (engl.: starvation) zu verhindern, wird jeder Sperre eine Mindesthaltezeit zugewiesen. Ein Knoten, der die Sperre noch nicht für die Mindesthaltezeit gehalten hat, darf diese Sperre behalten, bis diese Zeitspanne abgelaufen ist.

Wenn die Zeit abgelaufen ist, wird das D (demote) Flag gesetzt und der benötigte Status wird aufgezeichnet. In diesem Fall wird, wenn es das nächste Mal keine erteilten Sperren in der Halter-Warteschlange gibt, die Sperre herabgestuft werden. Wenn das Zeitintervall noch nicht abgelaufen ist,

dann wird stattdessen das d (demote pending) Flag gesetzt. Dies weist auch die State Machine dazu an, das Flag d (demote pending) zu löschen und stattdessen D (demote) zu setzen, sobald die Mindesthaltezeit abgelaufen ist.

Das I (initial) Flag wird gesetzt, wenn dem Glock eine DLM-Sperre zugewiesen wurde. Dies geschieht, wenn das Glock zum ersten Mal verwendet wird; das I Flag bleibt anschließend gesetzt, bis das Glock schließlich freigegeben wird (was die DLM-Sperre aufhebt).

C.5. GLOCK-HALTER

Tabelle C.5, »Glock-Halter-Flags« zeigt die Bedeutung der verschiedenen Glock-Halter-Flags.

Tabelle C.5. Glock-Halter-Flags

Flag	Name	Bedeutung
a	Async	Nicht auf das Glock-Ergebnis warten (Ergebnis wird später abgerufen)
A	Any	Jeder kompatible Sperrmodus ist zulässig
c	No cache	Wenn nicht gesperrt, sofort DLM-Sperre herabstufen
e	No expire	Nachfolgende Anfragen zur Aufhebung der Sperre ignorieren
E	Exact	Muss den exakten Sperrmodus haben
F	First	Gesetzt, wenn der Halter der Erste ist, dem diese Sperre gewährt wird
H	Holder	Zeigt an, dass die angeforderte Sperre gewährt wird
p	Priority	Reiht Halter an der Spitze der Warteschlange ein
t	Try	Eine „try“-Sperre
T	Try 1CB	Eine „try“-Sperre, die einen Callback sendet
W	Wait	Gesetzt, während auf den Abschluss einer Anfrage gewartet wird

Die wichtigsten Halter-Flags sind, wie bereits erwähnt, H (holder) und W (wait), da sie auf gewährte Sperranforderungen bzw. in der Warteschlange befindliche Sperranforderungen gesetzt werden. Die Reihenfolge der Halter in der Liste ist wichtig. Wenn es gewährte Halter gibt, werden sie immer an der Spitze der Warteschlange sein, gefolgt von jeglichen wartenden Haltern.

Wenn es keine gewährten Halter gibt, dann wird der erste Halter in der Liste derjenige sein, der die nächste Statusveränderung auslöst. Da Herabstufungsanfragen immer eine höhere Priorität als Anfragen aus dem Dateisystem haben, muss dies nicht immer direkt zu der angeforderten Statusänderung führen.

Das Glock-Subsystem unterstützt zwei Arten der „try“-Sperre. Diese sind hilfreich, weil sie erstens die

Entnahme von Sperren außerhalb der normalen Reihenfolge (mit angemessenem Back-off und Wiederholung) ermöglichen, und zweitens bei der Vermeidung von Ressourcen helfen, die von anderen Knoten verwendet werden. Die normale t (try) Sperre ist im Grunde genau wie der Name schon sagt eine „Versuchssperre“, die nichts besonderes macht. Die T (**`try 1CB`**) Sperre ist identisch mit der t-Sperre, mit dem Unterschied, dass der DLM einen einzelnen Callback an die derzeitigen Halter der inkompatiblen Sperre sendet. Ein Anwendungsfall der T (**`try 1CB`**) Sperre ist zusammen mit den **`iopen`**-Sperren, die verwendet werden, um zwischen den Knoten zu vermitteln, wenn die **`i_nlink`**-Anzahl eines Inodes Null ist, um zu bestimmen, welche der Knoten für das Freigeben des Inodes verantwortlich ist. Das **`iopen`**-Glock wird normalerweise im gemeinsam verwendeten Status gehalten, wenn jedoch die **`i_nlink`**-Anzahl auf Null sinkt und **`->delete_inode`**() aufgerufen wird, fragt es eine exklusive Sperre mit gesetztem T (**`try 1CB`**) Flag an. Es wird weiterhin den Inode freigeben, wenn die Sperre gewährt wird. Wenn die Sperre nicht gewährt wird, kennzeichnen die Knoten, die die Gewährung der Sperre verhinderten, ihre Glocks mit dem D (demote) Flag, das beim Zeitpunkt von **`->drop_inode`**() kontrolliert wird, um sicherzustellen, dass die Aufhebung der Zuordnung nicht vergessen wird.

Dies bedeutet, dass Inodes, die eine Linkanzahl von Null haben, aber noch offen sind, durch denjenigen Knoten aufgehoben werden, auf dem das endgültige **`close`**() stattfindet. Zur selben Zeit, wie der Linkzähler des Inodes auf Null gesetzt wird, wird der Inode für den speziellen Status gekennzeichnet, dass dieser zwar die Linkanzahl Null hat, aber immer noch in der Ressourcengruppen-Bitmap im Einsatz ist. Dies ähnelt der Waisenliste (engl: orphan list) im ext3-Dateisystem insofern, als es jedem späteren Leser der Bitmap mitteilt, dass es möglicherweise Platz gibt, der zurückgefordert werden könnte, und diesen dazu auffordert, den Platz zurückzufordern.

C.6. GLOCK-TRACEPOINTS

Die Tracepoints wurden auch konzipiert, um die Richtigkeit der Cache-Steuerung zu bestätigen, indem sie mit der blktrace-Ausgabe und mit der Kenntnis des Layouts der Festplatte kombiniert werden. Es ist dann möglich zu überprüfen, dass eine bestimmte I/O unter der richtigen Sperre ausgegeben und abgeschlossen wurde und dass keine Race-Bedingungen vorliegen.

Von allen Tracepoints ist es am wichtigsten, den Tracepoint **`gfs2_glock_state_change`** zu verstehen. Er verfolgt jede Statusänderung des Glocks von der ursprünglichen Erstellung bis hin zur endgültigen Herabstufung, die mit **`gfs2_glock_put`** und dem endgültigen Übergang von NL auf entsperrt endet. Das Glock-Flag l (locked) ist immer gesetzt, bevor eine Statusänderung erfolgt, und wird erst gelöscht, nachdem diese abgeschlossen ist. Während einer Statusänderung gibt es niemals gewährte Halter (das Glock-Halter-Flag H). Wenn es Halter in der Warteschlange gibt, werden sie immer im W (waiting) Status sein. Wenn die Statusänderung abgeschlossen ist, dann können die Halter gewährt werden, welches die letzte Operation ist, bevor das l-Glock-Flag gelöscht wird.

Der Tracepoint **`gfs2_demote_rq`** verfolgt Herabstufungsanfragen, sowohl lokal als auch remote. Vorausgesetzt, auf dem Knoten ist genügend Speicher vorhanden, dann treten die lokalen Herabstufungsanfragen nur selten auf. Meistens werden sie vom umount-Prozess oder durch gelegentliche Speicherrückforderungen erstellt. Die Anzahl der Remote-Herabstufungsanfragen ist ein Maß für den Konflikt zwischen den Knoten für eine bestimmte Inode- oder Ressourcengruppe.

Wenn einem Halter eine Sperre erteilt wird, wird **`gfs2_promote`** aufgerufen. Dies geschieht in der Endphase einer Statusänderung oder wenn eine Sperre angefordert wird, die sofort gewährt werden kann, da der Glock-Status bereits eine Sperre mit einem geeigneten Modus im Cache vorhält. Falls der Halter der erste ist, der auf diesem Glock gewährt wird, dann wird das f (first) Flag auf diesem Halter gesetzt. Dies wird derzeit nur von Ressourcengruppen verwendet.

C.7. BMAP-TRACEPOINTS

Die Blockzuweisung ist eine Aufgabe von zentraler Bedeutung für jedes Dateisystem. GFS2 verwendet ein herkömmliches Bitmap-basiertes System mit zwei Bits pro Block. Der Hauptzweck der Tracepoints in diesem Subsystem besteht darin, die Zeit zu überwachen, die das Zuweisen und Eintragen der Blöcke in Anspruch nimmt.

Der Tracepoint **gfs2_bmap** wird zweimal für jede bmap-Operation aufgerufen: einmal zu Beginn, um die bmap-Anforderung anzuzeigen, und einmal am Ende, um das Ergebnis anzuzeigen. Dies macht es einfach, die Anforderungen und Ergebnisse einander zuzuordnen und die Zeit zu messen, die benötigt wurde, um Blöcke in verschiedenen Teilen des Dateisystems, in verschiedenen Datei-Offsets oder sogar in verschiedenen Dateien zuzuweisen. Es ist auch möglich festzustellen, welche Größen physischer Extents durchschnittlich zurückgegeben wird im Vergleich zu denen, die angefordert werden.

Um den Überblick über die zugewiesenen Blöcke zu behalten, wird **gfs2_block_alloc** nicht nur bei Zuweisungen, sondern auch bei der Freigabe von Blöcken aufgerufen. Da die Zuweisungen anhand des Inodes referenziert werden, für den der Block beabsichtigt ist, kann dies dazu verwendet werden, um nachzuverfolgen, welche physischen Blöcke zu welchen Dateien in einem aktiven Dateisystem gehören. Dies ist besonders in Kombination mit **blktrace** nützlich, was problematische I/O-Muster zeigen kann, die dann den jeweiligen Inodes zugeordnet werden können, die über diesen Tracepoint herausgefunden werden können.

C.8. LOG-TRACEPOINTS

Die Tracepoints in diesem Subsystem verfolgen Blöcke, die zum Journal hinzugefügt und daraus entfernt werden (**gfs2_pin**), sowie die benötigte Zeit, um die Transaktionen in das Protokoll zu übergeben (**gfs2_log_flush**). Dies kann bei der Fehlersuche für Leistungsprobleme beim Journaling sehr nützlich sein.

Der Tracepoint **gfs2_log_blocks** verfolgt die reservierten Blöcke im Protokoll, was beispielsweise helfen kann aufzuzeigen, dass das Protokoll für die Auslastung zu klein ist.

Der Tracepoint **gfs2_ail_flush** (Red Hat Enterprise Linux 6.2 und höher) ähnelt dem Tracepoint **gfs2_log_flush** insofern, als er den Beginn und das Ende der Leerung der AIL-Liste verfolgt. Die AIL-Liste enthält Puffer, die durch das Protokoll gegangen sind, aber noch nicht wieder an ihren Ort zurückgeschrieben wurden; diese werden regelmäßig entleert, um mehr Protokollspeicherplatz für die Verwendung durch das Dateisystem freizugeben oder wenn ein Prozess ein sync oder fsync anfordert.

C.9. GLOCK-STATISTIKEN

GFS2 pflegt Statistiken, die dabei helfen, die Vorgänge im Dateisystem nachzuverfolgen. Dies ermöglicht es Ihnen, Leistungsprobleme zu identifizieren.

GFS2 pflegt zwei Zähler:

- **dcount**, der die Anzahl angefragter DLM-Operationen zählt. Dies zeigt, wie viele Daten in die Berechnungen von Mittel- und Varianzwerten einfließen.
- **qcount**, der die Anzahl angefragter **syscall**-Operationen zählt. Im Allgemeinen ist **qcount** gleich oder größer als **dcount**.

Darüber hinaus pflegt GFS2 drei Mittel-/Varianzpaare. Die Mittel-/Varianzpaare sind geglättete, exponentielle Schätzungen; der verwendete Algorithmus ist jener, der zur Berechnung von Rundtrips im Netzwerkcode verwendet wird. Die Mittel-/Varianzpaare, die in GFS2 gepflegt werden, sind nicht skaliert, sondern in Einheiten von ganzzahligen Nanosekunden.

- srtt/srttvar: Geglättete Rundtrip-Zeit für nicht sperrende Operationen

- srttb/srttvarb: Geglättete Rundtrip-Zeit für sperrende Operationen
- irtt/irttvar: Zeit zwischen Anfragen (z. B. Zeit zwischen DLM-Anfragen)

Eine nicht sperrende Anfrage ist eine Anfrage, die sofort abgeschlossen wird, unabhängig davon, in welchem Status sich die fragliche DLM-Sperre befindet. Darunter fallen derzeit jegliche Anfragen, wenn entweder (a) der aktuelle Status der Sperre exklusiv ist oder (b) der angeforderte Status entweder Null oder nicht gesperrt ist oder (c) das „try“-Sperr-Flag gesetzt ist. Alle anderen Sperranfragen fallen unter sperrende Anfragen.

Größere Zeiten sind besser für IRTTs, kleinere Zeiten sind dagegen besser für RTTs.

Statistiken werden in zwei **sysfs**-Dateien gespeichert:

- Die **glstats**-Datei. Diese Datei ähnelt der **glocks**-Datei, mit dem Unterschied, dass es Statistiken enthält für ein Glock pro Zeile. Die Daten werden initialisiert von „pro CPU“ Daten für den Glock-Typ, für den das Glock erstellt wurde (ausgenommen Zähler, die auf Null gesetzt werden). Diese Datei kann sehr umfangreich sein.
- Die **lkstats**-Datei. Diese Datei enthält „pro CPU“ Statistiken für jeden Glock-Typ. Sie enthält eine Statistik pro Zeile, wobei jede Spalte einen CPU-Kern darstellt. Es gibt acht Zeilen pro Glock-Typ, wobei die Typen einander nachfolgen.

C.10. VERWEISE

Weitere Informationen über Tracepoints und die GFS2-**glocks**-Datei finden Sie in den folgenden Quellen:

- Dieser Anhang basiert in Auszügen auf einer Veröffentlichung von Steve Whitehouse, das beim Linux Symposium 2009 veröffentlicht wurde, erhältlich unter 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.
- Informationen über die internen Glock-Sperrregeln finden Sie unter 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.
- Informationen über Event Tracing finden Sie unter http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=blob;f=Documentation/trace/events.txt;h=09bd8e9029892e4e1d48078de4d076e24eff3dd2
- Informationen über das **trace-cmd**-Dienstprogramm finden Sie unter http://lwn.net/Articles/341902/.

ANHANG D. VERSIONSGESCHICHTE

Version 7.1-3.2 **Fri Jun 26 2015** **Hedda Peters**
Deutsche Übersetzung fertiggestellt

Version 7.1-3.1 **Fri Jun 26 2015** **Hedda Peters**
Übersetzungsdateien synchronisiert mit XML-Quellen 7.1-3

Version 7.1-3 **Tue Dec 16 2014** **Steven Levine**
Implementierung von sort_order auf der RHEL 6 Splash-Seite.

Version 7.0-9 **Wed Oct 8 2014** **Steven Levine**
Version für 6.6 GA-Release

Version 7.0-8 **Thu Aug 7 2014** **Steven Levine**
Version für 6.6 Beta-Release

Version 7.0-4 **Thu Jul 17 2014** **Steven Levine**
Behebt #1102591
Fügt eine Prozedur zur Konfiguration von GFS2 in einem Pacemaker-Cluster hinzu

Version 7.0-3 **Wed Jul 16 2014** **Steven Levine**
Behebt #1035119
Aktualisiert die Tabelle der Glock-Flags und fügt einen Abschnitt über Glock-Statistiken hinzu

Version 7.0-1 **Thu Jun 5 2014** **Steven Levine**
Erster Entwurf für 6.6 Release

Version 6.0-6 **Wed Nov 13 2013** **Steven Levine**
Version für 6.5 GA-Release

Version 6.0-5 **Fri Sep 27 2013** **Steven Levine**
Version für 6.5 Beta-Release

Version 6.0-3 **Fri Sep 27 2013** **Steven Levine**
Behebt #960841
Verdeutlicht die fehlende Unterstützung für SELinux mit GFS2-Dateisystemen.

Version 6.0-1 **Fri Sep 06 2013** **Steven Levine**
Hinweis über Samba und GFS2 hinzugefügt

Version 5.0-7 **Mon Feb 18 2013** **Steven Levine**
Version für 6.4 GA-Release

Version 5.0-5 **Mon Nov 26 2012** **Steven Levine**
Version für 6.4 Beta Release

Version 5.0-4 **Tue Nov 13 2012** **Steven Levine**
Behebt #860324
Aktualisiert Kapitel über GFS2-Konfiguration und operative Überlegungen mit kleinen Klarstellungen.

Behebt #807057
Fügt Hinweis auf Empfehlung hinzu, Rücksprache mit einem autorisierten Red Hat Vertreter zu halten, um Ihre Konfiguration vor dem Einsatz überprüfen.

Version 5.0-1 **Mon Oct 15 2012** **Steven Levine**

Kapitel über operative Überlegungen aktualisiert.

Version 4.0-2 **Thu Mar 28 2012** **Steven Levine**

Version für 6.3 GA-Release

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

Behebt: #782482, #663944

Fügt Kapitel über GFS2-Konfiguration und funktionale Überlegungen hinzu.

Behebt: #757742

Klärt Notwendigkeit für die Verwendung von GFS2 mit CLVM.

Behebt: #786621

Behebt kleinen Tippfehler.

Version 3.0-2 **Thu Dec 1 2011** **Steven Levine**

Release für GA von Red Hat Enterprise Linux 6.2

Version 3.0-1 **Mon Sep 19 2011** **Steven Levine**

Erste Revision für Red Hat Enterprise Linux 6.2 Beta Release

Behebt: #704179

Dokumentiert Unterstützung für den `tunegfs2`-Befehl.

Behebt: #712390

Fügt neues Kapitel über GFS2-Tracepoints hinzu.

Behebt: #705961

Korrigiert kleine Tippfehler.

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

Erste Release für Red Hat Enterprise Linux 6.1

Behebt: #549838

Dokumentiert die Unterstützung der standardmäßigen Linux-Kontingentfunktionen in Red Hat Enterprise Linux 6.1.

Behebt: #608750

Verdeutlicht die Beschreibung der GFS2-Rückzugsfunktion.

Behebt: #660364

Korrigiert Angaben zur maximalen GFS2-Dateisystemgröße.

Behebt: #687874

Fügt Kapitel zur Suche und Bereinigung von GFS2-Problemen hinzu.

Behebt: #664848

Fügt Informationen über das Auffinden von kontextabhängigen Pfaden vor der Konvertierung von GFS nach GFS2 hinzu.

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

Erste Release für Red Hat Enterprise Linux 6

STICHWORTVERZEICHNIS

A

B

D

E

F

G

H

K

L

M

O

P

Q

R

S

T

U

V

Z