Red Hat Enterprise Linux 6 GFS 2 (Global File System 2)

Red Hat GFS 2 (Red Hat Global File System 2)
엮음 7

Landmann

Red Hat Enterprise Linux 6 GFS 2 (Global File System 2)

Red Hat GFS 2 (Red Hat Global File System 2)
엮음 7

Landmann
rlandmann@redhat.com

초록

다음 부분에서는 Red Hat Enterprise Linux 6 용 Red Hat GFS2 (Red Hat Global File System 2)를 설정 및 관리하는 방법에 대해 설명합니다.

차례

소개

다음에서는 장애 복구형 스토리지 추가 기능에 있는 Red Hat GFS2 (Red Hat Global File System 2) 설정 및 관리 방법에 대해 설명합니다.

1. 대상 그룹

이 문서는 다음과 같은 작업에 능숙한 Linux 시스템 관리자를 위한 것입니다:

- Linux 시스템 관리 절차, 커널 설정 포함
- 파이버 채널 SAN과 같은 공유 저장 장치 네트워크 설치 및 설정

2. 관련 문서

Red Hat Enterprise Linux 사용에 관한 보다 자세한 내용은 다음의 문서 자료에서 참조하시기 바랍니다:

- *설치 가이드* — Red Hat Enterprise Linux 6 설치에 관한 내용을 다루고 있습니다.
- *활용 가이드* — Red Hat Enterprise Linux 6 활용, 설정, 관리에 관한 내용을 다루고 있습니다.
- *스토리지 관리 가이드* — Red Hat Enterprise Linux 6의 스토리지 장치 및 파일 시스템을 효과적으로 관리하는 방법에 관한 내용을 다루고 있습니다.

Red Hat Enterprise Linux 6의 고가용성 추가 기능 및 장애 복구형 스토리지 추가 기능에 대한 자세한 내용은 다음 자료에서 참조하십시오:

- *고가용성 추가 기능 개요* — Red Hat 고가용성 추가 기능에 대한 상세 개요를 다루고 있습니다.
- *클러스터 관리* — 고가용성 추가 기능 설치, 설정, 관리에 관한 내용을 다루고 있습니다.
- *LVM (Logical Volume Manager) 관리* — 클러스터 환경에서 LVM (Logical Volume Manager)을 실행하는 방법을 포함하여 LVM에 대한 설명을 다루고 있습니다.
- *DM Multipath* — Red Hat Enterprise Linux 의 Device-Mapper Multipath 기능 사용에 관한 내용을 다루고 있습니다.
- *로드 밸런서(Load Balancer) 관리* — 실제 서버 그룹에 걸쳐 IP 부하를 분산하기 위한 LVS (Linux Virtual Servers)를 제공하는 일련의 통합 소프트웨어 구성 요소인 로드 밸런서 (Load Balancer) 추가 기능을 사용하여 고성능 시스템 및 서비스 설정에 대한 내용을 다루고 있습니다.
- *릴리즈 노트* — Red Hat 제품의 최신 릴리즈에 관한 내용을 다루고 있습니다.

고가용성 추가 기능 문서 및 기타 다른 Red Hat 문서는HTML, PDF, RPM 버전으로 Red Hat Enterprise Linux문서 CD에서나 또는 http://www.redhat.com/docs/에서 온라인으로 확인하실 수 있습니다.

3. 피드백

오자를 발견하셨거나, 보다 좋은 메뉴얼을 만들기 위한 제안이 있다면, 언제든지 저희에게 알려 주십시오! **Red Hat Enterprise Linux 6** 및 **doc-Global_File_System_2**에 대한 리포트를 버그질라 (http://bugzilla.redhat.com/)에 제출해 주십시오. 버그 리포트를 제출하실 때 메뉴얼의 식별자**rh-gfs2(EN)-6 (2011-05-19T15:15)**를 알려주셔야 합니다.

문서 자료 개선을 위한 제안이 있으시면, 최대한 상세하고 명확히 설명해 주시기 바랍니다. 오류를 발견하셨다면, 저희가 쉽게 식별할 수 있도록 섹션 번호와 주위의 문장들을 함께 보내주시기 바랍니다.

4. 문서화 규정

이 메뉴얼에서는 특정 단어 및 구문을 강조 표시하여 특정 정보 부분에 주의를 집중시키기 위해 문서화 규정

을 사용하고 있습니다.

PDF 및 문서 편집에서 이 메뉴얼은 Liberation 글꼴 모음에 있는 서체를 사용합니다. 시스템에 Liberation 글꼴 모음이 설치되어 있을 경우 이는 HTML 편집에서도 사용되지만 설치되어 있지 않을 경우, 다른 동일한 서체로 나타나게 됩니다. 알림: Red Hat Enterprise Linux 5 및 이후 버전에는 기본값으로 Liberation 글꼴 모음이 들어 있습니다.

4.1. 표기 규정

네 가지 표기 규정을 사용하여 특정 단어 및 구문에 주의를 집중시킵니다. 이러한 규정 및 적용 방식은 다음과 같습니다.

고정폭 굵은체

쉘 명령, 파일 이름 및 경로를 포함한 시스템 입력을 강조하기 위해 사용됩니다. 키 및 키 조합을 강조하기 위해 사용되기도 합니다. 예:

> 현재 작업 중인 디렉토리에 있는 **my_next_bestselling_novel** 파일 내용을 확인하려면, 쉘 프롬프트에서 **cat my_next_bestselling_novel** 명령을 입력하고 **Enter** 키를 눌러 명령을 실행합니다.

위에서 파일 이름, 쉘 명령, 키 모두는 고정폭 굵은체로 나타나 있어 내용과 구별될 수 있습니다.

키 조합은 키 조합의 각 부분을 더하기(+) 기호로 연결하여 개별 키과 구별되게 할 수 있습니다. 예:

> **Enter** 키를 눌러 명령을 실행합니다.
>
> **Ctrl+Alt+F2**를 눌러 가상 터미널로 전환합니다.

첫 번째 예에서는 눌러야 하는 특정 키를 강조하고 있습니다. 두 번째 예에서는 키 조합 (동시에 눌러야 하는 세 개의 키 묶음)을 강조하고 있습니다.

소스 코드를 설명해야 할 경우, 문장에서 언급된 클래스 이름, 방식, 기능, 변수 이름 및 반환값은 위와 같이 **고정폭 굵은체**로 나타나게 됩니다. 예:

> 파일 관련 클래스에는 파일 시스템의 경우 **filesystem**, 파일의 경우 **file**, 디렉토리의 경우 **dir**가 포함됩니다. 각각의 클래스에는 자체의 권한 설정이 있습니다.

가변폭 굵은체

이는 프로그램 이름; 대화 상자 텍스트; 레이블된 버튼; 체크 박스 및 라디오 버튼 레이블; 메뉴 제목; 하부 메뉴 제목을 포함하여 시스템에 있는 단어 또는 구문을 나타냅니다. 예:

> 주 메뉴 바에서 **시스템 → 기본설정 → 마우스**를 선택하여 **마우스 기본 설정**을 시작합니다. **버튼** 탭에서, **왼손 잡이 마우스** 체크 상자를 선택하고 **닫기**를 클릭하여 주요 마우스 버튼을 왼쪽에서 오른쪽으로 전환합니다 (왼손 잡이일 경우 보다 적절하게 마우스 사용을 할 수 있게 함).
>
> **gedit** 파일에 특수 문자를 입력하려면 주 메뉴 바에서 **프로그램 → 보조 프로그램 → 문자 표**를 선택합니다. 그 다음으로, **문자 표** 메뉴 바에서 **찾기 → 찾기...**를 선택하고 **찾기** 필드에 문자를 입력하고 **다음**을 클릭합니다. 찾기한 문자가 **문자 표**에 강조 표시되어 나타납니다. 강조 표시된 문자를 두 번 클릭하여 **복사할 텍스트** 필드에 나타나면 **복사** 버튼을 클릭합니다. 편집 중인 문서로 돌아가 **gedit** 메뉴 바에서 **편집 → 붙여넣기**를 선택합니다.

위의 내용에는 프로그램 이름, 다양한 시스템 메뉴 이름 및 항목; 특정 프로그램 메뉴 이름; GUI 인터페이스에 있는 버튼 및 텍스트가 포함되어 있으며, 텍스트와 구별 가능하도록 모두 가변폭 굵은체로 되어 있습니다.

고정폭 굵은 이텔릭체 또는 ***가변폭 굵은 이텔릭체***

고정폭 굵은체이던 가변폭 굵은체이던지 간에 이텔릭체가 추가될 경우 이는 교체 또는 변경 가능한 텍스트를 나타내는 것입니다. 글자 그대로 입력하지 말아야 할 텍스트나 또는 상황에 따라 변경해야 하는 텍스트의 경우 이텔릭체로 나타냅니다. 예:

> ssh를 사용하여 원격 컴퓨터에 연결하려면, 쉘 프롬프트에 **ssh *username@domain.name***을 입력합니다. 원격 컴퓨터가 **example.com**이고 사용자 이름이 john일 경우, **ssh john@example.com**을 입력합니다.
>
> **mount -o remount *file-system*** 명령은 지정한 파일 시스템을 다시 마운트합니다. 예를 들어, **/home** 파일 시스템을 다시 마운트하려면 **mount -o remount /home** 명령을 사용합니다.
>
> 현재 설치된 패키지 버전을 보려면, **rpm -q *package*** 명령을 사용합니다. 그러면 다음과 같은 값이 출력됩니다: ***package-version-release***.

사용자 이름, 도메인 이름, 파일 시스템, 패키지, 버전 및 릴리즈는 굵은 이탤릭체로 표시되는 점에 유의하십시오. 각 단어는 자리 표시자로 명령을 실행할 때 입력하는 텍스트나 도는 시스템에 의해 나타나는 텍스트 중 하나입니다.

작업 제목을 표시하기 위한 기본적인 사용을 제외하고 중요한 새로운 용어를 처음 사용할 때 이텔릭체로 표시합니다. 예:

> Publican은 *DocBook* 발행 시스템입니다.

4.2. 인용문 규정

터미널 출력 결과 및 소스 코드 목록은 주위의 문장에서 잘 보이는 위치에 설정됩니다.

터미널로 보내진 출력 결과는**mono-spaced roman**에 설정되어 다음과 같이 나타납니다:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

소스 코드 목록도 **mono-spaced roman**에 설정되지만, 다음과 같이 구문 강조가 추가되어 있습니다:

```
static int kvm_vm_ioctl_deassign_device(struct kvm *kvm,
                  struct kvm_assigned_pci_dev *assigned_dev)
{
          int r = 0;
          struct kvm_assigned_dev_kernel *match;

          mutex_lock(&kvm->lock);

          match = kvm_find_assigned_dev(&kvm->arch.assigned_dev_head,
                                        assigned_dev->assigned_dev_id);
          if (!match) {
                  printk(KERN_INFO "%s: device hasn't been assigned before, "
                    "so cannot be deassigned\n", __func__);
                  r = -EINVAL;
                  goto out;
          }

          kvm_deassign_device(kvm, match);

          kvm_free_assigned_device(kvm, match);

out:
          mutex_unlock(&kvm->lock);
          return r;
}
```

4.3. 알림 및 경고

마지막으로, 3 종류의 시각적 스타일을 사용하여 간과될 수 있는 정보에 주의를 집중시킵니다.

참고

알림에서는 현재 작업에 대한 도움말, 지름길 또는 대안적 방법을 제공합니다. 알림 내용을 무시해도 상관없지만 효율적으로 작업할 수 있는 방법을 놓칠 수 있습니다.

중요

중요 상자에서는 현재 세션에만 적용되는 설정을 변경하거나 업데이트를 적용하기 전 다시 시작해야 하는 서비스와 같이 간과하기 쉬운 세부 사항을 제공합니다. 중요 상자를 무시해도 데이터를 손실하게 되지 않지만 문제를 일으킬 수 있습니다.

주의

경고는 무시해서는 안됩니다. 경고를 무시할 경우 대부분 데이터가 손실될 수 있습니다.

1장. GFS2 개요

Red Hat GFS2 파일 시스템은 장애 복구형 스토리지 추가 기능에 포함되어 있습니다. 이는 Linux 커널 파일 시스템 인터페이스 (VFS 레이어)와 직접 연결시키는 원시적 파일 시스템입니다. 클러스터 파일 시스템으로 구현될 경우 GFS2는 분산형 메타데이터 및 다중 저널을 사용할 수 있습니다. Red Hat은 고가용성 추가 기능의 구현으로 GFS2 파일 시스템 사용만을 지원합니다.

참고

GFS2 파일 시스템을 독립형 시스템과 클러스터 구성의 일부로 구현할 수 있으나 Red Hat Enterprise Linux 6 릴리즈에서 Red Hat은 단일 모드 파일 시스템으로 GFS2 사용을 지원하지 않습니다. Red Hat은 단일 노드에 최적화된 여러 고성능 단일 노드 파일 시스템을 지원하므로 일반적으로 클러스터 파일 시스템 보다 오버헤드가 감소됩니다. 파일 시스템을 마운트하기 위해 단일 노드만을 필요로 하는 경우 Red Hat은 GFS2 환경 설정에 이러한 파일 시스템을 사용할 것을 권장합니다. Red Hat은 클러스터 파일 시스템의 스냅샷을 마운트하기 위해 (예: 백업 목적) 단일 노드 GFS2 파일 시스템을 계속 지원합니다.

참고

Red Hat은 16 노드 이상을 운용하는 클러스터 파일 스템에 대한 GFS2 사용을 지원하지 않습니다.

GFS2는 64 비트 아키텍처를 기반으로 하며, 이는 이론적으로 8 EB 파일 시스템을 수용할 수 있습니다. 하지만, 64 비트 하드웨어의 경우 현재 지원되는 최대 GFS2 파일 시스템 크기는 100 TB입니다. 32 비트 하드웨어의 경우 현재 지원되는 최대 GFS2 파일 시스템 크기는 16 TB입니다. 보다 용량이 큰 GFS2 파일 시스템이 필요하실 경우, Red Hat 서비스 담당자에게 문의하십시오.

파일 시스템 크기를 결정할 때, 복구에 필요한 용량을 고려하셔야 합니다. 대용량 파일 시스템에서 **`fsck.gfs2`** 명령을 실행하면 시간이 오래 걸리고 메모리 용량을 많이 차지하게 됩니다. 또한 디스크 또는 디스크 서브 장애 발생 시 복구 시간은 백업 미디어 속도에 의해 제한됩니다. **`fsck.gfs2`** 명령에 필요한 메모리 양에 대한 내용은 3.11절. “파일 시스템 복구 ”에서 참조하십시오.

클러스터에 설정할 때, Red Hat GFS2 노드는 고가용성 추가 기능 설정 및 관리 도구로 설정 및 관리될 수 있습니다. 그 후 Red Hat GFS2는 GFS2 노드를 통해 파일 시스템 이름 공간을 지속적으로 확인하여 Red Hat 클러스터에 있는 GFS2 노드 사이에서 데이터를 공유하게 합니다. 이는 동일한 노드 상의 프로세스가 로컬 파일 시스템에 있는 파일을 공유하는 것과 동일한 방식으로 다른 노드 상의 프로세스가 GFS2 파일을 공유하게 합니다. 고가용성 추가 기능에 관한 자세한 내용은 *Red Hat Cluster 설정 및 관리* 장을 참조하시기 바랍니다.

GFS2 파일 시스템이 LVM 외부에서 사용될 수 있는 반면, Red Hat은 CLVM 논리 볼륨에 생성된 GFS2 파일 시스템만을 지원합니다. CLVM은 장애 복구형 스토리지 추가 기능에 들어 있습니다. 이는 클러스터 전반에 걸친 LVM 구현으로 클러스터에서 LVM 논리 볼륨을 관리하는 CLVM 데몬 **`clvmd`**에 의해 활성화됩니다. 데몬은 LVM2를 사용하여 클러스터를 통해 논리 볼륨을 관리하며, 클러스터에 있는 모든 노드가 논리 볼륨을 공유할 수 있게 합니다. LVM 볼륨 관리자에 관한 내용은 *LVM (Logical Volume Manager) 관리*에서 참조하시기 바랍니다.

`gfs2.ko` 커널 모듈은 GFS2 파일 시스템을 구현하여 GFS2 클러스터 노드로 불러 오는 커널 모듈입니다.

알림:

클러스터 파일 시스템으로 GFS2 파일 시스템을 구성할 때, 클러스터의 모든 노드가 공유 스토리지에 액세스할 수 있는지 확인해야 합니다. 공유 스토리지에 액세스할 수 있는 노드와 액세스할 수 없는 노드가 있는 비균형적 클러스터 구성은 지원되지 않습니다. 모든 노드에 GFS2 파일 시스템 자체를 마운트할 필요는 없습니다.

다음 부분에서는 GFS2의 이해를 돕기 위해 기본적이고, 간략한 내용을 소개합니다. 이는 다음과 같은 부분으로 구성되어 있습니다:

1.1. 새로운 기능 및 변경된 기능

다음 부분에서는 Red Hat Enterprise Linux 6의 초기 및 후속 릴리즈에 포함된 GFS2 문서 및 GFS2 파일 시스템의 새로운 기능과 변경된 기능에 대해 설명합니다.

1.1.1. Red Hat Enterprise Linux 6.0의 새로운 기능 및 변경된 기능

Red Hat Enterprise Linux 6.0에는 다음과 같은 문서 및 기능 업데이트 그리고 변경 사항이 포함되어 있습니다.

- Red Hat Enterprise Linux 6 버전의 경우 Red Hat은 단일 노드 파일 시스템으로 GFS2 사용을 지원하지 않습니다.
- Red Hat Enterprise Linux 6 릴리즈의 경우 GFS에서 GFS2 파일 시스템으로 업그레이드하기 위해 **gfs2_convert** 명령이 개선되었습니다. 이 명령에 대한 자세한 내용은 *부록 B. GFS에서 GFS2로 파일 시스템 변경* 에서 참조하십시오.
- Red Hat Enterprise Linux 6 릴리즈에서는 **discard**, **nodiscard**, **barrier**, **nobarrier**, **quota_quantum**, **statfs_quantum**, **statfs_percent** 마운트 옵션을 지원합니다. GFS2 파일 시스템을 마운트하는 내용은 3.2절. "파일 시스템 마운트"에서 참조하십시오.
- Red Hat Enterprise Linux 6 버전에는 새로운 섹션 1.4절. "GFS2 노드 잠금 기능"이 포함되어 있습니다. 이 부분에서는 GFS2 파일 시스템의 내부적 사항에 대해 설명합니다.

1.1.2. Red Hat Enterprise Linux 6.1의 새롭게 변경된 기능

Red Hat Enterprise Linux 6.1에는 다음과 같은 문서 및 기능 업데이트 그리고 변경 사항이 포함되어 있습니다.

- Red Hat Enterprise Linux 6.1 릴리즈에서 GFS2는 표준 Linux 쿼터 기능을 지원합니다. GFS2 쿼터 관리는 3.5절. "GFS2 쿼터 관리"에서 설명하고 있습니다.

  Red Hat Enterprise Linux 릴리즈의 경우 GFS2에서 쿼터를 관리하기 위해 **gfs2_quota** 명령을 사용해야 합니다. **gfs2_quota**에 대한 문서는 *부록 A. gfs2_quota 명령을 사용하여 GFS2 쿼터 관리* 에 있습니다.
- 이 문서에는 새로운 장 *4장. GFS2 파일 시스템과 관련된 문제 진단 및 수정* 이 추가되어 있습니다.
- 이 문서 전반에 걸쳐 기술적 내용을 수정 및 설명하고 있습니다.

1.2. GFS2 설정 전

GFS2 설치 및 설정 전, 다음과 같은 GFS2 파일 시스템의 주요 특징에 유의합니다:

GFS2 노드

클러스터에 있는 어떤 노드가 GFS2 파일 시스템을 마운트하게 할 지를 지정합니다.

파일 시스템 수

처음으로 생성할 GFS2 파일 시스템 수를 지정합니다. (나중에 파일 시스템을 더 추가할 수 있습니다.)

파일 시스템 이름

각각의 파일 시스템에 대해 고유한 이름을 지정합니다. 클러스터를 통한 모든 **lock_dlm** 파일 시스템에 대해 고유한 이름이어야 합니다. 각각의 파일 시스템이름은 매개 변수 형식으로 되어야 합니다. 예를 들어, 이 문서의 일부 예에서 **mydata1** 및 **mydata2**라는 파일 시스템 이름을 사용하고 있습니다.

저널

GFS2 파일 시스템에 대한 저널 수를 지정합니다. GFS2 파일 시스템을 마운트하는 각각의 노드에 대해 하나의 저널이 필요합니다. 나중에 추가 서버가 파일 시스템을 마운트함으로서 GFS2는 동적으로 저널을 추가할 수 있게 합니다. GFS2 파일 시스템에 저널을 추가하는 방법에 대한 내용은 3.7 절. “파일 시스템에 저널 추가 ”에서 참조하시기 바랍니다.

저장 장치 및 파티션

파일 시스템에 논리 볼륨 (CLVM을 통해)을 생성하기 위해 사용할 저장 장치 및 파티션을 지정합니다.

알림:

동시에 동일한 디렉토리에 있는 하나 이상의 노드에서 생성 및 삭제 작업 실행이 문제가 될 때 GFS2에서 실행 문제를 확인하실 수 있습니다. 이로 인해 시스템에서 실행 상의 문제가 발생할 경우, 노드에 의한 파일 생성 및 삭제를 노드에 지정된 디렉토리로 가능한 많이 로컬라이징해야 합니다.

1.3. GFS와 GFS2의 차이점

다음 부분에서는 GFS2가 GFS 이상으로 제공하는 개선 사항 및 변경 사항을 다루고 있습니다.

GFS에서 GFS2로 이전할 경우 GFS 파일 시스템을 **gfs2_convert** 유틸리티를 사용하여 GFS2로 변경해야 합니다. **gfs2_convert** 유틸리티에 대한 내용은 부록 B. *GFS에서 GFS2로 파일 시스템 변경* 에서 참조하시기 바랍니다.

1.3.1. GFS2 명령어

일반적으로, GFS2 기능은 GFS와 동일합니다. 다만, 파일 시스템 명령어를 GFS 대신 GFS2로 지정한 것입니다. 표 1.1. “GFS 및 GFS2 명령 ”에서는 동일한 GFS 및 GFS2 명령을 보여주고 있습니다.

표 1.1. GFS 및 GFS2 명령

GFS 명령	GFS2 명령	설명
mount	**mount**	파일 시스템을 마운트합니다. 시스템에 파일 시스템을 GFS 또는 GFS2 유형으로 할 지를 지정할 수 있습니다. GFS2 마운트 옵션에 대한 내용은 gfs2_mount(8) 맨 페이지를 참조하시기 바랍니다.
umount	**umount**	파일 시스템을 마운트 해제합니다.
fsck **gfs_fsck**	**fsck** **fsck.gfs2**	마운트 해제된 파일 시스템을 확인 및 복구합니다.
gfs_grow	**gfs2_grow**	마운트된 파일 시스템을 확장합니다.
gfs_jadd	**gfs2_jadd**	마운트된 파일 시스템에 저널을 추가합니다.
gfs_mkfs **mkfs -t gfs**	**mkfs.gfs2** **mkfs -t gfs2**	저장 장치에 파일 시스템을 생성합니다.
gfs_quota	**gfs2_quota**	마운트된 파일 시스템에 쿼터를 할당합니다.
gfs_tool	**gfs2_tool**	파일 시스템에 관한 정보를 설정, 조정, 확장합니다.
gfs_edit	**gfs2_edit**	파일 시스템 내부 구조를 보여주거나 출력 또는 편집합니다. **gfs2_edit** 명령은 GFS 파일 시스템 및 GFS2 파일 시스템에도 사용할 수 있습니다.
gfs_tool setflag jdata/inherit_jdata	**chattr +j** (선호하는 방법)	파일 또는 디렉토리에서 저널링을 활성화합니다.
setfacl/getfacl	**setfacl/getfacl**	파일 또는 디렉토리의 파일 액세스 제어 목록을 설정하거나 가져옵니다.
setfattr/getfattr	**setfattr/getfattr**	파일의 확장 속성을 설정하거나 가져옵니다.

GFS2 파일 시스템 명령에 대해 지원되는 옵션의 완전 목록은 해당 명령의 맨 페이지를 참조하시기 바랍니다.

1.3.2. GFS 및 GFS2의 기타 다른 차이점

다음 부분에서는 1.3.1절. “GFS2 명령어 ”에서 설명되지 않은 GFS 및 GFS2 관리에 있어서 기타 다른 차이점에 대해 간략하게 설명하고 있습니다.

문맥 의존적 (Context-Dependent) 경로 이름

GFS2 파일 시스템은 가변 목적 파일이나 디렉토리로 연결된 심볼릭 링크를 생성하게 하는 문맥 의존적 경로 이름을 지원하지 않습니다. GFS2에 있는 이러한 기능에 대해, **mount** 명령의 **bind** 옵션을 사용할 수 있습니다. GFS2에서의 문맥 의존적 경로 이름과 바인딩 마운트에 관한 자세한 내용은 3.12절. “바인드 마운트 및 문맥 의존적 경로 이름 ”에서 참조하시기 바랍니다.

gfs2.ko 모듈

GFS 파일 시스템을 구현하는 커널 모듈은 **gfs.ko**입니다. GFS2 파일 시스템을 구현하는 커널 모듈은 **gfs2.ko**입니다.

GFS2에서 쿼터 강제 활성화

GFS2 파일 시스템에서 쿼터 강제는 기본값으로 비활성화되어 있으므로 활성화시켜야 합니다. 쿼터 강제 활성화 및 비활성화에 대한 내용은 3.5절. “GFS2 쿼터 관리 ”에서 확인하십시오.

데이터 저널링

GFS2 파일 시스템은 **chattr** 명령의 사용을 지원하여 파일 또는 디렉토리에 있는 **j** 플래그를 설정 및 삭제합니다. 파일에 있는 **+j** 플래그 설정은 해당 파일에 있는 데이터 저널링을 활성화합니다. 디렉토리에 있는 **+j** 플래그 설정은 "inherit jdata"를 의미하며, 이는 결과적으로 디렉토리에 생성된 모든 파일 및 디렉토리가 저널링됨을 의미합니다. **chattr** 명령을 사용하는 것은 파일에서 데이터 저널링을 활성화 및 비활성화하기 위해 선호되는 방식입니다.

동적으로 저널 추가

GFS 파일 시스템에서 저널은 임베디드 메타데이터로 파일 시스템 외부에 존재하며 저널을 추가하기 전 파일 시스템이 있는 논리 볼륨 크기를 확장할 수 있게 합니다. GFS2 파일 시스템에서 저널은 (숨겨진) 평문 파일입니다. 이는 GFS2 파일 시스템의 경우, 추가저널에 대해 파일 시스템에 공간이 남아있는 한 추가 서버가 파일 시스템을 마운트함으로서 저널이 동적으로 추가될 수 있음을 의미합니다. GFS2 파일 시스템에 저널을 추가하는 방법에 관한 내용은 3.7절. “파일 시스템에 저널 추가 ”에서 참조하시기 바랍니다.

atime_quantum 매개 변수 삭제

GFS2 파일 시스템은 **atime_quantum** 조정 가능 매개 변수를 지원하지 않으며, 이는 얼마나 자주 **atime** 업데이트를 실행할 것인지를 지정하기 위해 GFS 파일 시스템에 의해 사용될 수 있습니다. 여기서 GFS2는 **relatime** 및 **noatime** 마운트 옵션을 지원합니다. GFS에서 **atime_quantum** 매개 변수를 설정하기 위해 **relatime** 마운트 옵션을 사용하는 것이 좋습니다.

마운트 명령의 data= 옵션

GFS2 파일 시스템을 마운트할 경우, **mount** 명령의 **data=ordered** 또는 **data=writeback** 옵션을 지정할 수 있습니다. **data=ordered**가 설정되면, 트랜젝션에 의해 수정된 사용자 데이터는 트랜젝션이 디스크로 커밋되기 이전에 디스크에서 삭제됩니다. 이는 크래시 후 파일에 있는 초기화되지 않은 블록을 사용자가 보지 못하게 합니다. **data=writeback**이 설정되면, 사용자 데이터는 언제든지 디스크에 작성될 수 있습니다. 이는 **ordered** 모드에서 처럼 동일한 일관성이 보장되지 않지만, 일부 작업 부하에 대해 좀 더 빠르게 처리될 수 있습니다. 기본값은 **ordered** 모드입니다.

gfs2_tool 명령

gfs2_tool 명령은 GFS에 대한 **gfs_tool** 명령이 지원하는 것 이외에 GFS2에 대한 다른 옵션 모음을 지원합니다.

- **gfs2_tool** 명령은 파일 시스템에 있는 저널 수를 포함하여 현재 설정된 저널에 관한 정보를 출력하는 **journals** 매개 변수를 지원합니다.
- **gfs2_tool** 명령은 GFS 통계를 보기위해 **gfs_tool** 명령이 사용하는 **counters** 플래그를 지원하지 않습니다.
- **gfs2_tool** 명령은 **inherit_jdata** 플래그를 지원하지 않습니다. 디렉토리를 "inherit jdata"로 플래그하기 위해, 디렉토리에 **jdata** 플래그를 설정하거나 **chattr** 명령을 사용하여 **+j** 플래그를 설정할 수 있습니다. 파일에 데이터 저널링을 활성화 및 비활성화하기 위해 **chattr** 명령을 사용하는 것이 선호되는 방식입니다.

gfs2_edit 명령

gfs2_edit 명령은 GFS에 대해 **gfs_edit** 명령이 지원하는 것 이외에 GFS2에 대해 다른 옵션 모음을 지

원합니다. 각각의 명령 버전이 지원하는 특정 옵션에 대한 자세한 내용은 **gfs2_edit** 및 **gfs_edit** man 페이지에서 참조하십시오.

1.3.3. GFS2 성능 개선

GFS2 파일 시스템의 여러 기능은 GFS 파일 시스템에서의 사용자 인터페이스와 다른점이 없지만 파일 시스템 성능을 향상시킵니다.

다음과 같은 방식으로 GFS2 파일 시스템은 파일 시스템 성능이 개선되었습니다:

- 단일 디렉토리에서 과도한 사용에 대해 보다 나은 성능 발휘
- 보다 빠른 동기식 I/O 실행
- 보다 빠른 캐시 읽기 (잠금 오버헤드 없음)
- 사전 할당된 파일로 보다 빠르게 직접 I/O (부여된 I/O 크기는 4M 블록과 같이 큼)
- 일반적으로 보다 빠른 I/O 실행
- 보다 빠른 **statfs** 호출로 인해 **df** 명령 실행이 더 빨라짐
- GFS와 비교할 때 **atime**에 의해 생성되는 여러 쓰기 I/O 동작을 줄이기 위해 **atime** 모드 개선

다음과 같은 방식에서 GFS2 파일 시스템은 보다 방대하고 주력적인 지원을 제공합니다.

- GFS2는 업스트림 커널 부분입니다 (2.6.19로 통합됨)
- GFS2는 다음과 같은 기능을 지원합니다:
  - SELinux 확장 속성
  - 표준 **ioctl**() 호출을 통한 **lsattr**() 및 **chattr**() 속성 설정
  - 나노단위초 타임스탬프

GFS2 파일 시스템은 파일 시스템의 내부적 효율성에 있어서 다음과 같은 사항이 개선되었습니다.

- GFS2는 보다 적은 커널 메모리 사용
- GFS2는 메타데이터 생성 번호가 필요하지 않음

  GFS2 메타데이터 할당에서는 읽기가 필요하지 않습니다. 잠금 해제 전 저널에서 블록을 삭제하여 다중 저널에 있는 메타데이터 블록 복사본을 관리합니다.
- GFS2에는 링크되지 않은 inode 또는 쿼터 변경에 관한 정보를 모르는 보다 간단한 로그 관리자가 포함되어 있습니다.
- **gfs2_grow** 및 **gfs2_jadd** 명령은 동시에 여러 인스턴스가 실행되지 않게 하기 위해 잠금 기능을 사용합니다.
- ACL 코드는 **creat**() 및 **mkdir**() 과 같은 호출로 단순화되었습니다.
- 링크되지 않은 inode, 쿼터 변경, **statfs** 변경 사항은 저널을 다시 마운트하지 않고 복구됩니다.

1.4. GFS2 노드 잠금 기능

GFS2 파일 시스템에서 최상의 성능을 얻기 위해서는 이의 동작에 대한 기본적인 이론을 이해하고 있는 것이 매우 중요합니다. 단일 노드 파일 시스템은 캐시와 함께 구현되며 요청되는 데이터를 자주 사용하는 경우 디스크 액세스의 대기 시간을 없애는 것을 목적으로 하고 있습니다. Linux에서 페이지 캐시 (및 버퍼 캐시)는 이러한 캐싱 기능을 제공합니다.

GFS2와 함께 각 노드에는 온디스크 데이터의 일부가 들어있는 자체적 페이지 캐시가 있습니다. GFS2는 *glocks* (gee-locks라고 발음함)라는 잠금 메커니즘을 사용하여 노드 사이의 캐시의 무결성을 관리합니다. glock 서브시스템은 기본 통신 계층으로 *DLM* (distributed lock manager)를 사용하여 구현되는 캐시 관리 기능을 제공합니다.

glocks는 inode 기반 캐시에 대해 보호되므로 inode 당 하나의 잠금 기능은 캐싱 계층을 제어하는데 사용됩니다. glock이 공유 모드 (DLM 잠금 모드: PR)에서 부여되면 glock 하의 데이터는 동시에 하나 이상의 노드에서 캐시될 수 있으므로 모든 노드가 데이터에 로컬로 액세스할 수 있게 됩니다.

glock이 전용 모드 (DLM 잠금 모드: EX)에서 부여되면 단일 노드만이 glock 하에 있는 데이터를 캐시할 수 있습니다. 이러한 모드는 데이터를 수정하는 모든 작업에 사용됩니다. (**`write`** 시스템 호출 등)

다른 노드가 즉시 부여될 수 없는 glock을 요청할 경우, DLM은 노드 또는 새로운 요청을 차단하는 glock을 현재 보유하고 있는 노드에 메세지를 보내 잠금을 드롭하도록 지시합니다. glock을 드롭하는 과정은 오랜 시간이 걸릴 수 있습니다 (대부분의 파일 시스템 작업 기준에 의해). 공유 glock을 드롭하는 경우 캐시된 데이터의 양에 비례하며 비교적 신속한 무효화된 캐시만을 필요로 합니다.

전용 glock을 드롭하려면 로그 플러시 후, 변경된 데이터를 디스크에 쓰는 작업이 필요하며, 이후에 공유 glock 별로 무효화 작업을 합니다.

단일 노드 파일 시스템의 캐시는 하나이지만 GFS2는 각 노드에 캐시가 따로 있는 것이 단일 노드 파일 시스템과 GFS2의 다른점입니다. 두 경우 모두 캐시된 데이터로의 액세스 대기 시간에 대한 크기의 정도가 비슷하지만 캐시되지 않은 데이터로의 액세스 대기 시간에 대해서는 다른 노드가 동일한 데이터를 캐시할 경우 GFS2 에서 훨씬 길어집니다.

참고

GFS2 캐싱 기능을 구현하는 방법은 다음과 같은 경우에 최상의 성능을 얻습니다:

- inode가 모든 노드에 걸쳐 읽기 전용으로 사용되는 경우
- inode가 단일 노드에서만 쓰기 또는 수정되는 경우

파일 작성 및 삭제 시 디렉토에 항목을 삽입하고나 제거하는 것은 디렉토리 inode에 쓰기로 계산되므로 주의하십시오.
규칙을 깨는 것은 극히 드물지만 이러한 규칙을 깨는 것이 가능합니다. 너무 자주 규칙을 무시할 경우 성능에 심각한 영향을 미칠 수 있습니다.
읽기/쓰기 맵핑으로 GFS2에 있는 파일을 **`mmap`**() 하고 있지만 읽기 전용으로할 경우, 이는 읽기로만 계산됩니다. GFS에서는 쓰기로 계산되기 때문에 **`mmap`**() I/O를 사용하는 경우 GFS2가 더 확장성이 높아집니다.
`noatime mount` 매개 변수를 설정하지 않은 경우, 파일의 타임 스탬프를 업데이트하기 위해 읽기도 기입됩니다. 모든 GFS2 사용자는 **`atime`**에 대해 특정 요구 사항이 없는 한 **`noatime`**으로 마운트하는 것이 좋습니다.

1.4.1. GFS2로 성능 조정

성능 향상을 위해 문제를 일으키는 응용 프로그램은 데이터를 저장하는 방법을 변경할 수 있습니다.

문제를 일으키는 응용 프로그램의 전형적인 예는 이메일 서버입니다. 이는 일반적으로 각 사용자에 대한 파일이 들어있는 스풀 디렉토리 (**`mbox`**) 또는 각 메세지에 대한 파일이 들어있는 각 사용자의 디렉토리 (**`maildir`**)로 구성되어 있습니다. IMAP를 통해 요청이 도착하면 특정 노드에 대한 동질 관계를 각 사용자에게 제공하는 이상적인 구성입니다. 이러한 방식에서 이메일 메세지 보기 및 삭제 요청은 하나의 노드에 있는 캐시에서 제공되는 경향이 있습니다. 노드에 장애가 발생했을 경우 다른 노드에서 세션을 다시 시작할 수 있습니다.

메일이 SMTP를 통해 도착한 경우에도 기본값으로 특정 사용자의 메일이 특정 노드에 전달되도록 노드를 개별적으로 설정할 수 있습니다. 기본 노드가 설정되어 있지 않은 경우, 메세지는 수신 노드에 의해 사용자의 메일 스풀에 직접 저장될 수 있습니다. 이는 일반적인 경우 하나의 노드에 캐시된 특정 파일 모음을 보관하기 위한 것으로 노드 장애 발생 시 직접 액세스를 허용하기 위함입니다.

이 설정은 GFS2의 페이지 캐시를 최대한 활용하여 **imap** 또는 **smtp** 응용 프로그램 중 하나에 장애를 투명하게 할 수 있습니다.

백업 또한 까다로운 부분 중 하나입니다. 가능하다면 특정 inode 집합을 캐싱하고 있는 노드에서 작업중인 각 노드 집합을 직접 백업하는 것이 좋습니다. 일정 시점에 실행하는 백업 스크립트가 있고 GFS2에서 실행되고 있는 응용 프로그램의 응답 시간의 증가와 일치할 경우, 페이지 캐시가 클러스터에 의해 효율적으로 사용되고 있지 않을 가능성이 높습니다.

백업을 수행하기 위해 응용 프로그램을 중지할 수 있는 경우에는 문제가 없지만, 백업이 하나의 노드에서만 실행되는 경우 백업이 완료되면 파일 시스템의 대부분이 노드에 캐시되며 다른 노드에서의 차후 액세스 성능이 저하됩니다. 이는 다음과 같은 명령을 사용하여 백업 완료 후 백업 노드에 VFS 페이지 캐시를 드롭하여 어느정도 완화될 수 있습니다.

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

하지만 각 노드에서 작동 중인 집합이 클러스터에 걸쳐 대부분 읽기 전용으로 공유되거나 단일 노드에서널리 사용되도록 항상 주의하는 것이 적합한 솔루션입니다.

1.4.2. GFS2 잠금 덤프를 사용하여 GFS2 성능 문제 해결

GFS2 캐싱의 비효율적 사용으로 인해 클러스터 성능에 영향을 받고 있는 경우 I/O 대기 시간이 길고 또한 늘어나고 있을 수 있습니다. GFS2의 잠금 덤프를 사용하여 문제의 원인을 확인할 수 있습니다.

GFS2 잠금 덤프 정보는 **debugfs** 파일에서 수집할 수 있으며 이는 다음과 같은 경로에 있습니다. 여기서 **debugfs**는 **/sys/kernel/debug/**에 마운트되어 있다고 가정합니다:

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

파일의 내용은 행 집합으로 구성되어 있습니다. G:로 시작하는 각 행은 하나의 glock을 나타내며 하나의 공백으로된 다음 행은 파일에서 바로 앞에 있는 glock과 관련된 정보의 항목을 나타냅니다.

debugfs 파일을 사용하는 데 있어서 가장 좋은 방법은 **cat** 명령을 사용하여 응용 프로그램에 문제가 발생하면 파일의 전체 내용의 복사본을 얻어 (대용량 RAM과 캐시된 inode가 많은 경우 시간이 걸릴 수 있음) 나중에 결과 데이터를 살펴보는 것입니다.

정보

debugfs 파일 복사본을 두 개 만들어 두는 것이 유용할 수 있습니다. 하나는 다른 사본 사용 후 몇 초 후 또는 몇 분 후에 사용합니다. 동일한 glock 번호와 관련된 두 개의 추적 홀더 정보를 비교하면, 작업 부하가 진행되고 있는지 (즉 단순한 속도 저하) 또는 걸려있는 상태 (이 경우 버그로 Red Hat 지원팀에 즉시 보고해야 함)인지 가려낼 수 있습니다.

H: (홀더)로 시작하는 **debugfs** 파일의 행은 부여되거나 부여 대기 중인 잠금 요청을 나타냅니다. 홀더 행 f: 에 있는 플래그 필드는 다음을 나타냅니다. 'W' 플래그는 대기 중인 요청을 나타내고, 'H' 플래그는 부여된 요청을 나타냅니다. 대기 중인 요청이 많은 glocks는 경합이 발생할 가능성이 높습니다.

표 1.2. “Glock 플래그 ”에서는 다른 glock 플래그의 의미를 보여줍니다. 표 1.3. “Glock 홀더 플래그 ”에서는 다른 glock 홀더 플래그의 의미를 보여줍니다.

표 1.2. Glock 플래그

플래그	이름	의미
d	강등 보류 (Pending demote)	보류된 (원격) 강등 요청
D	강등 (Demote)	강등 요청 (로컬 또는 원격)
f	로그 플러시 (Log flush)	glock을 해제하기 전 로그를 커밋해야 함
F	동결 (Frozen)	원격 노드에서 회신이 무시됨 - 복구 진행 중.
i	비활성화 진행 중 (Invalidate in progress)	glock 하에 있는 페이지 비활성화를 진행 중
I	초기화 (Initial)	DLM 잠금 기능이 glock과 관련된 경우에 설정됨
l	잠금 (Locked)	glock 상태 변경 중
p	강등 진행 중 (Demote in progress)	glock은 강등 요청에 응답하는 중
r	응답 보류 (Reply pending)	원격 노드에서 받은 응답을 처리 대기 중
y	더티 (Dirty)	glock을 해제하기 전에 데이터를 디스크에 플러시해야 함

표 1.3. Glock 홀더 플래그

플래그	이름	의미
a	비동기	glock 결과를 기다리지 않음 (결과를 나중에 poll함)
A	모두	호환되는 잠금 모드는 모두 사용 가능
c	캐시 없음	잠금을 해제하면 즉시 DLM 잠금으로 강등됨
e	만기 없음	차후의 잠금 취소 요청을 무시함
E	완전 일치	완전 일치하는 잠금 모드로 해야함
F	첫번째	홀더에 첫번째로 잠금이 부여된 경우 설정됨
H	홀더	요청된 잠금이 부여되었음을 나타냄
p	우선 순위	대기열의 선두에 있는 인큐 홀더
t	시도	"try" 잠금
T	1CB 시도	콜백을 보내는 "try" 잠금
W	대기	요청이 완료될 때까지 대기 중으로 설정

문제의 원인이 되고 있는 glock을 확인한 후, 관련된 inode를 찾습니다. glock 번호 (G: 행의 n:)에 나와 있습니다. ***type*/*number***의 형식이며, ***type***이 2이면, glock은 inode glock이고 ***number***는 inode 번호가 됩니다. inode를 추적하려면, **`find -inum` *`number`***를 실행합니다. 여기서 ***number***는 glocks 파일에서 16 진수에서 10 진수로 변환하는 inode 번호입니다.

경고

잠금 경합이 발생할 경우, 파일 시스템에서 **`find`**를 실행하면 상황이 악화될 가능성이 높습니다. 경합하는 inode를 찾을 경우 **`find`**를 실행하기 전 응용 프로그램을 중지하는 것이 좋습니다.

표 1.4. “Glock 유형 ”에서는 다른 glock 유형의 의미를 보여줍니다.

표 1.4. Glock 유형

유형 번호	잠금 유형	사용
1	Trans	트랜젝션 잠금
2	Inode	Inode 메타데이터 및 데이터
3	Rgrp	리소스 그룹 메타데이터
4	Meta	슈퍼 블록
5	Iopen	가장 가까운 마지막 Inode 검색
6	Flock	**`flock`**(2) syscall
8	Quota	Quota 작업
9	Journal	저널 뮤텍스

확인된 glock이 다른 유형이면, 이는 유형 3: (리소스 그룹)일 것입니다. 일반 부하에서 다른 glock 유형을 기다리고 있는 여러 프로세스가 있을 경우 Red Hat 지원팀에 알려주십시오.

리소스 그룹 잠금 대기열에 여러 대기 요청이 나타나는 경우에는 몇 가지 이유가 있습니다. 그 중 하나로 파일 시스템의 리소스 그룹 수에 비해 노드 수가 많은 경우가 그러하며 파일 시스템이 꽉 찬 경우도 그러합니다 (평균적으로 길게 여유 블록 검색 시간을 요청). 두 경우 모두 스토리지를 추가하고 **`gfs2_grow`** 명령을 사용하여 파일 시스템을 확장하여 향상될 수 있습니다.

2장. 시작하기

다음 부분에서는 GFS2 초기 설정 절차에 대해 설명하며 다음과 같은 부분으로 구성되어 있습니다:

2.1. 선수 작업

Red Hat GFS2를 설정하기 전, GFS2 노드의 주요 특징을 확인합니다 (1.2절. “GFS2 설정 전 ” 참조). 또한, GFS2 노드에 있는 시계가 동기화되어 있는지를 확인합니다. Red Hat Enterprise Linux 배포판과 함께 제공된 NTP (Network Time Protocol) 소프트웨어를 사용할 것을 권장합니다.

알림:

GFS2 노드에 있는 시스템 시계는 불필요한 inode 타임스태프를 업데이트하지 못하게 하기 위해 몇 분 이내로 되어 있어야 합니다. 불필요한 inode 타임스탬프 업데이트로 클러스터 실행에 심각한 영향을 미칠 수 있습니다.

2.2. 초기 설정 작업

초기 GFS2 설정은 다음과 같은 작업으로 구성되어 있습니다:

1. 논리 볼륨 설정
2. GFS2 파일 시스템 작성
3. 파일 시스템 마운트

GFS2 설정을 초기화하기 위해 다음과 같은 단계를 실행합니다.

1. LVM을 사용하여, 각각의 Red Hat GFS2 파일 시스템에 해당하는 논리 볼륨을 생성합니다.

   

   알림:

   Red Hat Cluster Suite에 포함된 **`init.d`** 스크립트를 사용하여 논리 볼륨을 자동으로 활성화 및 비활성화할 수 있습니다. **`init.d`** 스크립트에 관한 보다 자세한 내용은 *Red Hat Cluster 설정 및 관리* 부분을 참조하시기 바랍니다.

2. 1 단계에서 생성한 논리 볼륨에 GFS2 파일 시스템을 생성합니다. 각각의 파일 시스템에 대해 고유한 이름을 선택합니다. GFS2 파일 시스템 생성에 관한 보다 자세한 내용은 3.1절. “파일 시스템 작성 ”에서 참조하시기 바랍니다.

   클러스터 GFS2 파일 시스템을 생성하기 위해 다음과 같은 포맷을 사용하실 수 있습니다:

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

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

   GFS2 파일 시스템을 생성하는 방법에 관한 보다 자세한 내용은 3.1절. “파일 시스템 작성 ”에서 참조하시기 바랍니다.

3. 각각의 노드에서 GFS2 파일 시스템을 마운트합니다. GFS2 파일 시스템 마운트 방법에 관한 보다 자세한 내용은 3.2절. “파일 시스템 마운트”에서 참조하시기 바랍니다.

   명령어 사용법:

   mount *BlockDevice MountPoint*

   mount -o acl *BlockDevice MountPoint*

   -o acl mount 옵션으로 ACL 파일을 조작할 수 있습니다. **-o acl** 마운트 옵션 없이 파일 시스템이 마운트되어 있을 경우, 사용자는 (**getfacl** 명령을 사용하여) ACL 파일을 볼 수 있지만 (**setfacl** 명령을 사용하여) 이를 설정할 수 없습니다.

알림:

Red Hat 고가용성 추가 기능에 포함되어 있는 **init.d** 스크립트를 사용하여 GFS2 파일 시스템 마운트 및 마운트 해제를 자동화할 수 있습니다.

2.3. GFS2 클러스터 배포

클러스터 파일 시스템의 배포는 단일 노드 배포에 대한 "일시적인" 대체가 아닙니다. 시스템을 테스트하고 필요한 성능 수준으로 작동하는지 확인하기 위해 새로 설치 시 8-12 주 정도의 시험 기간을 갖을 것을 권장합니다. 이 기간 동안 성능과 기능에 관한 문제를 해결하고 문의 사항이 있을 경우 Red Hat 지원팀에 문의해야 합니다. 또한 클러스터의 배포를 고려하고 있는 경우 향후 지원 관련 문제가 발생하지 않도록 배포 전 Red Hat 지원팀에게 클러스터 구성에 대한 평가를 받으실 것을 권장합니다.

3장. GFS2 관리

다음 부분에서는 GFS2 관리를 위한 작업 및 명령에 대해 다루고 있으며 다음과 같은 부분으로 구성되어 있습니다:

3.1. 파일 시스템 작성

mkfs.gfs2 명령을 사용하여 GFS2 파일 시스템을 생성합니다. **mkfs**를 **-t gfs2** 옵션과 함께 사용할 수도 있습니다. 파일 시스템은 활성화된 LVM 볼륨에 생성됩니다. 다음은 **mkfs.gfs2** 명령을 실행하기 위해 필요한 내용입니다:

- 잠금 프로토콜/모듈 이름 (클러스터에 해당하는 잠금 프로토콜은 **lock_dlm** 임)
- 클러스터 이름 (클러스터 설정 부분으로 실행할 경우)
- 저널 수 (파일 시스템을 마운트할 수 있는 각각의 노드에 필요한 하나의 저널)

GFS 파일 시스템을 생성할 때, **mkfs.gfs2** 명령을 직접 사용하거나 또는 **mkfs** 명령을 ***-t*** 매개 변수와 함께 **gfs2** 유형의 파일 시스템을 지정한 다음 gfs2 파일 시스템 옵션을 사용할 수 있습니다.

알림:

mkfs.gfs2 명령으로 GFS2 파일 시스템을 생성하면 파일 시스템 크기를 줄일 수 없습니다. 하지만 3.6절. “파일 시스템 확장하기 ”에 설명하고 있듯이 **gfs2_grow** 명령을 사용하여 기존 파일 시스템 크기를 늘릴 수는 있습니다.

사용법

클러스터된 GFS2 파일 시스템을 생성할 때, 다음의 포맷 중 하나를 사용하실 수 있습니다:

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

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

로컬 GFS2 파일 시스템을 생성할 때, 다음의 포맷 중 하나를 사용하실 수 있습니다:

참고

Red Hat Enterprise Linux 6 버전의 경우 Red Hat은 단일 노드 파일 시스템으로 GFS2 사용을 지원하지 않습니다.

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

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

경고

LockProtoName 및 ***LockTableName*** 매개 변수 사용에 익숙한 지를 확인합니다. ***LockProtoName*** 및 ***LockTableName*** 매개 변수의 부적절한 사용으로 파일 시스템이나 잠금 공간 손실의 원인이 될 수 있습니다.

LockProtoName

사용할 잠금 프로토콜 이름을 지정합니다. 클러스터 용 잠금 프로토콜은 **lock_dlm**입니다.

LockTableName

이러한 매개 변수는 클러스터 설정에서의 GFS2 파일 시스템 용으로 지정되어 있습니다. 이는 다음과 같이 (띄어쓰기 없이) 콜론을 사용하여 두 부분으로 나뉘어 집니다: ***ClusterName:FSName***

- ***ClusterName***, 생성되고 있는 GFS2 파일 시스템에 대한 클러스터의 이름입니다.
- ***FSName*** 파일 시스템 이름은 1에서 16자 길이로 될 수 있으며 클러스터에 있는 모든 **lock_dlm** 파일 시스템 및 각각의 로컬 노드에 있는 모든 파일 시스템 (**lock_dlm** 및 **lock_nolock**)에 대해 고유한 이름이어야 합니다.

Number

mkfs.gfs2 명령에 의해 생성된 저널 수를 지정합니다. 파일 시스템을 마운트하는 각각의 노드에 대해 하나의 저널이 필요합니다. 3.7절. “파일 시스템에 저널 추가 ”에서 설명하고 있듯이 GFS2 파일 시스템의 경우, 파일 시스템을 확장하지 않고 나중에 저널을 추가할 수 있습니다.

BlockDevice

논리 또는 물리 볼륨을 지정합니다.

예시

예에서, **lock_dlm**이 클러스터 파일 시스템이 된 이후 이는 파일 시스템이 사용하는 잠금 프로토콜이 됩니다. 클러스터 이름은 **alpha**이며, 파일 시스템 이름은 **mydata1**입니다. 파일 시스템에는 8 개의 저널이 포함되어 있으며 **/dev/vg01/lvol0**에 생성됩니다.

```
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
```

예에서, 클러스터 **alpha**에서 사용될 수 있는 두 번째 **lock_dlm** 파일 시스템이 생성되어 있습니다. 파일 시스템 이름은 **mydata2**입니다. 파일 시스템에는 8 개의 저널이 포함되어 있으며 **/dev/vg01/lvol1**에 생성됩니다.

```
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
```

전체 옵션

표 3.1. “명령 옵션: **mkfs.gfs2** ”에서는 **mkfs.gfs2** 명령 옵션 (플래그 및 매개 변수)을 설명합니다.

표 3.1. 명령 옵션: mkfs.gfs2

플래그	매개 변수	설명
-c	***Megabytes***	***Megabytes***에 각각의 저널 쿼터 변경 파일의 처음 크기를 설정합니다.
-D		디버깅 출력 결과를 활성화합니다.
-h		도움말. 사용 가능한 옵션을 보여줍니다.
-J	***MegaBytes***	저널 크기를 메가바이트 단위로 지정합니다. 기본값 저널 크기는 128 메가 바이트입니다. 최소 크기는 8 메가 바이트입니다. 보다 크기가 큰 저널은 크기가 작은 저널보다 더 많은 메모리를 사용하여도 성능이 향상됩니다.
-j	***Number***	**mkfs.gfs2** 명령에 의해 생성된 저널 수를 지정합니다. 파일 시스템을 마운트하는 각각의 노드에 대해 하나의 저널이 필요합니다. 이 옵션이 지정되지 않았을 경우, 하나의 저널이 생성됩니다. GFS2 파일 시스템의 경우, 파일 시스템을 확장하지 않고 나중에 저널을 추가할 수 있습니다.
-O		파일 시스템을 작성하기 전 **mkfs.gfs2** 명령은 확인 질문을 하지 않게 됩니다.
-p	***LockProtoName***	사용할 잠금 프로토콜 수를 지정합니다. 잠금 프로토콜에는 다음과 같은 것이 포함됩니다: **lock_dlm** — 클러스터 파일 시스템에 필요한 표준 잠금 모듈입니다. **lock_nolock** — GFS2가 로컬 파일 시스템처럼 작동할 경우 사용됩니다 (하나의 노드에서만).
-q		정숙 모드. 아무것도 보여주지 않습니다.
-r	***MegaBytes***	리소스 그룹 크기를 메가 바이트 단위로 지정합니다. 최소 리소스 그룹 크기는 32 MB입니다. 최대 리소스 그룹 크기는 2048 MB입니다. 크기가 큰 리소스 그룹은 대용량 파일 시스템에서 성능이 향상될 수 있습니다. 크기가 지정되어 있지 않을 경우, mkfs.gfs2는 파일 시스템 크기에 기반하여 리소스 그룹 크기를 선택합니다: 평균 파일 시스템 크기는 256 MB 리소스 그룹을 갖게 되며, 용량이 큰 파일 시스템은 성능 향상을 위해 크기가 큰 리소스 그룹을 갖습니다.
-t	***LockTableName***	**lock_dlm** 프로토콜을 사용할 경우 잠금 테이블 영역에 지정된 유일한 식별자; **lock_nolock** 프로토콜은 이러한 매개 변수를 사용하지 않습니다. 다음과 같이 이러한 매개 변수는 콜론 (빈 칸 없이)으로 두 부분으로 나뉘어 집니다: ***ClusterName:FSName***. ***ClusterName***은 GFS2 파일 시스템이 생성되어 있는 Red Hat 클러스터 이름입니다; 클러스터의 멤버에게만 이 파일 시스템의 사용 권한이 주어집니다. 클러스터 이름은 **클러스터 설정 도구**를 통해 **/etc/cluster/cluster.conf** 파일에 설정되며 Red Hat Cluster Suite 클러스터 관리 GUI에 있는 **클러스터 상**

		태 도구에 나타납니다. ***FSName***, 파일 시스템 이름으로, 1에서 16자 길이로 될 수 있으며, 클러스터에 있는 모든 파일 시스템 중에서 고유한 이름이어야 합니다.
-u	***MegaBytes***	각 저널의 링크되지 않은 태그 파일의 처음 크기를 지정합니다.
-V		명령 버전 정보를 보여줍니다.

3.2. 파일 시스템 마운트

GFS2 파일 시스템을 마운트하기 전, 파일 시스템이 있어야 하며 (3.1절. "파일 시스템 작성 " 참조), 파일 시스템이 위치해 있는 볼륨을 활성화하여 클러스터링 지원 및 잠금 시스템을 시작해야 합니다 (*Red Hat Cluster 설정 및 관리* 참조). 이러한 요구 사항이 설정되어 있어야, Linux 파일 시스템에서와 같이 GFS2 파일 시스템을 마운트할 수 있습니다.

참고

클러스터 관리자 (**cman**)가 시작되지 않을 때 GFS2 파일 시스템을 마운트하려고 하면 다음과 같은 오류 메세지가 표시됩니다:

```
[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
```

ACL 파일을 조작하려면, **-o acl** 마운트 옵션을 사용하여 파일 시스템을 마운트해야 합니다. **-o acl** 마운트 옵션을 사용하지 않고 파일 시스템이 마운트될 경우, 사용자는 ACL (**getfacl** 사용)을 볼 수 있으나 이를 설정할 수 는 (**setfacl** 사용) 없게 됩니다.

사용법

ACL을 조작하지 않고 마운트하기

```
mount BlockDevice MountPoint
```

ACL을 조작하여 마운트하기

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

-o acl

ACL 파일을 조작 허용하기 위한 GFS2 특정 옵션입니다.

BlockDevice

GFS2 파일 시스템이 위치할 블록 장치를 지정합니다.

MountPoint

GFS2 파일 시스템을 마운트할 디렉토리를 지정합니다.

예시

예에서, **/dev/vg01/lvol0**에 있는 GFS2 파일 시스템은 **/mygfs2** 디렉토리에 마운트되어 있습니다.

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

전체 사용법

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

-o option 매개 변수는 GFS2 특정 옵션이나 (표 3.2. “GFS2-특정 마운트 옵션 ” 참조) 또는 표준 Linux **mount -o** 옵션, 또는 이 두가지 옵션의 조합으로 구성되어 있습니다. 다중 ***option*** 매개 변수는 빈 칸없이 콤마로 구분됩니다.

알림:

mount 명령은 Linux 시스템 명령입니다. 다음에서 설명하는 GFS2 특정 옵션을 사용하는 것에 더하여, 기타 다른 표준 **mount** 명령 옵션을 사용할 수 있습니다 (예, **-r**). 기타 다른 Linux **mount** 명령 옵션에 대한 자세한 내용은 Linux **mount** 맨 페이지를 참조하시기 바랍니다.

표 3.2. “GFS2-특정 마운트 옵션 ”에서는 마운트 시 GFS2를 통과할 수 있는 사용 가능한 GFS2 특정 **-o option** 값을 설명합니다.

참고

다음 표에서는 로컬 파일 시스템에서만 사용할 수 있는 옵션을 설명하고 있습니다. 하지만 Red Hat Enterprise Linux 6 릴리즈의 경우, Red Hat은 단일 노드 파일 시스템으로 GFS2 사용을 지원하지 않음에 유의합니다. Red Hat은 클러스터 파일 시스템의 스냅샷을 마운트하기 위해 단일 노드 GFS2 파일 시스템을 지속적으로 지원합니다 (예: 백업 목적 등).

표 3.2. GFS2-특정 마운트 옵션

옵션	설명
acl	ACL 파일을 조작 허용합니다. 파일 시스템이 **acl** 마운트 옵션 없이 마운트되어 있을 경우, 사용자는 ACL을 볼 수 있지만 (**getfacl** 사용), 이를 설정할 수 없습니다 (**setfacl** 사용).
data=[ordered|writeback]	**data=ordered**가 설정되면, 트랜젝션에 의해 수정된 사용자 데이터는 트랜젝션이 디스크로 커밋되기 이전에 디스크에서 삭제됩니다. 이는 크래시 후 파일에 있는 초기화되지 않은 블록을 사용자가 보지 못하게 합니다. **data=writeback** 모드가 설정되면, 사용자 데이터는 언제든지 디스크에 작성될 수 있습니다; 이는 **ordered** 모드에서 처럼 동일한 일관성이 보장되지 않지만, 일부 작업 부하에 대해 좀 더 빠르게 처리될 수 있습니다. 기본값은 **ordered** 모드입니다.
ignore_local_fs **주의 :** GFS2 파일 시스템이 공유될 경우 이 옵션을 사용해서는 *안됩니다*.	파일 시스템을 멀티 호스트 파일 시스템 처럼 다루기 위해 GFS2를 강제합니다. 기본값으로, **lock_nolock**을 사용하면 **localflocks** 플래그가 자동으로 활성화됩니다.
localflocks **주의 :** GFS2 파일 시스템이 공유될 경우 이 옵션을 사용해서는 안됩니다.	GFS2에게 VFS (virtual file system) 레이어가 모든 flock 및 fcntl 작업을 하도록 지시합니다. **localflocks** 플래그는 **lock_nolock**에 의해 자동으로 활성화됩니다.
lockproto=*LockModuleName*	사용자가 파일 시스템과 함께 사용할 잠금 프로토콜을 지정할 수 있게 합니다. ***LockModuleName***이 지정되어 있지 않을 경우, 잠금 프로토콜 이름은 모든 파일 시스템 슈퍼 블록에서 읽어오게 됩니다.
locktable=*LockTableName*	사용자가 파일 시스템과 함께 사용할 잠금 테이블을 지정할 수 있게 합니다.
quota=[off/account/on]	파일 시스템에 해당하는 쿼터를 활성화 또는 비활성화합니다. **account** 상태에서 쿼터를 설정하면 UID/GID 사용 통계가 파일 시스템에 의해 올바르게 관리됩니다; 한계 및 경고 값은 무시됩니다. 기본값은 **off**입니다.
errors=panic|withdraw	**errors=panic**을 지정하면 파일 시스템 오류로 인해 커널 패닉이 발생하게 됩니다. **errors=withdraw**를 지정하는 것과 동일한 기본값 동작은 파일 시스템에서 시스템을 철회하는 것으로 다음 부팅 시 까지 액세스하지 못하게 합니다. 일부 경우 시스템은 계속 가동할 수 있습니다. GFS2 철회 기능에 대한 내용은 3.14절. “GFS2 철회 (Withdraw) 기능 ”에서 참조하십시오.
discard/nodiscard	GFS2는 여유 블록의 "삭제 " I/O 요청을 생성시키게 합니다. 이는 적절한 하드웨어가 씬 프로비저닝과 유사한 체계를 구현하는데 사용될 수 있습니다.
barrier/nobarrier	저널을 삭제할 때 GFS2가 I/O 장벽을 전송하게 합니다. 기본값은 **on**입니다. 이 옵션은 기본 장치가 I/O 장벽을 지원하지 않을 경우 자동으로 **off**됩니다. 블록 장치가 고안되어 쓰기 캐시 내용을 손실하지 않도록 설계된 경우를 제외

	하고 (예: UPS에 있거나 쓰기 캐시가없는 경우) GFS2와 함께 I/O 장벽을 사용할 것을 적극 권장합니다.
quota_quantum=*secs*	쿼터 정보 변경이 쿼터 파일에 기록되기 전 노드에 머무를 수 있는 시간 (초)을 설정합니다. 이는 이러한 매개 변수를 설정하는데 선호되는 방법입니다. 값은 0보다 큰 정수입니다. 기본값은 60초 입니다. 이보다 짧게 설정하면 지연 쿼터 정보의 업데이트가 빨라지며 쿼터를 초과할 가능성이 적어 집니다. 길게 설정하면 커터에 관련된 파일 시스템의 동작 속도가 빨라지고 효율성이 향상됩니다.
statfs_quantum=*secs*	**statfs**의 느린 버전을 설정하기 위해 **statfs_quantum**을 0으로 설정하는 것이 바람직한 방법입니다. 기본값은 30초로 **statfs** 변경이 마스터 **statfs** 파일에 동기화되기 전 최대 시간을 설정합니다. 이는 보다 빠르지만 덜 정확한 **statfs** 값이나 또는 느리지만 보다 정확한 값으로 조정할 수 있습니다. 이 옵션을 0으로 설정하면 **statfs**는 항상 true 값을 보고하게 됩니다.
statfs_percent=*value*	기간이 만료되지 않은 경우에도, 마스터 **statfs** 파일로 다시 동기화되기 전 로컬 기반 **statfs** 정보의 최대 변경 비율에 대한 한도를 부여합니다. **statfs_quantum**이 0으로 설정되어 있을 경우, 이 설정은 무시됩니다.

3.3. 파일 시스템 마운트 해제

GFS2 파일 시스템은 Linux 파일 시스템에서와 동일한 방식으로 — **umount** 명령을 사용하여 마운트 해제될 수 있습니다.

알림:

umount 명령은 Linux 시스템 명령입니다. 이 명령에 대한 내용은 Linux **umount** 명령 맨 페이지에서 확인하실 수 있습니다.

사용법

```
umount MountPoint
```

MountPoint

GFS2 파일 시스템이 마운트된 디렉토리를 지정합니다.

3.4. GFS2 파일 시스템을 마운트하는 경우 주의 사항

시스템 종료 시 파일 시스템이 마운트 해제될 경우 **fstab** 파일에 있는 항목을 통해 자동으로 마운트하는 대신 수동으로 마운트된 GFS2 파일 시스템은 시스템에 인식되지 않습니다. 결과적으로 GFS2 스크립트는 GFS2 파일 시스템을 마운트 해제하지 못하게 됩니다. GFS2 종료 스크립트가 실행된 후, 표준 종료 프로세스는 클러터스터 인프라 등 남아있는 사용자 프로세스를 종료시키고 파일 시스템을 마운트 해제하려 합니다. 이러한 마운트 해제는 클러스터 인프라 없이 실패하게 되어 시스템은 정지하게 됩니다.

GFS2 파일 시스템이 마운트 해제되었을 경우 시스템이 중지되지 않도록 하려면 다음 중 하나를 수행하십시오:

- **fstab** 파일에 있는 항목을 사용하여 GFS2 파일 시스템을 마운트합니다.
- GFS2 파일 시스템이 **mount** 명령을 사용하여 수동으로 마운트되었을 경우, 시스템을 종료하거나 다시 시작하기 전 **umount** 명령을 사용하여 파일 시스템을 수동으로 마운트 해제해야 합니다.

시스템 종료 중 파일 시스템을 마운트 해제시 파일 시스템이 중지될 경우 하드웨어를 다시 시작해 주십시오. 종료 과정에서 파일 시스템은 이전에 동기화되므로 데이터를 손실할 가능성은 없습니다.

3.5. GFS2 쿼터 관리

파일 시스템 쿼터는 사용자 또는 그룹이 사용할 수 있는 파일 시스템 공간을 제한하기 위해 사용됩니다. 사용자 또는 그룹은 쿼터가 설정될 때 까지 쿼터 제한을 갖지 않습니다. GFS2 파일 시스템이 **quota=on** 또는 **quota=account** 옵션으로 마운트될 때 공간 제한이 없을 경우에도 GFS2는 각각의 사용자 및 그룹에 의해 사용되는 공간을 추적합니다. GFS2는 트랜잭션 방식으로 쿼터 정보를 업데이트하므로 시스템 크래시 경우 재구성할 쿼터 사용량이 필요하지 않습니다.

실행 속도가 감소되지 않게 하기 위해, GFS2 노드는 주기적으로 쿼터 파일 업데이트를 동기화합니다. "fuzzy" 쿼터 계산은 사용자 또는 그룹이 설정 한계를 약간 초과하는 것을 허용합니다. 이를 최소화하려면, GFS2는 "hard" 쿼터 제한을 사용하여 동기화 주기를 동적으로 감소시킵니다.

참고

Red Hat Enterprise Linux 6.1 릴리즈에서 GFS2는 표준 Linux 쿼터 기능을 지원합니다. 이 기능을 사용하려면 **quota** RPM을 설치해야 합니다. 이는 GFS2에서 권장되는 쿼터 관리 방법이며 쿼터를 사용하는 모든 GFS2 배포에서 사용해야 합니다. 다음 부분에서는 이러한 기능을 사용하여 GFS2 쿼터 관리에 대해 설명합니다.
이전 Red Hat Enterprise Linux 릴리즈의 경우 GFS2는 쿼터를 관리하기 위해 **gfs2_quota** 명령이 필요했습니다. **gfs2_quota** 명령 사용에 대한 자세한 내용은 *부록 A. gfs2_quota 명령을 사용하여 GFS2 쿼터 관리* 에서 참조하십시오.

3.5.1. 디스크 쿼터 설정

디스크 쿼터를 구현하려면, 다음 단계를 사용합니다:

1. 강제 또는 사용 계산 모드에서 쿼터를 설정합니다.
2. 현재 블록 사용량 정보가 들어있는 쿼터 데이터베이스 파일을 초기화합니다.
3. 쿼터 정책을 할당합니다. (사용 계산 모드에서 이러한 정책은 강제되지 않습니다.)

이러한 각 단계는 다음 부분에서 자세히 설명하고 있습니다.

3.5.1.1. 강제 또는 사용 계산 모드에서 쿼터 설정

GFS2 파일 시스템에서 기본값으로 쿼터는 비활성화되어 있습니다. 파일 시스템에 대한 쿼터를 활성화하려면, **quota=on** 옵션을 지정하여 파일 시스템을 마운트합니다.

디스크 사용량을 추적하여 제한이나 경고 값을 강제하지 않고 모든 사용자 및 그룹에 대한 쿼터 사용 계산을 관리할 수 있습니다. 이를 실행하려면, **quota=account** 옵션을 지정하여 파일 시스템을 마운트합니다.

사용법

쿼터를 활성화하여 파일 시스템을 마운트하려면, **quota=on** 옵션을 지정하여 파일 시스템을 마운트합니다.

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

쿼터 사용 계산 관리로 파일 시스템을 마운트하려면 쿼터 제한이 강제되어 있지 않아도 **quota=account** 옵션을 지정하여 파일 시스템을 마운트합니다.

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

쿼터를 비활성화하여 파일 시스템을 마운트하려면 **quota=off** 옵션을 지정하여 파일 시스템을 마운트합니다. 이는 기본값 설정입니다.

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

quota={on|off|account}

on - 파일 시스템이 마운트되어 있을 경우 쿼터를 활성화로 지정합니다.

off - 파일 시스템이 마운트되어 있을 경우 쿼터를 활성화 또는 비활성화로 지정합니다.

account - 쿼터 제한이 강제되어 있지 않다라도 사용자 및 그룹 사용량 통계가 파일 시스템에 의해 관리되도록 지정합니다.

BlockDevice

GFS2 파일 시스템이 위치할 블록 장치를 지정합니다.

MountPoint

GFS2 파일 시스템을 마운트할 디렉토리를 지정합니다.

예시

예제에서 **/dev/vg01/lvol0**에 있는 GFS2 파일 시스템은 쿼터를 활성화하여 **/mygfs2** 디렉토리에 마운트되어 있습니다.

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

예제에서 **/dev/vg01/lvol0**에 있는 GFS2 파일 시스템은 쿼터 계산을 사용하여 **/mygfs2** 디렉토리에 마운트되어 있으나 강제되어 있지 않습니다.

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

3.5.1.2. 쿼터 데이터베이스 파일 생성

쿼터가 활성화된 각 파일 시스템이 마운트되면 시스템은 디스크 쿼터를 사용하여 작업 가능합니다. 하지만 파일 시스템 자체는 아직 쿼터를 지원할 준비가 되어 있지 않습니다. 다음 단계로 **quotacheck** 명령을 실행합니다.

quotacheck 명령은 쿼터가 활성화된 파일 시스템을 확인하고 파일 시스템 마다 현재 디스크 사용에 대한 표를 구축합니다. 이러한 표는 디스크 사용량의 운영 체제 복사본을 업데이트하는데 사용됩니다. 또한 파일 시스템은 디스크 쿼터 파일이 업데이트됩니다.

파일 시스템에 쿼터 파일을 생성하려면 **quotacheck** 명령의 **-u** 및 **-g** 옵션을 사용합니다. 사용자 및 그룹

쿼터를 초기화하려면 이러한 두 옵션을 지정해야 합니다. 예를 들어, 쿼터가 **/home** 파일 시스템에 대해 활성화되어 있는 경우 **/home** 디렉토리에 파일을 생성합니다:

```
quotacheck -ug /home
```

3.5.1.3. 사용자 마다 쿼터 할당

마지막 단계는 **edquota** 명령을 사용하여 디스크 쿼터를 지정합니다. 파일 시스템을 사용 계산 모드로 마운트한 경우 (**quota=account** 옵션을 지정), 쿼터는 강제되지 않음에 유의하십시오.

사용자에 대한 쿼터를 설정하려면 쉘 프롬프트에서 root로 다음 명령을 실행하십시오:

```
edquota username
```

쿼터를 필요로 하는 사용자에 대해 이 단계를 수행합니다. 예를 들어, 쿼터가 **/home** 파티션 (아래 예제에서는 **/dev/VolGroup00/LogVol02**)의 **/etc/fstab**에서 활성화되어 있는 경우 **edquota testuser** 명령을 실행하면 편집기에서 다음과 같은 시스템 기본 설정이 표시됩니다:

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

알림:

edquota는 **EDITOR** 환경 변수에 의해 정의된 텍스트 편집기를 사용합니다. 편집기를 변경하려면 **~/.bash_profile** 파일에 **EDITOR** 환경 변수를 원하는 편집기의 전체 경로로 설정합니다.

첫 번째 열에는 쿼터가 활성화된 파일 시스템의 이름이 있습니다. 두 번째 열에서는 사용자가 현재 사용하고 있는 블록 수를 보여줍니다. 다음의 두 열은 파일 시스템에 있는 사용자의 소프트 및 하드 블록 제한을 설정하는데 사용됩니다.

소프트 블록 제한은 사용 가능한 최대 디스크 공간을 정의합니다.

하드 블록 제한은 사용자 또는 그룹이 사용할 수 있는 절대적인 최대 디스크 공간입니다. 이 제한에 도달하면 더 이상 디스크 공간을 사용할 수 없습니다.

GFS2 파일 시스템은 노드에 대한 쿼터를 유지 관리하지 않기 때문에 이 열은 GFS2 파일 시스템에는 해당하지 않아 빈 칸으로 되어 있습니다.

값이 0으로 설정되어 있는 경우, 제한이 설정되지 않은 것입니다. 텍스트 편집기에서 원하는 값으로 변경합니다. 예:

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

사용자에 대한 쿼터가 설정되어 있는지 확인하려면 다음 명령을 사용합니다:

```
quota testuser
```

3.5.1.4. 그룹 마다 쿼터 할당

쿼터는 그룹 별로 할당할 수 있습니다. 파일 시스템을 사용 계산 모드 (**account=on** 옵션을 지정)로 마운트한 경우 쿼터는 강제되지 않는 점에 유의하십시오.

devel 그룹에 대해 그룹 쿼터를 설정하려면 (그룹 쿼터를 설정하기 전 그룹이 존재해야 함), 다음 명령을 사용합니다:

```
edquota -g devel
```

이 명령은 그룹의 기존 쿼터를 텍스터 편집기에 표시합니다:

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

GFS2 파일 시스템은 노드에 대한 쿼터를 유지 관리하지 않기 때문에 이러한 열은 GFS2 파일 시스템에 적용되지 않으며 빈 칸으로 둡니다. 제한을 수정하여 파일을 저장합니다.

그룹 쿼터가 설정되었는지를 확인하려면 다음 명령을 사용합니다:

```
quota -g devel
```

3.5.2. 디스크 쿼터 관리

쿼터를 구현하는 경우 일부 유지 관리가 필요합니다 — 대부분은 쿼터가 초과되었는지 그리고 쿼터가 정확한지를 확인하는 형태입니다.

물론 사용자가 반복적으로 쿼터를 초과하거나 지속적으로 소프트 제한에 도달하는 경우, 사용자 유형과 작업에 디스크 공간이 미치는 영향의 정도에 따라 시스템 관리자는 몇몇 선택할 수 있는 사항이 있습니다. 관리자는 사용자가 사용할 디스크 공간을 절약하는 방법을 알려주거나 사용자의 디스크 쿼터를 증가시킬 수 있는 방법에 대해 도움을 줄 수 있습니다.

repquota 유틸리티를 실행하여 디스크 사용량 보고서를 작성할 수 있습니다. 예를 들어 **repquota /home** 명령은 다음과 같은 출력 결과를 표시합니다:

```
*** 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
```

쿼터가 활성화된 모든 파일 시스템 (option **-a**)의 디스크 사용량 보고서를 표시하려면 다음 명령을 사용합니다:

```
repquota -a
```

보고서는 읽기 쉽지만 몇 가지 설명해 두어야 할 부분이 있습니다. 각 사용자 뒤에 표시되는 **--**는 블록 제한을 초과하는지 여부를 확인할 수 있는 빠른 방법입니다. 블록 소프트 제한을 초과하는 경우 출력의 첫 번째 **-**에 **+**가 표시됩니다. 두 번째 **-**는 inode 제한을 나타내지만 GFS2 파일 시스템은 inode 제한을 지원하지 않기 때문에 이 부분은 **-**의 상태로 남아 있게 됩니다. GFS2 파일 시스템은 유예 기간을 지원하지 않으므로 **grace** 열은 빈 칸으로 둡니다.

기본 파일 시스템과 관계없이 **repquota** 명령을 NFS를 통해 지원되지 않음에 유의하십시오.

3.5.3. 쿼터 정확도 유지

쿼터를 비활성화하여 일정 기간 실행 후 파일 시스템에 있는 쿼터를 활성화하는 경우, **quotacheck** 명령을 실행하여 쿼터 파일 생성, 확인, 복구해야 합니다. 또한 쿼터 파일이 정확하지 않다고 생각되는 경우 **quotacheck**을 실행할 수 도 있습니다. 이는 시스템 크래쉬 후 파일 시스템이 정상적으로 마운트 해제되지 않을 경우 발생할 수 있습니다.

quotacheck 명령에 대한 보다 자세한 내용은 **quotacheck** man 페이지에서 확인하십시오.

참고

디스크 작업에 의해 계산되는 쿼터 값에 차질이 생길 수 있으므로 모든 노드에서 파일 시스템이 비교적 유휴 상태일 때 **quotacheck**을 실행합니다.

3.5.4. quotasync 명령으로 쿼터 동기화

GFS2는 디스크 상의 내부 파일에 모든 쿼터 정보를 저장합니다. GFS2 노드는 파일 시스템이 작성될 때 마다 이러한 쿼터 파일을 업데이트하지 않습니다; 대신 기본값으로 60 초 마다 한 번씩 쿼터 파일을 업데이트 합니다. 이는 노드 사이에서 쿼터 파일로의 작성 경합이 일어나지 않게 하기 위해 필요합니다, 이러한 경합은 실행 속도를 감소시키는 원인이 될 수 있습니다.

사용자 또는 그룹이 쿼터 제한에 도달하면 GFS2는 제한에 초과되지 않게 하기 위해 쿼터 파일 업데이트 간의 시간을 동적으로 감소시킵니다. 쿼터 동기화 사이에서 일반적 시간 주기는 가변 매개 변수 **quota_quantum**이며 **gfs2_tool** 명령을 사용하여 기본값 60초에서 변경될 수 있습니다. 또한 **quota_quantum** 매개 변수는 각 노드 상에서 파일 시스템이 마운트될 때 마다 설정되어야 합니다. (마운트 해제를 통해 **quota_quantum** 매개 변수로의 변경은 지속되지 않습니다.)

quotasync 명령을 사용하여 GFS2의해 실행되는 자동 업데이트 간의 쿼터 정보를 하나의 노드에서 디스크 상의 쿼터 파일로 동기화할 수 있습니다.

사용법

쿼터 정보 동기화하기

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

u

사용자 쿼터 파일을 동기화합니다.

g

그룹 쿼터 파일을 동기화합니다.

a

현재 쿼터가 활성화되어 동기화를 지원하는 모든 파일 시스템을 동기화합니다. -a를 사용하지 않는 경우에는 파일 시스템의 마운트 지점을 지정해야 합니다.

mntpnt

작업을 적용할 GFS2 파일 시스템을 지정합니다.

동기화 시간 조정

```
gfs2_tool settune MountPoint quota_quantum Seconds
```

MountPoint

작업을 적용할 GFS2 파일 시스템을 지정합니다.

Seconds

GFS2로 주기적 쿼터 파일 동기화에 있어서 새 시간 주기를 지정합니다. 값이 적을 수록 경합이 증가되어 실행 속도가 느려질 수 있습니다.

예시

예제에서는 명령이 실행되는 노드에서 디스크 상의 파일 시스템 **/mnt/mygfs2**에 캐시된 더티 쿼터를 동기화합니다.

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

예제에서는 단일 노드 상의 파일 시스템 **/mnt/mygfs2**에 대해 주기적 쿼터 파일 업데이트의 기본값 시간 간격을 한 시간 (3600초)으로 변경하고 있습니다.

```
gfs2_tool settune /mnt/mygfs2 quota_quantum 3600
```

3.5.5. 참고 자료

디스크 쿼터에 대한 보다 상세한 내용은 다음에 있는 명령의 **man** 페이지를 참조하십시오:

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

3.6. 파일 시스템 확장하기

파일 시스템이 위치한 장치를 확장한 후, **gfs2_grow** 명령을 사용하여 GFS2 파일 시스템을 확장합니다. 기존 GFS2 파일 시스템에서 **gfs2_grow** 명령을 실행하여 현재 파일 시스템의 끝과 새로 초기화된 GFS2 파일 시스템이 확장된 장치 끝 사이의 모든 여유 공간을 채웁니다. 채우기 작업이 완료되면, 파일 시스템의 리소스 색인이 업데이트됩니다. 그 후 클러스터에 있는 모든 노드는 추가된 저장 장치 공간을 사용할 수 있게 됩니다.

gfs2_grow 명령은 마운트된 파일 시스템에서 실행되어야 하지만, 클러스터에 있는 하나의 노드에서만 실행되어야 합니다. 모든 다른 노드는 확장 작업이 실행되었음을 감지하고 자동으로 새 공간을 사용합니다.

알림:

mkfs.gfs2 명령으로 GFS2 파일 시스템을 생성하면 파일 시스템 크기를 줄일 수 없습니다.

사용법

```
gfs2_grow MountPoint
```

MountPoint

작업을 적용할 GFS2 파일 시스템을 지정합니다.

주석

gfs2_grow 명령을 실행하기 전:

- 파일 시스템에 있는 중요한 데이터를 백업합니다.
- **df *MountPoint*** 명령을 실행하여 확장된 파일 시스템에 의해 사용되는 볼륨을 지정합니다.
- LVM으로 기본적인 클러스터 볼륨을 확장합니다. LVM 볼륨 관리에 관한 내용은 *LVM 관리자 가이드*에서 참조하시기 바랍니다.

gfs2_grow 명령을 실행한 후, **df** 명령을 실행하여 현재 파일 시스템에서 새 공간을 사용할 수 있는 지를 확인합니다.

예시

예에서, **/mygfs2fs** 디렉토리에 있는 파일 시스템이 확장되었습니다.

```
[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.
```

전체 사용법

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

MountPoint

GFS2 파일 시스템이 마운트된 디렉토리를 지정합니다.

Device

파일 시스템의 장치 노드를 지정합니다.

표 3.3. “파일 시스템을 확장하는 동안에 사용 가능한 GFS2 특정 옵션 ”에서는 GFS2 파일 시스템을 확장하는 동안에 사용할 수 있는 GFS2 특정 옵션에 대해 설명합니다.

표 3.3. 파일 시스템을 확장하는 동안에 사용 가능한 GFS2 특정 옵션

옵션	설명
-h	도움말. 사용법에 대한 간단한 메세지를 보여줍니다.
-q	정숙 모드. 상세 정보 레벨을 낮춥니다.
-r MegaBytes	새 리소스 그룹 크기를 지정합니다. 기본값 크기는 256MB입니다.
-T	테스트. 모든 계산을 실행하지만, 디스크에 어떤 데이터도 작성하지 않으며 파일 시스템을 확장하지 않습니다.
-V	명령 버전 정보를 보여줍니다.

3.7. 파일 시스템에 저널 추가

gfs2_jadd 명령을 사용하여 GFS2 파일 시스템에 저널을 추가합니다. 기본적인 논리 볼륨을 확장하지 않고 아무 지점에서 GFS2 파일 시스템을 저널에 동적으로 추가할 수 있습니다. **gfs2_jadd** 명령은 마운트된 파일 시스템에서 실행되어야 하지만 클러스터에 있는 하나의 노드에서만 실행되어야 합니다. 모든 다른 노드는 확장 작업이 이루어졌음을 감지하게 됩니다.

알림:

GFS2 파일 시스템이 꽉 찬 경우, 파일 시스템이 들어 있는 논리 볼륨이 확장되고 파일 시스템보다 큰 경우에도 **gfs2_jadd**는 실패하게 됩니다. 이는 GFS2 파일 시스템에서 저널은 임베디드 메타데이터가 아닌 일반적인 파일이기 때문입니다. 따라서 기본 논리 볼륨을 간단하게 확장함으로 저널을 위한 공간이 주어지는 것이 아닙니다.

GFS 파일 시스템에 저널을 추가하기 전, **gfs2_tool**의 **journals** 옵션을 사용하여 현재 GFS2 파일 시스템에 있는 저널 수를 알 수 있습니다. 다음 예에서는 **/mnt/gfs2**에 마운트된 파일 시스템에 있는 저널 수 및 크기를 보여주고 있습니다.

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

사용법

```
gfs2_jadd -j Number MountPoint
```

Number

추가될 새 저널 수를 지정합니다.

MountPoint

GFS2 파일 시스템이 마운트된 디렉토리를 지정합니다.

예시

예에서, 하나의 저널이 **/mygfs2** 디렉토리에 있는 파일 시스템에 추가되어 있습니다.

```
gfs2_jadd -j1 /mygfs2
```

예에서, **/mygfs2** 디렉토리에 있는 파일 시스템으로 두 개의 저널이 추가되었습니다.

```
gfs2_jadd -j2 /mygfs2
```

전체 사용법

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

MountPoint

GFS2 파일 시스템이 마운트된 디렉토리를 지정합니다.

Device

파일 시스템의 장치 노드를 지정합니다.

표 3.4. “저널 추가 시 사용 가능한 GFS2 특정 옵션 ”에서는 GFS2 파일 시스템에 저널을 추가할 때 사용할 수 있는 GFS2 특정 옵션을 설명합니다.

표 3.4. 저널 추가 시 사용 가능한 GFS2 특정 옵션

플래그	매개 변수	설명
-h		도움말. 사용법에 대한 간단한 메세지를 보여줍니다.
-J	***MegaBytes***	메가 바이트 단위로 새 저널 크기를 지정합니다. 기본값 저널 크기는 128 메가바이트입니다. 최소 크기는 32 메가바이트입니다. 파일 시스템에 다른 크기의 저널을 추가하려면, 각각의 저널 크기에 대해 **gfs2_jadd** 명령을 실행해야 합니다. 지정된 크기는 반내림되어 파일 시스템 생성시 지정된 다수의 저널 세그먼트 크기가 됩니다.
-j	***Number***	**gfs2_jadd** 명령으로 추가할 새 저널 수를 지정합니다. 기본값은 1입니다.
-q		정숙 모드. 상세 정보 레벨을 낮춥니다.
-V		명령 버전 정보를 보여줍니다.

3.8. 데이터 저널링

일반적으로 GFS2는 메타 데이터만을 저널에 작성합니다. 결과적으로 파일 내용물은 파일 시스템 버퍼를 삭제하는 커널의 주기적 동기화 작업에 의해 디스크에 작성됩니다. 파일 상의 **fsync()** 호출로 파일의 데이터는 디스크로 바로 작성됩니다. 디스크가 모든 데이터가 안전하게 작성되었음을 보고하면 호출이 반환됩니다.

파일 데이터는 메타데이터와 함께 저널에 기록되므로 아주 작은 파일의 경우 데이터 저널링으로 **fsync()** 시간을 감소시킬 수 있습니다. 파일 크기가 커질 수록 이러한 장점은 급속하게 감소됩니다. 데이터 저널링 기능을 사용하면 매체 및 대용량 파일로 쓰기 속도는 상당히 느려지게 됩니다.

파일 데이터를 동기화하기 위해 **fsync()**에 의존하는 어플리케이션은 데이터 저널링 사용으로 인해 성능이 향상되었음을 확인할 수 있습니다. 플래그된 디렉토리 (및 모든 하부 디렉토리)에 생성된 데이터 GFS2 파일에 대해 저널링은 자동으로 활성화될 수 있습니다. 0 값을 갖는 기존 파일은 데이터 저널링을 활성화 또는 비활성화시킬 수 있습니다.

디렉토리에 데이터 저널링을 활성화하는 것은 디렉토리를 "inherit jdata"로 설정하는 것으로 이는 결과적으로 저널링된 디렉토리에 생성된 모든 파일 및 디렉토리를 나타냅니다. **chattr** 명령을 사용하여 파일의 모든 데이터 저널링을 활성화 및 비활성화할 수 있습니다.

다음의 명령으로 **/mnt/gfs2/gfs2_dir/newfile** 파일에 있는 데이터 저널링을 활성화한 후 플래그가 올바르게 설정되어 있는 지를 확인합니다.

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

다음의 명령으로 **/mnt/gfs2/gfs2_dir/newfile** 파일에 있는 데이터 저널링을 비활성화한 후 플래그가 올바르게 설정되어 있는 지를 확인합니다.

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

디렉토리에 **j** 플래그를 설정하기 위해 **chattr**를 사용할 수 있습니다. 디렉토리에 이러한 플래그가 설정되어 있을 때, 결과적으로 디렉토리에 생성된 모든 파일 및 디렉토리는 저널링됩니다. 다음의 명령은 **gfs2_dir** 디렉토리에 **j** 플래그를 설정하고, 플래그가 올바르게 설정되어 있는 지를 확인합니다. 그 후, 이러한 명령은 **/mnt/gfs2/gfs2_dir** 디렉토리에 **newfile**이라는 새 파일을 생성한 후 파일에 **j** 플래그가 설정되었는 지를 확인합니다. 디렉토리에 **j** 플래그가 설정된 후, **newfile**에 대한 저널링을 활성화해야 합니다.

```
[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
```

3.9. atime 업데이트 설정

각각의 파일 inode 및 디렉토리 inode에는 다음과 같은 세 개의 타임 스탬프가 있습니다:

- **ctime** — 마지막으로 inode 상태가 변경된 시간
- **mtime** — 마지막으로 파일 (또는 디렉토리) 데이터가 수정된 시간
- **atime** — 마지막으로 파일 (또는 디렉토리) 데이터가 액세스된 시간

atime 업데이트가 기본값으로 GFS2 및 다른 Linux 파일 시스템에서 처럼 활성화되어 있을 경우 파일을 읽을 때 마다, inode을 업데이트해야 합니다.

일부 어플리케이션은 **atime**에서 제공한 정보를 사용하기 때문에, 이러한 업데이트에서 상당한 량의 불필요한 쓰기 트래픽 및 파일 잠금 트래픽을 요청할 수 있습니다. 이러한 트래픽으로 성능이 감소될 수 있으므로 **atime** 업데이트를 비활성화시키거나 업데이트 빈도수를 감소시키는 것이 좋습니다.

atime 업데이트의 효과를 감소시키는 두 가지 방법이 있습니다:

- 이전 **atime** 업데이트가 **mtime** 또는 **ctime** 업데이트보다 오래된 것일 경우, **atime**을 업데이트하는 **relatime** (relative atime)으로 마운트하기
- 파일 시스템에서 **atime** 업데이트를 비활성화하는 **noatime**으로 마운트하기

3.9.1. relatime으로 마운트하기

relatime (relative atime) Linux 마운트 옵션은 파일 시스템을 마운트할 때 지정할 수 있습니다. 이는 이전 **atime** 업데이트가 **mtime** 또는 **ctime** 업데이트 보다 오래된 것일 경우 **atime**이 업데이트되도록 지정합니다.

사용법

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

BlockDevice

GFS2 파일 시스템이 위치할 블록 장치를 지정합니다.

MountPoint

GFS2 파일 시스템을 마운트할 디렉토리를 지정합니다.

예시

예에서, GFS2 파일 시스템은 **/dev/vg01/lvol0**에 위치하여 **/mygfs2** 디렉토리에 마운트되어 있습니다. 이전 **atime** 업데이트가 **mtime** 또는 **ctime** 업데이트 보다 오래된 것일 경우에만 **atime**이 업데이트됩니다.

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

3.9.2. noatime으로 마운트하기

파일 시스템이 마운트되어 있을 경우 **noatime**Linux 마운트 옵션을 지정할 수 있으며, 이는 파일 시스템에서 **atime** 업데이트를 비활성화합니다.

사용법

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

BlockDevice

GFS2 파일 시스템이 위치할 블록 장치를 지정합니다.

MountPoint

GFS2 파일 시스템을 마운트할 디렉토리를 지정합니다.

예시

예에서, GFS2 파일 시스템은 **/dev/vg01/lvol0**에 위치하고 있으며 **atime** 업데이트를 비활성화하여 **/mygfs2** 디렉토리에 마운트되어 있습니다.

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

3.10. 파일 시스템에 있는 작업 중지하기

gfs2_tool freeze 명령을 사용하여 파일시스템에 쓰기 작업을 중지할 수 있습니다. 쓰기 작업을 중지하면 하드웨어 기반 장치 스냅샷이 일관성있게 파일 시스템을 캡쳐하는데 사용됩니다. **gfs2_tool unfreeze** 명령으로 작업 중지 상태를 종료합니다.

사용법

작업 중지 시작

```
gfs2_tool freeze MountPoint
```

작업 중지 종료

```
gfs2_tool unfreeze MountPoint
```

MountPoint

파일 시스템을 지정합니다.

예시

예에서는 **/mygfs2** 파일 시스템에 쓰기 작업을 중지하고 있습니다.

```
gfs2_tool freeze /mygfs2
```

예에서 **/mygfs2** 파일 시스템에 쓰기 작업 중지를 종료하고 있습니다.

```
gfs2_tool unfreeze /mygfs2
```

3.11. 파일 시스템 복구

노드가 파일 시스템 마운트를 실패했을 경우, 파일 시스템 저널링은 빠른 복구를 허용합니다. 하지만, 저장 장치에 전력이 없거나 물리적으로 접속되지 않았을 경우, 파일 시스템이 손상될 수 있습니다. (저장 장치 하부 시스템에서의 장애를 복구하기 위해 저널링을 사용할 수 없습니다.) 이러한 문제가 발생하면, **fsck.gfs2** 명령을 사용하여 GFS2 파일 시스템을 복구할 수 있습니다.

경고

fsck.gfs2 명령은 모든 노드에서 마운트 해제된 파일 시스템에서만 실행되어야 합니다.

알림:

이전에 GFS 파일 시스템에서 gfs_fsck 명령을 사용해 보셨을 경우, 다음과 같은 방식에서 **fsck.gfs2** 명령은 이전 **gfs_fsck** 릴리즈와 다르다는 점에 유의하시기 바랍니다.

- **fsck.gfs2** 실행 중 **Ctrl+C**를 누르면 명령을 중단하하고 현재 남아있는 경로를 생략하거나 프로세스를 계속할지에 대한 여부를 묻게 됩니다.
- **-v** 플래그를 사용하여 상세 정보 레벨을 증가시킬 수 있습니다. 두 번째 **-v** 플래그를 추가하면 상세 정보 레벨을 증가시키게 됩니다.
- **-q** 플래그를 사용하여 상세 정보 레벨을 감소시킬 수 있습니다. 두 번째 **-q** 플래그를 추가하면 상세 정보 레벨을 감소시키게 됩니다.
- **-n** 옵션은 파일 시스템을 읽기 전용으로 열어 질의에 대해 **no**라고 자동 답변하게 합니다. 이 옵션은 실질적으로 **fsck.gfs2** 명령을 실행하지 않고 오류를 드러내기 위해 명령을 사용하는 방법을 제공합니다.

기타 다른 명령 옵션에 대한 추가 정보는 **fsck.gfs2** 맨 페이지를 참조하시기 바랍니다.

fsck.gfs2 명령을 실행하면 운영체제 및 커널에 사용되는 메모리 보다 상위의 시스템 메모리를 필요로 합니다. GFS2 파일 시스템 자체에 있는 각각의 메모리 블록에는 약 5비트 추가 메모리 또는 5/8 바이트가 필요합니다. 따라서 파일 시스템에서 **fsck.gfs2** 명령을 실행하기 위해 필요한 메모리의 바이트 수를 측정하기 위해 파일 시스템에 들어있는 블록 수를 지정하고 이를 5/8로 곱하면 됩니다.

예를 들어, 블록 하나의 크기가 4K인 16TB의 GFS2 파일 시스템에서 **fsck.gfs2** 명령을 실행하기 위해 필요한 메모리 수를 측정하려면, 먼저 16Tb를 4K로 나누어 파일 시스템에 포함된 메모리의 블록 수를 구합니다:

```
17592186044416 / 4096 = 4294967296
```

이 파일 시스템에는 4294967296 블록이 있으므로, 이 숫자에 5/8을 곱하여 필요한 메모리의 바이트 수를 계산합니다:

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

이 파일 시스템에는 **fsck.gfs2** 명령을 실행하기 위해 약 2.6GB의 여유 메모리가 필요합니다. 블록 크기가 1K일 경우, **fsck.gfs2** 명령을 실행하면 4배의 메모리 또는 11GB의 메모리가 필요합니다.

사용법

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

-y

-y 플래그는 모든 질의에 대해 **yes**라고 답변하게 합니다. **-y** 플래그를 지정하면, **fsck.gfs2** 명령은 변경 사항을 적용하기 전 답변을 요청하지 않게 됩니다.

BlockDevice

GFS2 파일 시스템이 위치할 블록 장치를 지정합니다.

예시

예에서, **/dev/testvol/testlv** 블록 장치에 있는 GFS2 파일 시스템이 복구되어 있습니다. 복구를 위한 모든 질의는 자동으로 **yes**라고 답변하게 되어 있습니다.

```
[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
```

3.12. 바인드 마운트 및 문맥 의존적 경로 이름

GFS2 파일 시스템은 문맥 의존적 경로 이름 (CDPN)을 지원하지 않고, 가변 목적 파일이나 디렉토리로의 심볼릭 링크를 생성하게 합니다. GFS2에서의 이러한 기능을 위해 **mount** 명령의 **bind** 옵션을 사용할 수 있습니다.

mount 명령의 **bind** 옵션은 파일의 본래 위치에서 사용하면서 동시에 다른 위치에 파일 구조의 일부분을 다시 마운트할 수 있게 합니다. 이 명령의 포맷은 다음과 같습니다.

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

이러한 명령을 실행한 후, ***olddir*** 디렉토리의 내용물은 ***olddir*** 및 ***newdir***에서 사용할 수 있게 됩니다. 또한 이 옵션을 사용하여 이 두 개의 디렉토리에서 개별적 파일을 사용 가능하게 할 수 있습니다.

예에서, 다음과 같은 명령을 실행한 후 **/root/tmp** 내용물은 이전에 마운트된 **/var/log** 디렉토리의 내용물과 동일하게 됩니다.

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

다른 방법으로, 마운트 시 동일한 결과를 아카이브하기 위해 **/etc/fstab** 파일에 있는 항목을 사용할 수 있습니다. 다음의 **/etc/fstab** 항목은 **/root/tmp**의 내용물이 **/var/log** 디렉토리의 내용물과 일치하게 합니다.

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

다음의 예에서와 같이, 파일 시스템을 마운트한 후에, **mount** 명령을 사용하여 파일 시스템이 마운트되었는

지를 확인할 수 있습니다.

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

문맥 의존적 경로 이름을 지원하는 파일 시스템을 사용하여, 시스템 아키텍처에 따라, 다음과 같은 경로 중 하나로 **/bin** 디렉토리를 문맥 의존적 경로 이름으로 지정할 수 있습니다.

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

비어있는 **/bin** 디렉토리를 생성하여 이러한 동일한 기능을 아카이브할 수 있습니다. 그 후, 스크립트나 **/etc/fstab** 파일에 있는 항목을 사용하여, **mount -bind** 명령으로 각각의 개별적 아키텍쳐 디렉토리를 **/bin** 디렉토리로 마운트할 수 있습니다. 예를 들어, 스크립트에 다음과 같은 명령행을 사용합니다.

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

다른 방법으로, **/etc/fstab** 파일에서 다음과 같은 항목을 사용할 수 있습니다.

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

바인드 마운트는 지정한 기준 (예: 파일 시스템에 대한 **%fill** 값)에 따라 다른 디렉토리를 마운트할 수 있게 하므로 문맥 의존적 경로 이름보다 더 방대한 유연성을 제공하며, 문맥 의존적 경로 이름은 보다 제한적으로 사용됩니다. 하지만, **%fill** 값과 같은 기준이 따라 마운트하기 위해 스크립트를 직접 작성해야 함에 유의하셔야 합니다.

경고

bind 옵션을 사용하여 파일 시스템을 마운트하며 원래 파일 시스템은 **rw** 마운트되어 있을 때, **ro** 플래그를 사용해도 **ro** 플래그는 무시되어 새 파일 시스템은 **rw** 마운트됩니다. 이러한 경우, 새 파일 시스템은 **/proc/mounts** 디렉토리에 **ro**라고 표시되므로, 잘못 인식될 수 있습니다.

3.13. 바인드 마운트 및 파일 시스템 마운트 순서

mount 명령의 **bind** 옵션을 사용할 때, 파일 시스템이 올바른 순서로 마운트되어 있는지를 반드시 확인해야 합니다. 다음의 예에서 **/var/log** 디렉토리는 **/tmp** 디렉토리에 바인드(bind) 마운트를 실행하기 전 **/var/log** 디렉토리가 마운트되어야 합니다:

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

파일 시스템 마운트 순서는 다음과 같이 결정됩니다:

- 일반적으로 파일시스템 마운트 순서는 **fstab** 파일에 나타난 파일 시스템 순서대로 지정됩니다. **_netdev** 플래그로 마운트된 파일 시스템 또는 자체 **init** 스크립트를 갖는 파일 시스템은 이러한 순서에서 제외됩니다.
- 자체 **init** 스크립트가 있는 파일 시스템은 **fstab** 파일에 있는 파일 시스템 이후에, 초기화 과정 후반에 마운트됩니다.
- **_netdev** 플래그로 마운트된 파일 시스템은 시스템의 네트워크가 활성화되었을 때 마운트됩니다.

GFS2 파일 시스템을 마운트하기 위해 bind 마운트 생성을 필요로하는 설정의 경우 다음과 같이 **fstab** 파일 순서를 지정할 수 있습니다:

1. 바인딩 마운트에 필요한 로컬 파일 시스템을 마운트합니다.
2. GFS2 파일 시스템이 마운트된 디렉토리를 바인드 마운트합니다.
3. GFS2 파일 시스템을 마운트합니다.

로컬 디렉토리 또는 파일 시스템을 GFS2 파일 시스템에 bind 마운트하는 설정이 필요한 경우, **fstab** 파일에 파일 시스템을 올바른 순서로 나열하면 GFS2 **init** 스크립트가 실행될 때 까지 GFS2 파일 시스템은 마운트되지 않기 때문에 파일 시스템을 마운트하지 않게 됩니다. 이러한 경우, bind 마운트를 실행하기 위해 **init** 스크립트를 작성하여 GFS2 파일 시스템이 마운트되기 전 까지 bind 마운트되지 않게 합니다.

다음의 스크립트는 사용자 설정 **init** 스크립트의 예입니다. 이 스크립트는 GFS2 파일 시스템의 두 개의 디렉토리에 두 디렉토리의 바인딩 마운트를 실행합니다. 예에서 **/mnt/gfs2a**에 기존 GFS2 마운트 지점이 있어 이는 클러스터가 시작된 후 GFS2 **init** 스크립트가 실행될 때 마운트됩니다.

예시 스크립트에서 **chkconfig** 문의 값은 다음을 나타냅니다:

- 345는 스크립트가 시작하는 런레벨입니다
- 29는 시작 우선 순위입니다. 이 경우 26 시작 우선 순위를 갖는 GFS2 **init** 스크립트 후 시작 시 스크립트가 실행됨을 나타냅니다.
- 73은 종료 우선 순위입니다. 이 경우 74 종료 우선 순위를 갖는 GFS2 스크립트 전 종료 시 스크립트가 중지됨을 나타냅니다.

시작 및 종료 값은 **service start** 및 **service stop** 명령을 실행하여 나타난 동작을 수동으로 실행할 수 있음을 나타냅니다. 예를 들어, 스크립트 이름이 **fredwilma**일 경우, **service fredwilma start**를 실행할 수 있습니다.

이 스크립트는 **/etc/init.d** 디렉토리에 있는 다른 스크립트와 동일한 권한을 갖는 해당 디렉토리에 배치합니다. 그 후 **chkconfig on** 명령을 실행하여 스트립트를 나타난 런레벨에 연결할 수 있습니다. 예를 들어, 스크립트 이름이 **fredwilma**일 경우, **chkconfig fredwilma on**을 실행합니다.

```
#!/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
```

3.14. GFS2 철회 (Withdraw) 기능

GFS2 *철회 (withdraw)* 기능은 클러스터에 있는 GFS2 파일 시스템의 데이터 무결성 기능입니다. GFS2 커널 모듈이 I/O 작업에 이어 GFS2 파일 시스템에서 불일치를 감지하면, 파일 시스템은 클러스터에 사용할 수 없게 됩니다. I/O 작업을 중지하고 시스템 오류 출력을 위해 I/O 작업을 대기 상태로 두어 더 이상의 피해를 방지합니다. 이러한 문제가 발생하면, 저널을 재생하기 위해 GFS2 파일 시스템을 재부팅하여 다시 마운트한 후 다른 서비스 또는 어플리케이션을 수동으로 중지할 수 있습니다. 문제가 해결되지 않으면, 클러스터의 모든 노드에서 파일 시스템을 다시 마운트하고 **`fsck.gfs2`** 명령으로 파일 시스템 복구를 수행합니다. GFS 철회 기능은 커널 패닉에 비해 덜 심각하지만 다른 노드에서 노드를 차단하는 원인이 될 수 있습니다.

gfs2 시작 스크립트가 활성화되어 있고 GFS2 파일 시스템이 **/etc/fstab** 파일에 포함되어 있도록 시스템이 구성되어 있을 경우, GFS2 파일 시스템은 재부팅시 다시 마운트됩니다. 손상된 파일 시스템이 인식되어 GFS2 파일 시스템으로 withdrew가 발생하는 경우 파일 시스템을 다시 마운트하기 전 **fsck.gfs2** 명령을 실행하는 것이 좋습니다. 이러한 경우, 부팅시 파일 시스템을 다시 마운트하지 않도록 다음과 같은 절차를 실행할 수 있습니다:

1. 다음 명령을 사용하여 영향을 받는 노드에서 시작 스크립트를 일시적으로 해제합니다:

   ```
   # chkconfig gfs2 off
   ```

2. 영향을 받은 노드를 다시 시작하여 클러스터 소프트웨어를 시작합니다. GFS2 파일 시스템은 마운트되지 않습니다.
3. 클러스터에 있는 모든 노드에서 파일 시스템을 마운트 해제합니다.
4. 손상된 파일 시스템이 없는 지를 확인하기 위해 하나의 노드에서의 파일 시스템에서만 **fsck.gfs2**를 실행합니다.
5. 다음 명령을 실행하여 영향을 받는 노드에서 시작 스크립트를 다시 활성화합니다:

   ```
   # chkconfig gfs2 on
   ```

6. 클러스터에 있는 모든 노드에서 GFS2 파일 시스템을 다시 마운트합니다.

GFS2 철회를 초래하는 불일치의 예에는 잘못된 블록 수가 있습니다. GFS 커널이 파일 시스템에서 파일을 삭제할 때, 파일에 관련된 데이터 및 메타데이터 블록을 모두 제거하게 됩니다. 이러한 작업을 완료하면 이는 블록 수를 확인하게 됩니다. 블록 수가 하나가 아닌 경우 (남아있는것이 디스크 inode 자체에 한함을 의미) 블록 수가 감지된 블록 목록과 일치하지 않기 때문에 파일 시스템이 일치하지 않음을 나타내게 됩니다.

지정된 **-o errors=panic** 옵션으로 파일 시스템을 마운트하면 GFS2 철회 기능을 덮어쓰기할 수 있습니다. 이 옵션이 지정되어 있을 경우, 일반적으로 시스템 철회의 원인이 되었던 모든 오류가 시스템 패닉의 원인으로 됩니다. 이는 노드의 클러스터 통신을 중지하여 노드가 고립되게 합니다.

내부적으로 커널이 **gfs_controld** 데몬에 메세지를 전송하고 철회를 요청하면 GFS2 철회 기능이 작동합니다. **gfs_controld** 데몬은 **dmsetup** 프로그램을 실행하여 장치 맵퍼 오류 대상을 파일 시스템 아래에 배치하여 블록 장치에 더 많이 액세스하지 못하게 합니다. 다음으로 데몬은 커널이 동작 완료되었음을 알립니다. 이는 GFS2 지원 필요 요건으로 CLVM 장치를 항상 GFS2에서 사용해야 하는 이유입니다. 그렇지 않으면 장치 맵퍼 대상을 삽입할 수 없습니다.

장치 매퍼 오류 대상의 목적은 이후의 모든 I/O 작업에 I/O 오류가 발생하여 파일 시스템이 순서대로 마운트 해제되는 지를 확인하는 것입니다. 이 때문에 철회가 발생했을 때 장치 매퍼 장치에서 다수의 I/O 오류가 시스템 로그에 보고되는 것은 일반적인 것입니다.

일부 경우 **dmsetup** 프로그램이 요청대로 오류 대상을 삽입할 수 없는 경우에는 철회가 실패할 수 있습니다. 이는 철회 지점에서 메모리 부족이 발생하는 경우와 먼저 철회를 트리거한 문제로 인해 메모리를 다시 사용할 수 없는 경우에 발생할 수 있습니다.

모든 철회가 항상 GFS2에 오류가 있다는 것을 의미하는 것은 아닙니다. 경우에 따라 철회 기능은 기본 블록 장치와 관련된 장치 I/O 오류로 인해 트리거될 수 도 있습니다. 철회가 발생한 경우 이에 해당하는지에 대한 여부를 로그에서 확인하는것이 좋습니다.

4장. GFS2 파일 시스템과 관련된 문제 진단 및 수정

다음 부분에서는 일반적인 GFS2 문제 해결 방법에 대해 설명합니다.

4.1. GFS2 파일 시스템의 성능 저하

GFS2 파일 시스템이 EXT3 파일 시스템 보다 성능이 저하될 수 있습니다. GFS2 성능은 여러 요인 및 특정 사용의 경우에 의해 영향을 받을 수 있습니다. GFS2 성능 문제 해결을 위한 내용은 이 문서를 통해 설명합니다.

4.2. GFS2에서 NFS 설정

GFS2 잠금 하브 시스템과 클러스터된 환경을 통한 추가된 복잡성으로 인해 GFS2에서 NFS를 설정하려면 많은 주의와 신중한 설정이 필요합니다. 다음 부분에서는 GFS2 파일 시스템을 통해 NFS 서버를 설정할 때 고려해야 할 사항에 대해 설명합니다.

경고

GFS2 파일 시스템이 NFS 내보내기하고 있고 NFS 클라이언트 어플리케이션이 POSIX 잠금을 사용하는 경우, **`localflocks`** 옵션을 사용하여 파일 시스템을 마운트해야 합니다. 이것이 의도하는 효과는 각 서버에서 POSIX 잠금이 로컬화 (즉, 클러스터되지 않으면 상호 독립적인 상태) 되도록 강제하는 것입니다. (클러스터 노드 전역에 걸쳐 NFS에서 GFS2가 POSIX 잠금 구현을 시도하는 경우에 여러 문제가 있습니다.) NFS 클라이언트에서 실행되고 있는 어플리케이션의 경우, 로컬화된 POSIX 잠금은 두 클라이언트가 다른 서버에서 마운트될 때 이 두 클라이언트는 동일한 잠금을 동시에 유지할 수 있다는 것을 의미합니다. 모든 클라이언트가 하나의 서버에서 NFS를 마운트하는 경우 별도의 서버가 동일한 잠금을 독립적으로 허용하는데는 문제가 없습니다.

GFS2 파일 시스템에서 NFS 서비스를 설정할 때 잠금 기능을 고려하는 것 이외에 다음과 같은 사항도 고려해야 합니다.

- Red Hat은 다음과 같은 특성의 active/passive 설정의 잠금 기능을 갖는 NFSv3를 사용하는 Red Hat 고가용성 추가 기능 설정만을 지원합니다:
  - 백엔드 파일 시스템은 2 ~ 16 노드 클러스터에서 실행되는 GFS2 파일 시스템입니다.
  - NFSv3 서버는 단일 클러스터 노드에서 GFS2 파일 시스템 전체를 한번에 내보내기할 서비스로 정의됩니다.
  - NFS 서버는 하나의 클러스터 노드에서 다른 클러스터 노드로 장애 조치할 수 있습니다 (active/passive 설정)
  - NFS 서버를 통해 *예외적인* GFS2 파일 시스템으로의 액세스는 허용되지 않습니다. 여기에는 로컬 GFS2 파일 시스템 액세스와 Samba 또는 클러스터된 Samba를 통한 액세스가 모두 포함됩니다.
  - 시스템에서 NFS 쿼터는 지원하지 않습니다.

  이 설정에서는 노드에 장애가 발생해도 하나의 노드에서 다른 노드로 NFS 서버가 장애 조치하여 **`fsck`** 명령을 실행할 필요가 생기지 않기 때문에 파일 시스템에 고가용성을 제공하고 시스템 다운타임을 줄일 수 있습니다.
- **`fsid=`** NFS 옵션은 GFS2의 NFS 내보내기가 필요합니다.
- 클러스터에 문제가 생긴 경우 (예: 클러스터가 쿼럼에 도달하지 않고 펜싱이 실패한 경우)에는 클러스터된 논리 볼륨 및 GFS2 파일 시스템은 동결되고 클러스터가 쿼럼을 충족할 때 까지 액세스할 수 없습니다. 이러한 절차에서 설명한 것과 같은 간단한 장애 조치 해결 방법이 시스템에 가장 적합한지 여부를 결정할 때 이러한 가능성을 고려해야 합니다.

4.3. GFS2 파일 시스템 중단 및 단일 노드 재부팅 요청

GFS2 파일 시스템이 중단되어 이에 대한 실행 명령을 반환하지 않지만 특정 노드를 재부팅하면 시스템이 정상적인 상태로 돌아갈 경우 잠금 문제 또는 버그 징후 가능성이 있습니다. 이러한 문제가 발생한 경우에는 다음과 같은 데이터를 수집합니다:

- 각 노드에서 파일 시스템에 대한 gfs2 잠금 덤프:

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

- 각 노드에서 파일 시스템에 대한 DLM 잠금 덤프: 이 정보는 **dlm_tool** 명령을 사용하여 얻을 수 있습니다.

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

  이 명령 ***lsname***은 문제가 있는 파일 시스템에 대해 DLM이 사용하는 잠금 공간 이름입니다. 이 값은 **group_tool** 명령의 출력 결과에서 확인할 수 있습니다.
- **sysrq -t** 명령의 출력 결과
- **/var/log/messages** 파일의 내용.

데이터를 수집한 후, Red Hat 지원 티켓을 오픈하여 수집한 데이터를 제공합니다.

4.4. GFS2 파일 시스템 중지 및 모든 노드의 재부팅 요청

GFS2 파일 시스템이 중단하여 이에 대한 실행 명령을 반환하지 않고 이를 사용하기전 클러스터에 있는 모든 노드를 재부팅해야 하는 경우 다음 문제를 확인하십시오.

- 차단 장치에 장애가 발생할 수 있습니다. GFS2 파일 시스템은 중지하여 장애가 발생한 차단 장치에 있는 데이터 무결성을 확인하게 됩니다. 메세지 로그를 확인하고 중지 시 차단 장치에 장애가 발생했는지 여부를 확인합니다. 펜싱이 제대로 설정되어 있는지 여부를 확인합니다.
- GFS2 파일 시스템이 해제될 수 있습니다. 메세지 로그에 **withdraw**라는 단어가 있는지 확인하고 GFS에서 파일 시스템이 해제되었다는 메세지 및 호출 추적 여부를 확인합니다. 해제는 파일 시스템 손상, 스토리지 오류 또는 버그 중 하나의 증상입니다. 파일 시스템을 마운트 해제하여 **gfs2-utils** 패키지를 업데이트하고 **fsck** 명령을 파일 시스템에서 실행하여 이를 서비스로 복귀시킵니다. Red Hat 지원 티켓을 오픈하여 GFS2 해제가 발생한 것을 알리고 로그와 함께 sosreport를 보냅니다.

  GFS2 해제 기능에 대한 자세한 내용은 3.14절. “GFS2 철회 (Withdraw) 기능 ”에서 참조하십시오.
- 이러한 오류는 잠금 문제 또는 버그 증상입니다. 이러한 상황이 발생하면 데이터를 수집하여 4.3절. “GFS2 파일 시스템 중단 및 단일 노드 재부팅 요청 ”에서 설명하고 있듯이 Red Hat 지원 티켓을 오픈합니다.

4.5. 새로 추가된 클러스터 노드에 GFS2 파일 시스템이 마운트되지 않음

클러스터에 새로운 노드를 추가하여 해당 노드에 GFS2 파일 시스템을 마운트할 수 없는 경우, GFS2 파일 시스템에 액세스하려는 노드보다 GFS2 파일 시스템에 있는 저널 수가 더 적을 수 있습니다. 파일 시스템을 마운트하려는 GFS2 호스트마다 하나의 저널이 필요합니다. (단 **spectator** 마운트 옵션이 설정된 상태에서 마운트된 GFS2 파일 시스템은 저널이 필요하지 않으므로 제외됩니다.) 3.7절. “파일 시스템에 저널 추가 ”에서 설명하고 있듯이 **gfs2_jadd** 명령을 사용하여 GFS2 파일 시스템에 저널을 추가할 수 있습니다.

4.6. 빈 파일 시스템에서 사용중인 것으로 표시되는 공간

빈 GFS2 파일 시스템이 있는 경우 **df** 명령은 사용되고 있는 공간을 표시합니다. 이는 GFS2 파일 시스템 저널이 디스크 상의 공간 (저널 수 및 저널 크기)을 사용하고 있기 때문입니다. 다수의 저널 수 또는 대형 저널 크기를 사용하여 GFS2 파일 시스템을 생성할 경우 **df** 명령을 실행했을 때 이미 사용 중인 (저널 수 및 저널 크기)를 확인할 수 있습니다. 다수의 저널 또는 대형 저널을 지정하지 않은 경우에도 소형 GFS2 파일 시스템 (1GB 이하의 범위)은 기본 GFS2 저널 크기를 갖는 대량의 공간이 시용 중인것으로 표시됩니다.

gfs2_quota 명령을 사용하여 GFS2 쿼터 관리

Red Hat Enterprise Linux 6.1 릴리즈에서 GFS2는 표준 Linux 쿼터 기능을 지원합니다. 이 기능을 사용하려면 **quota** RPM을 설치해야 합니다. 이는 GFS2에서 권장되는 쿼터 관리 방법이며 쿼터를 사용하는 모든 새로운 GFS2 배포에서 사용해야 합니다. 표준 Linux 쿼터 기능 사용에 대한 자세한 내용은 3.5절. “GFS2 쿼터 관리 ”에서 참조하십시오.

Red Hat Enterprise Linux의 이전 릴리즈에서 GFS2는 **gfs2_quota** 명령을 사용하여 쿼터를 관리해야 했습니다. 여기서는 **gfs2_quota** 명령을 사용하여 GFS2 파일 시스템 쿼터를 관리하는 방법에 대해 설명합니다.

A.1. gfs2_quota 명령을 사용하여 쿼터 설정

각각의 사용자 ID (UID) 또는 그룹 ID (GID)에 대해 *하드 제한* 및 *소프트 제한*이라는 두 개의 쿼터 설정을 사용할 수 있습니다.

하드 제한은 사용할 수 있는 공간의 크기입니다. 파일 시스템은 사용자 또는 그룹 사용자가 디스크 공간 크기를 넘는 용량의 사용은 허용하지 않습니다. 하드 제한 값이 *zero*인 경우 제한이 강제되지 않음을 의미합니다.

일반적으로 소프트 제한은 하드 제한 보다 값이 적습니다. 파일 시스템은 소프트 제한이 사용하고 있는 공간 크기의 소프트 한계에 도달하면 사용자 또는 그룹에게 통지합니다. 소프트 제한 값이 *제로*인 경우 제한이 강제되지 않은 것입니다.

gfs2_quota 명령을 사용하여 제한을 설정할 수 있습니다. 이러한 명령은 GFS2가 마운트된 단일 노드에서만 실행될 수 있습니다.

기본값으로 쿼터 강제는 GFS2 파일 시스템에서 설정되지 않습니다. 쿼터 사용 계산을 활성화하려면, A.4절. “쿼터 강제 활성화/비활성화 ”에서 설명하고 있듯이 GFS2 파일 시스템을 마운트할 때 **mount** 명령의 **quota=**를 사용합니다.

사용법

쿼터 설정, 하드 제한

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

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

쿼터 설정, 경고 제한

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

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

User

제한 또는 경고하는 사용자 ID입니다. 암호 파일에서 사용자 이름 또는 UID 번호를 사용할 수 있습니다.

Group

제한 또는 경고하는 그룹 ID입니다. 그룹 파일에서 그룹 이름이나 GID 번호를 사용할 수 있습니다.

Size

제한 또는 경고를 위한 새로운 값을 지정합니다. 값은 기본적으로 메가 바이트 단위로 되어 있습니다. **-k**, **-s**, **-b** 플래그를 추가하면 단위는 각각 킬로바이트, 섹터, 파일 시스템 블록으로 변경됩니다.

MountPoint

동작을 적용하는 GFS2 파일 시스템을 지정합니다.

예제

다음 예제에서는 파일 시스템 **/mygfs2**에서 사용자 ***Bert***에 대해 1023 메가바이트 (1 기가 바이트)의 하드 제한을 설정하고 있습니다.

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

다음 예제에서는 파일 시스템 **/mygfs2**에서 그룹 ID 21에 대해 50 킬로 바이트의 소프트 제한을 설정하고 있습니다.

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

A.2. gfs2_quota 명령을 사용하여 쿼터 제한 및 사용량 표시

특정 사용자 및 그룹에 대한 쿼터 제한 및 현재 사용량은 **gfs2_quota get** 명령을 사용하여 볼 수 있습니다. 쿼터 파일의 전체 내용은 **gfs2_quota list** 명령을 사용하여 볼 수 있으며, 이 경우, 0 이외의 하드 제한, 소프트 제한, 값을 값는 모든 ID가 나열됩니다.

사용법

사용자 용 쿼터 제한 표시

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

그룹 용 쿼터 제한 표시

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

전체 쿼터 파일 표시

```
gfs2_quota list -f MountPoint
```

User

특정 사용자에 대해 정보를 표시하기 위한 사용자 ID입니다. 암호 파일에서 사용자 이름 또는 UID 번호를 사용할 수 있습니다.

Group

특정 그룹에 대해 정보를 표시하기 위한 그룹 ID입니다. 그룹 파일에서 그룹 이름이나 GID 번호를 사용할 수 있습니다.

MountPoint

동작을 적용하는 GFS2 파일 시스템을 지정합니다.

명령 출력

gfs2_quota 명령을 사용하면 GFS2 쿼터 정보는 다음과 같이 표시됩니다:

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

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

LimitSize, ***WarnSize***, ***Value*** 값은 기본적으로 메가바이트 단위로 되어 있습니다. 명령행에 **-k**, **-s**, **-b** 플래그를 추가하면 단위는 각각 킬로바이트, 섹터, 파일 시스템 블록으로 변경됩니다.

User

데이터가 연결되는 사용자 이름 또는 ID입니다.

Group

데이터가 연결되는 그룹 이름 또는 ID입니다.

LimitSize

사용자 또는 그룹에 대해 설정된 하드 제한입니다. 제한이 설정되지 않은 경우 값은 제로가 됩니다.

Value

사용자 또는 그룹이 사용하는 디스크 공간의 실제 크기입니다.

주석

쿼터 정보를 표시할 때 **-n** 옵션을 명령행에 추가하면 **gfs2_quota** 명령은 UID 및 GID를 이름으로 확인하지 않습니다.

명령행에 **-d** 옵션을 추가하면 GFS2의 숨김 파일에 할당된 공간은 root UID 및 GID에 대해 표시된 값에 제외됩니다. 이는 **gfs2_quota**에서 숫자와 **du** 명령의 출력 결과를 일치시키려 할 때 유용합니다.

예제

다음 예제에서는 제한이 설정되어 있거나 또는 파일 시스템 **/mygfs2**에 있는 디스크 공간을 사용하는 모든 사용자 및 그룹의 쿼터 정보를 표시하고 있습니다.

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

다음 예제에서는 파일 시스템 **/mygfs2**에 있는 그룹 **users**의 쿼터 정보를 섹터 단위로 표시하고 있습니다.

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

A.3. gfs2_quota 명령을 사용하여 쿼터를 동기화

GFS2는디스크 자체의 내부 파일에 모든 쿼터 정보를 저장합니다. GFS2 노드는 모든 파일 시스템 쓰기에 대해 이러한 쿼터 파일을 업데이트하지 않고 기본값으로 매 60초 마다 쿼터 파일을 업데이트합니다. 이는 쿼터 파일에 기록하는 동안 노드 간 충돌에 의한 성능 저하를 방지하기 위해 필요합니다.

사용자 또는 그룹이 쿼터 제한에 도달하면 GFS2는 쿼터 파일 업데이트 간격을 동적으로 단축하여 제한을 넘지 않도록 합니다. 쿼터 동기화 간의 일반적인 간격은 조정 가능한 매개 변수 **quota_quantum**이며 **gfs2_tool** 명령을 사용하여 기본값 60 초에서 변경할 수 있습니다. 또한 **quota_quantum** 매개 변수는 각 노드에 대해 파일 시스템이 마운트될 때 마다 설정해야 합니다. (**quota_quantum** 매개 변수로의 변경은 마운트 해제 시 비활성화됩니다.)

gfs2_quota sync 명령을 사용하여 GFS2에 의해 실행되는 자동 업데이트 간의 쿼터 정보를 노드에서 디스크 쿼터 파일에 동기화할 수 있습니다.

사용법

쿼터 정보 동기화

```
gfs2_quota sync -f MountPoint
```

MountPoint

동작을 적용하는 GFS2 파일 시스템을 지정합니다.

동기화 간 시간 조정

```
gfs2_tool settune MountPoint quota_quantum Seconds
```

MountPoint

동작을 적용하는 GFS2 파일 시스템을 지정합니다.

Seconds

GFS2에 의한 일반 쿼터 파일 동기화 간의 새로운 시간 간격을 지정합니다. 값이 작을수록 충돌이 발생하여 성능이 저화될 가능성이 높습니다.

예제

이 예제에서는 파일 시스템 **/mygfs2**에서 실행되는 노드에서 쿼터 정보를 동기화합니다.

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

다음 예제에서는 단일 노드에 있는 파일 시스템 **/mygfs2**에 대해 일반 쿼터 파일 업데이트 간격을 한 시간 (3600초)로 변경합니다.

```
gfs2_tool settune /mygfs2 quota_quantum 3600
```

A.4. 쿼터 강제 활성화/비활성화

GFS2 파일 시스템에서 쿼터 강제는 기본값으로 비활성화되어 있습니다. 파일 시스템에 쿼터 강제를 활성화하려면, **quota=on** 옵션을 지정하여 파일 시스템을 마운트합니다.

사용법

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

쿼터 강제를 비활성화하여 파일 시스템을 마운트하려면, **quota=off** 옵션을 지정하여 파일 시스템을 마운트합니다. 이는 기본 설정입니다.

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

-o quota={on|off}

파일 시스템을 마운트할 때 쿼터 강제를 활성화 또는 비활성화로 지정합니다.

BlockDevice

GFS2 파일 시스템이 위치한 곳에 블록 장치를 지정합니다.

MountPoint

GFS2 파일 시스템이 마운트되는 곳에 디렉토리를 지정합니다.

예제

예제에서 **/dev/vg01/lvol0**에 있는 GFS2 파일 시스템은 쿼터 강제가 활성화된 **/mygfs2** 디렉토리에 마운트되어 있습니다.

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

A.5. 쿼터 사용 계산 활성화

제한이나 경고 값을 강제하지 않고 디스크 사용량을 기록하여 모든 사용자 및 그룹에 대한 쿼터 사용 계산을 유지 관리할 수 있습니다. 이를 위해 **quota=account** 옵션을 지정하여 파일 시스템을 마운트합니다.

사용법

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

-o quota=account

쿼터 제한이 강제되지 않은 경우에도 사용자 및 그룹 사용량 통계는 파일 시스템에 의해 유지 관리되도록 지정합니다.

BlockDevice

GFS2 파일 시스템이 위치한 곳에 블록 장치를 지정합니다.

MountPoint

GFS2 파일 시스템이 마운트되는 곳에 디렉토리를 지정합니다.

예제에서 **/dev/vg01/lvol0**에 있는 GFS2 파일 시스템은 쿼터 사용 계산이 활성화된 상태에서 **/mygfs2** 디렉토리에 마운트됩니다.

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

GFS에서 GFS2로 파일 시스템 변경

Red Hat Enterprise Linux 6 버전은 GFS 파일 시스템을 지원하지 않으므로 **gfs2_convert** 명령을 사용하여 기존 GFS 파일 시스템을 GFS2 파일 시스템으로 업그레이드해야 합니다. 이러한 변경 과정은 Red Hat Enterprise Linux 6으로 업그레이드하기 전에 Red Hat Enterprise Linux 5 시스템에서 실행해야 함에 유의하십시오.

주의

변환 프로세스를 철회할 수 없고 변환 도중 발생한 오류로 인해 프로그램이 갑자기 중단되어 파일 시스템을 사용할 수 없게 될 수 있으므로 GFS 파일 시스템을 변경하기 전, 파일 시스템을 백업해 두어야 합니다.
GFS 파일 시스템을 변경하기 전에 **gfs_fsck** 명령을 사용하여 파일 시스템을 검사하고 오류를 수정하십시오.
GFS에서 GFS2로 변환이 정전이나 다른 문제로 인해 중단된 경우, 변환 도구를 다시 시작합니다. 변환이 완료될 때 까지 파일시스템에서 **fsck.gfs2** 명령을 실행하지 않습니다.

문맥 의존적 (Context-Dependent) 경로 이름

GFS2 파일 시스템은 문맥 의존적 경로 이름 (CDPN)을 지원하지 않고, 가변 목적 파일이나 디렉토리로의 심볼릭 링크를 생성하게 합니다. GFS2 파일 시스템에서 CDPN으로 이러한 기능을 실행하기 위해 **mount** 명령의 **bind** 옵션을 사용할 수 있습니다.
gfs2_convert 명령은 CDPN을 확인하고 이를 동일한 이름의 빈 디렉토리로 변경합니다. CDPN을 변경하기 위해 바인딩 마운트를 설정하려면 변경하려는 CDPN의 링크 대상에 대한 전체 경로를 알고 있어야 합니다. 파일 시스템을 변환하기 전 **find** 명령을 사용하여 링크를 확인할 수 있습니다.
다음 명령은 **hostname** CDPN을 가리키는 심볼릭 링크를 나열합니다:

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

마찬가지로 다른 CDPN (**mach**, **os**, **sys**, **uid**, **gid**, **jid**)에도 **find** 명령을 실행할 수 있습니다.
CDPN 이름은 **@hostname** 또는 **{hostname}** 형식이 될 수 있으므로 각 변형에 대해 **find** 명령을 실행해야 함에 유의하십시오.
GFS2에서 바인딩 마운트 및 문맥 의존적 경로 이름에 대한 자세한 내용은 3.12절, “바인드 마운트 및 문맥 의존적 경로 이름 ”에서 참조하십시오.

파일 시스템을 완전하게 또는 거의 완전하게 변환할 때, 모든 GFS2 파일 시스템 데이터 구조를 저장할 수 있는 공간이 충분하지 않을 수 있습니다. 이러한 경우, 사용 가능한 공간에 맞게 모든 저널 크기가 균일하게 감소됩니다.

다음 절차를 사용하여 GFS 파일시스템을 GFS2 파일 시스템으로 변경합니다.

1. Red Hat Enterprise Linux 시스템에서 기존 GFS 파일 시스템을 백업합니다.
2. 클러스터에 있는 모든 노드에서 GFS 파일 시스템을 마운트 해제합니다.
3. GFS 파일 시스템에서 **gfs_fsck** 명령을 실행하여 손상된 파일 시스템이 없는 지를 확인합니다.
4. **gfs2_convert *gfsfilesystem***을 실행합니다. ***gfsfilesystem***을 GFS2로 변경하기 전 시스템에 경고 메세지가 나타나서 확인 질문을 하게 됩니다.
5. Red Hat Enterprise Linux 6로 업그레이드하십시오.

다음의 예에서는 블록 장치 **/dev/shell_vg/500g**에 있는 GFS 파일 시스템을 GFS2 파일 시스템으로 변

환하고 있습니다.

```
[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.
```

개정 내역

고침 7-8.400	**2013-10-31**	**Rüdiger Landmann**

Rebuild with publican 4.0.0

고침 7-8	**2012-07-18**	**Anthony Towns**

Rebuild for Publican 3.0

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

Red Hat Enterprise Linux 6.1 초기 릴리즈

문제 해결: #549838
Red Hat Enterprise Linux 6.1의 표준 Linux 쿼터 기능 지원에 대해 문서화.

문제 해결: #608750
GFS2 해제 기능에 대해 설명을 명확히함.

문제 해결: #660364
GFS2 파일 시스템의 최대 크기에 대한 정보를 수정.

문제 해결: #687874
GFS2 문제 해결에 대한 새로운 장을 추가.

문제 해결: #664848
GFS에서 GFS2로 변환하기 전 문맥 의존적 경로 이름을 확인하는 방법에 대한 정보를 추가.

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

Red Hat Enterprise Linux 6 초기 릴리즈

색인

Symbols

M

Q