

커널의 관리, 모니터링 및 업데이트 (MANAGING, MONITORING AND UPDATING THE KERNEL)

RED HAT ENTERPRISE LINUX⁸

Red Hat Enterprise Linux 8에서 Linux 커널을 관리하기 위한 가이드

Red Hat 고객 컨텐츠 서비스

[법적 고지](#)

요약

본 문서에서는 Linux 커널 레벨에서 워크 스테이션을 설정하는 데 필요한 정보를 사용자 및 관리자를 대상으로 설명합니다. 이러한 조정을 통해 성능 향상 문제 해결 및 시스템 최적화가 쉬워집니다.

Red Hat 문서에 대한 피드백

설명서에 대한 의견을 보내 주셔서 감사합니다. 더 나은 방법을 알려주세요. 그렇게 하려면 :

- 특정 글에 대한 간단한 의견을 남겨하려면 다음을 수행합니다.

1. 문서의 표시가 **Multi-page HTML** 형식으로되어 있고, 문서의 오른쪽 상단에 **Feedback** 버튼이 있는지 확인하십시오.
2. 마우스 커서에 댓글을 추가 할 부분을 강조 표시합니다.
3. 텍스트 아래에 표시되는 **Add Feedback** 팝업을 클릭합니다.
4. 표시되는 지침에 따릅니다.

- 보다 상세한 피드백을 실시하는 경우는 **Bugzilla** 의 티켓을 만듭니다.

1. [Bugzilla](#) 의 Web 사이트에 액세스합니다.
2. Component로 **Documentation** 을 선택합니다.
3. **Description** 필드에 문서의 개선에 대한 의견을 기입하십시오. 문서의 해당 부분에 대한 링크를 작성하십시오.
4. **Submit Bug** 을 클릭합니다.

한글번역에 대한 의견, 오역에 대한 피드백은 [KR-Trans-FeedBack](#)으로 의견을 보내주십시오.

제 1 장 Linux 커널 RPM

다음 섹션에서는 Red Hat 이 제공 및 관리하는 Linux 커널 RPM 패키지를 설명합니다.

1.1. RPM 이란?

RPM 패키지는 다른 파일과 메타 데이터 (시스템이 필요로하는 파일에 대한 정보)가 포함 된 파일입니다.

특히 RPM 패키지는 cpio 아카이브로 구성되어 있습니다.

cpio 아카이브에는 다음이 포함됩니다.

- 파일
- RPM 헤더 (패키지 메타 데이터)

rpm 패키지 관리자는이 메타 데이터를 사용하여 종속성 파일의 설치 위치 및 기타 정보를 결정합니다.

RPM 패키지의 유형

RPM 패키지에는 두 가지 유형이 있습니다. 두 유형 모두 동일한 파일 형식과 도구를 사용하지만, 내용이 다르기 때문에 목적이 다릅니다.

● Source RPM (SRPM)

SRPM 에는 소스 코드와 SPEC 파일이 포함되어 있습니다. 여기에는 소스 코드를 바이너리 RPM 을 빌드하는 방법이 적혀 있습니다. 필요에 따라 소스 코드에 대한 패치도 포함되어 있습니다.

● Binary RPM

바이너리 RPM 은 소스 및 패치에서 구축 된 바이너리가 포함되어 있습니다.

1.2 Linux 커널 RPM 패키지의 개요

kernel RPM 은 파일을 포함하지 않는 메타 패키지에서 다음 하위 패키지가 제대로 설치되도록 합니다.

`kernel-core` - 핵심 기능에 필요한 최소한의 커널 모듈이 포함되어 있습니다. 이 서브 패키지만으로 가상화 및 클라우드 환경에서 **Red Hat Enterprise Linux** 8 커널에 빠른 부팅 시간과 작은 디스크 크기를 제공 할 수 있습니다.

`kernel-modules` - 기타 커널 모듈이 포함되어 있습니다.

`kernel-modules-extra` - 드문(rare) 하드웨어의 커널 모듈이 포함되어 있습니다.

위의 작은 `kernel` 서브 패키지를 일부 제공함으로써, 특히 가상 환경 및 클라우드 환경에서 시스템 관리자에게 유지 관리 영역을 축소하는 것을 목표로합니다.

다른 일반적인 커널 패키지의 예는 다음과 같습니다.

`kernel-debug` - 커널 진단에 효과적인 디버깅 옵션이 다수 포함 된 커널이 포함되어 있습니다. 그러나 성능이 저하됩니다.

`kernel-tools` - **Linux** 커널을 조작하는 툴과 지원 문서가 포함되어 있습니다.

`kernel-devel` - `kernel` 패키지에 모듈을 구축하는 데 충분한 커널 헤더와 `makefiles` 이 포함됩니다.

`kernel-abi-whitelists` - **Red Hat Enterprise Linux** 커널 ABI에 대한 정보가 포함되어 있습니다. 여기에는 강화(enforcement)를 지원하기위한 외부 **Linux** 커널 모듈 및 `yum` 플러그인에 필요한 커널 심볼의 목록이 포함되어 있습니다.

`kernel-headers` - **Linux** 커널과 사용자 공간 라이브러리 및 프로그램 사이의 인터페이스를 제공하는 C 헤더 파일이 포함되어 있습니다. 헤더 파일은 대부분의 표준 프로그램을 구축하는 데 필요한 구조와 상수를 정의합니다.

1.3 커널 패키지 내용보기

다음 단계에서는 `rpm` 명령을 사용하여 설치하지 않고 커널 패키지 및 그 서브 패키지의 내용을 표시하는 방법을 설명합니다.

전제 조건

사용하고있는 **CPU** 아키텍처용 `kernel`, `kernel-core`, `kernel-modules`, `kernel-modules-extra` **RPM** 패키지 준비

단계

- kernel 용 모듈을 나열합니다.

```
$ rpm -qlp <kernel_rpm>
(contains no files)
...
```

- kernel-core 모듈을 나열합니다.

```
$ rpm -qlp <kernel-core_rpm>
...
/lib/modules/4.18.0-80.el8.x86_64/kernel/fs/udf/udf.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/kernel/fs/xfs
/lib/modules/4.18.0-80.el8.x86_64/kernel/fs/xfs/xfs.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/kernel/kernel
/lib/modules/4.18.0-80.el8.x86_64/kernel/kernel/trace
/lib/modules/4.18.0-80.el8.x86_64/kernel/kernel/ring_buffer_benchmark.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/kernel/lib
/lib/modules/4.18.0-80.el8.x86_64/kernel/lib/cordic.ko.xz
...
```

- kernel-modules 모듈을 나열합니다.

```
$ rpm -qlp <kernel-modules_rpm>
...
/lib/modules/4.18.0-80.el8.x86_64/kernel/drivers/infiniband/hw/mlx4/mlx4_ib.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/kernel/drivers/infiniband/hw/mlx5/mlx5_ib.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/kernel/drivers/infiniband/hw/qedr/qedr.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/kernel/drivers/infiniband/hw/usnic/usnic_verbs.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/kernel/drivers/infiniband/hw/vmw_pvrdma/vmw_pvrdma.ko.xz
...
```

- kernel-modules-extra 모듈을 나열합니다.

```
$ rpm -qlp <kernel-modules_rpm>
...
/lib/modules/4.18.0-80.el8.x86_64/kernel/drivers/infiniband/hw/mlx4/mlx4_ib.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/kernel/drivers/infiniband/hw/mlx5/mlx5_ib.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/kernel/drivers/infiniband/hw/qedr/qedr.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/kernel/drivers/infiniband/hw/usnic/usnic_verbs.ko.xz
/lib/modules/4.18.0-80.el8.x86_64/kernel/drivers/infiniband/hw/vmw_pvrdma/vmw_pvrdma.ko.xz
...
```

관련 정보

- 이미 설치된 kernel RPM (서브 패키지를 포함)에서 rpm 명령을 사용하는 방법은 man 페이지의 `rpm(8)`을 참조하십시오.
- [RPM 패키지](#)의 개요

제 2 장 yum 을 사용하여 커널 업데이트

다음 섹션에서는 **Red Hat (Red Hat kernel)**을 제공하고 관리하는 **Linux** 커널과 **Red Hat** 커널의 업데이트를 유지하는 방법을 설명합니다. 결과적으로, 이 운영 체제에는 최신 버그 수정 및 성능 향상 패치를 모두 적용되는 새로운 하드웨어와의 호환성이 보장됩니다.

2.1 커널의 개요

커널은 **Linux** 운영 체제의 핵심 부분으로 시스템 자원을 관리하고 하드웨어 응용 프로그램 및 소프트웨어 응용 프로그램 사이의 인터페이스를 설정합니다. **Red Hat** 커널은 업스트림 **Linux** 메인 라인 커널을 기반으로하는 사용자 정의 커널입니다. **Red Hat** 엔지니어는 안정성과 최신 기술 및 하드웨어와의 호환성에 중점을 두고 추가 개발 및 개선을 실시하고 있습니다.

Red Hat은 새로운 커널 버전을 출시하기 전에 커널은 엄격한 품질 보증 테스트를 통과해야 합니다.

Red Hat 커널은 **RPM** 형식으로 패키지화되기 때문에 **yum** 패키지 관리자에 의한 업그레이드 및 확인이 용이합니다.

경고

Red Hat 의해 컴파일되지 않은 커널은 **Red Hat**은 지원되지 않습니다.

2.2. yum 이란?

이 섹션에서는 **yum 패키지 관리자**의 설명합니다.

관련 정보

- **yum** 자세한 내용은 ["기본 시스템 설정 구성"](#)의 관련 섹션을 참조하십시오.

2.3 커널 업데이트

다음 단계에서는 **yum** 패키지 매니저를 사용하여 커널을 업데이트하는 방법을 설명합니다.

단계

1. 커널을 업데이트하려면 다음을 수행합니다.

```
# yum update kernel
```

이 명령은 커널과 사용 가능한 최신 버전으로 모든 종속성을 업데이트합니다.

2. 시스템을 다시 시작하여 변경 사항을 적용합니다.

참고

Red Hat Enterprise Linux 7에서 Red Hat Enterprise Linux 8로 업그레이드하는 경우 ["RHEL 8로 업그레이드"](#)의 관련 섹션을 참조하십시오.

2.4 커널의 설치

다음 단계에서는 **yum** 패키지 매니저를 사용하여 새로운 커널을 설치하는 방법을 설명합니다.

단계

- 특정 커널 버전을 설치하려면 다음을 사용합니다.

```
# yum install kernel-{version}
```

관련 정보

- 사용 가능한 커널 목록은 ["Red Hat Code Browser"](#)를 참조하십시오.
- 특정 커널 버전 출시일 목록은 [이 문서](#)를 참조하십시오.

제 3 장 커널 모듈 관리

다음 섹션에서는 커널 모듈 개요, 커널 모듈의 표시 방법, 커널 모듈에서의 기본적인 관리 작업을 수행하는 방법을 설명합니다.

3.1 모듈 소개

Red Hat Enterprise Linux 커널은 시스템을 다시 시작하지 않고도 커널 모듈이라는 추가 기능으로 확장 할 수 있습니다. Red Hat Enterprise Linux 8에서 커널 모듈은 추가 커널 코드로 압축 된 `<KERNEL_MODULE_NAME>.ko.xz` 오브젝트 파일에 포함되어 있습니다.

커널 모듈에 의해 사용 된 가장 일반적인 기능은 다음과 같습니다.

- 새로운 하드웨어에 대한 지원을 추가하는 장치 드라이버
- GFS2 와 NFS 같은 파일 시스템 지원
- 시스템 콜(System Call)

최신 시스템은 필요에 따라 자동으로 커널 모듈이 로드됩니다. 그러나 경우에 따라서는 모듈을 수동으로 로드하거나 언로드해야합니다.

모듈은 커널 자체뿐만 아니라 필요에 따라 그 동작을 사용자 정의 매개 변수를 받을 수 있습니다.

도구는 현재 실행중인 모듈, 커널에 로드 할 수 있는 모듈 및 모듈이 허용하는 매개 변수를 확인할 수 있습니다. 이 도구는 실행중인 커널 모듈 로드 및 언로드를 위한 메커니즘을 제공합니다.

3.2 커널 모듈 종속성

특정 커널 모듈은 여러 다른 커널 모듈에 의존하는 경우가 있습니다.

다. `/lib/modules/<KERNEL_VERSION>/modules.dep` 파일에는 각 커널 버전에 대한 커널 모듈 종속성의 전체 목록이 포함되어 있습니다.

종속 파일은 `kmod` 패키지의 일부인 `depmod` 프로그램에 의해 생성됩니다. `kmod`에 의한 유ти리티의 많은 작업을 수행 할 때 모듈 종속성을 고려 때문에 **수동**으로 종속성을 추적 할 필요는 거의 없습니다.

경고

커널 모듈의 코드는 커널 모드에서 무제한(**UNRESTRICTED**) 모드로 실행됩니다. 이 때문에 어떤 모듈을 로드하고 있는지 주의해야 합니다.

관련 정보

- `/lib/modules/<KERNEL_VERSION>/modules.dep` 자세한 내용은 `modules.dep(5)man` 페이지를 참조하십시오.
- `depmod` 개요 및 옵션의 자세한 내용은 `man` 페이지의 `depmod(8)`을 참조하십시오.

3.3. 현재 로드 된 커널 모듈 목록보기

다음 단계에서는 현재 로드 된 커널 모듈을 확인하는 방법을 설명합니다.

전제 조건

- `kmod` 패키지가 설치되어 있다.

단계

- 현재로드 된 커널 모듈을 나열하려면 다음 명령을 실행합니다.

```
$ lsmod
```

Module	Size	Used by
fuse	126976	3
uinput	20480	1
xt_CHECKSUM	16384	1
ipt_MASQUERADE	16384	1
xt_conntrack	16384	1
ipt_REJECT	16384	1
nft_counter	16384	16
nf_nat_tftp	16384	0
nf_conntrack_tftp	16384	1 nf_nat_tftp
tun	49152	1
bridge	192512	0
stp	16384	1 bridge
llc	16384	2 bridge, stp
nf_tables_set	32768	5
nft_fib_inet	16384	1
...		

위의 예 :

- 첫 번째 열에는 현재로드 된 모듈의 **이름**을 나타냅니다.
- 다음 칼럼은 각 모듈의 **메모리** 용량을 킬로바이트 단위로 표시합니다.
- 마지막 칼럼은 숫자와 선택적으로 특정 모듈에 **의존** 하는 모듈 이름을 표시합니다.

관련 정보

- kmod 자세한 내용은 /usr/share/doc/kmod/README 파일 또는 lsmod (8)man 페이지를 참조하십시오.

3.4 모듈 정보보기

커널 모듈로 작업 할 때 해당 모듈에 대한 추가 정보가 표시되는 경우가 있습니다. 이 단계에서는 커널 모듈에 대한 추가 정보를 표시하는 방법을 설명합니다.

전제 조건

- kmod 패키지가 설치되어 있어야 함.

단계

- 커널 모듈에 대한 정보를 표시하려면 다음을 수행합니다.

```
$ modinfo <KERNEL_MODULE_NAME>

For example:
$ modinfo virtio_net

filename:      /lib/modules/4.18.0-94.el8.x86_64/kernel/drivers/net/virtio_net.ko.xz
license:       GPL
description:   Virtio network driver
rhelversion:   8.1
srcversion:    2E9345B281A898A91319773
alias:        virtio:d00000001v*
depends:      net_failover
intree:       Y
name:         virtio_net
vermagic:     4.18.0-94.el8.x86_64 SMP mod_unload modversions
...
parm:          napi_weight:int
parm:          csum:bool
parm:          gso:bool
```

```
parm:          napi_tx:bool
```

`modinfo` 명령은 지정한 커널 모듈에 대한 자세한 정보를 표시합니다. 로드되어 있는지 여부에 관계없이 사용 가능한 모든 모듈의 정보를 조회 할 수 있습니다. `parm` 항목은 사용자가 모듈에 설정할 수있는 매개 변수와 예상되는 값의 유형을 나타냅니다.

참고

커널 모듈의 이름을 입력 할 때, `.ko.xz` 확장자는 이름의 끝에 추가하지 마십시오. 커널 모듈 이름에는 파일 확장명이 없습니다. 그러나 해당 파일에는 확장자가 있습니다.

관련 정보

- `modinfo` 자세한 내용은 `man` 페이지의 `modinfo(8)`을 참조하십시오.

3.5 시스템 런타임 커널 모듈로드

`Linux` 커널의 기능을 확장하는 가장 좋은 방법은 커널 모듈을 로드하는 것입니다. 다음 단계에서는 `modprobe` 명령을 사용하여 커널 모듈을 감지하고 현재 실행중인 커널에 로드하는 방법을 설명합니다.

전제 조건

- `root` 권한
- `kmod` 패키지가 설치되어있음.
- 해당 커널 모듈이 로드되지 않았습니다. 이 경우 로드 된 커널 모듈을 나열하십시오.

단계

1. 로드 할 커널 모듈을 선택합니다.

모듈은 `/lib/modules/$(uname -r)/kernel/<SUBSYSTEM>/` 디렉토리에 있습니다.

2. 해당 커널 모듈을 로드합니다.

```
# modprobe < MODULE_NAME >
```

참고

커널 모듈의 이름을 입력 할 때, .ko.xz 확장자는 이름의 끝에 추가하지 마십시오. 커널 모듈 이름에는 파일 확장명이 없습니다. 그러나 해당 파일에는 확장자가 있습니다.

- 필요한 경우 관련 모듈이 로드되었는지 확인합니다.

```
$ lsmod | grep < MODULE_NAME >
```

모듈이 올바르게 로드 된 경우이 명령은 관련 커널 모듈을 표시합니다. 다음 예를 나타냅니다.

```
$ lsmod | grep serio_raw  
serio_raw 16384 0
```

중요

이 절차에 설명 된 변경 사항은 시스템을 재시작하는 **유지되지 않습니다**.

관련 정보

- `modprobe` 자세한 내용은 `man` 페이지의 `modprobe(8)`을 참조하십시오.

3.6 시스템 런타임 커널 모듈 언로드

때로는 실행 중인 커널에서 특정 커널 모듈을 언로드 할 필요성을 느끼게 될 것입니다. 다음 단계에서는 `modprobe` 명령을 사용하여 현재 로드 된 커널에서 시스템의 실행 시에 커널 모듈을 찾아 언로드하는 방법을 설명합니다.

전제 조건

- `root` 권한
- `kmod` 패키지가 설치되어 있음.

단계

1. `lsmod` 명령을 실행하여 언로드하는 커널 모듈을 선택합니다.

커널 모듈에 의존 관계 (**dependencies**)는 커널 모듈을 언로드하기 전에이 언로드합니다. 종속 모듈의 구체적인 내용은 "[현재로드 된 커널 모듈 목록](#)"을 참조하십시오.

2. 해당 커널 모듈을 언로드합니다.

```
# modprobe -r < MODULE_NAME >
```

커널 모듈의 이름을 입력 할 때, `.ko.xz` 확장자는 이름의 끝에 추가하지 마십시오. 커널 모듈 이름에는 파일 확장명이 없습니다. 그러나 해당 파일의 확장자가 있습니다.

경고

실행중인 시스템에서 사용되는 경우는 커널 모듈을 언로드하지 마십시오. 이렇게하면 시스템이 불안정하거나 작동하지 못하게 할 수 있습니다.

3. 필요한 경우 관련 모듈이 언로드 된 것을 확인합니다.

```
$ lsmod | grep < MODULE_NAME >
```

모듈이 성공적으로 언로드 된 경우이 명령은 출력을 표시하지 않습니다.

중요

이 절차를 완료하면 시스템 시작시 자동으로 로드되도록 정의 된 커널 모듈은 시스템을 다시 시작해도 **언로드되지 않습니다**. 이 결과를 추적하는 방법은 [시스템 부팅시 커널 모듈이 자동으로 로드되지 않도록 하려면](#)을 참조하십시오.

관련 정보

- `modprobe` 자세한 내용은 `man` 페이지의 `modprobe(8)`을 참조하십시오.

3.7 시스템 부팅시 커널 모듈 자동 로드

다음 단계에서는 부팅 과정에서 자동으로 로드되도록 커널 모듈을 설정하는 방법을 설명합니다.

전제 조건

- root 권한
- kmod 패키지가 설치되어있음.

단계

1. 시작 프로세스 중에 로드 커널 모듈을 선택합니다.

모듈은 /lib/modules/\$(uname -r)/kernel/<SUBSYSTEM>/디렉토리에 있습니다.

2. 모듈 설정 파일을 만듭니다.

```
# echo < MODULE_NAME >> /etc/modules-load.d/ < MODULE_NAME > .conf
```

참고

커널 모듈의 이름을 입력 할 때, .ko.xz 확장자는 이름의 끝에 추가하지 마십시오. 커널 모듈 이름에는 파일 확장명이 없습니다. 그러나 해당 파일에는 확장자가 있습니다.

3. 필요한 경우 관련 모듈이 로드되었는지 확인합니다.

```
$ lsmod | grep < MODULE_NAME >
```

위의 예제 명령은 성공하고 관련 커널 모듈을 표시합니다.

중요

이 단계에서 설명하는 변경 사항은 시스템을 다시 시작해도 **지속됩니다**.

관련 정보

- 시스템의 부팅 과정에서 커널 모듈로드에 대한 자세한 내용은 `man` 페이지의 `modules-load.d` (5)을 참조하십시오.

3.8 시스템 부팅시 커널 모듈이 자동 로드 방지

다음 단계에서는 시스템의 부팅 과정에서 커널 모듈이 자동으로 로드되지 않도록 블랙리스트에 추가하는 방법을 설명합니다.

전제 조건

- `root` 권한
- `kmod` 패키지가 설치되어있음.
- 블랙리스트에 지정한 커널 모듈이 현재의 시스템 설정에 중요하지 않은 것을 확인합니다.

단계

1. 블랙리스트에 등록하는 커널 모듈을 선택합니다.

```
$ lsmod
Module           Size  Used by
fuse            126976  3
xt_CHECKSUM     16384  1
ipt_MASQUERADE 16384  1
uinput          20480  1
xt_conntrack    16384  1
...
...
```

`lsmod` 명령은 현재 실행중인 커널에로드 된 모듈 목록을 표시합니다.

- 또는 읽기를 저지하기 언로드 한 커널 모듈을 식별합니다.

모든 커널 모듈 `/lib/modules/<KERNEL_VERSION>/kernel/<subsystem>/` 디렉토리에 있습니다.

2. 블랙리스트 설정 파일을 만듭니다.

```
# vim /etc/modprobe.d/blacklist.conf
# Blacklists <KERNEL_MODULE_1>
blacklist <MODULE_NAME_1>
install <MODULE_NAME_1> /bin/false
```

```
# Blacklists <KERNEL_MODULE_2>
blacklist <MODULE_NAME_2>
install <MODULE_NAME_2> /bin/false

# Blacklists <KERNEL_MODULE_n>
blacklist <MODULE_NAME_n>
install <MODULE_NAME_n> /bin/false
...
```

이 예에서는 `vim` 편집기에서 편집 한 `blacklist.conf` 파일의 내용을 보여줍니다. `blacklist` 행은 시스템의 부팅 과정에서 관련 커널 모듈이 자동으로 로드되지 않도록 지정합니다. 그러나 `blacklist` 명령은 블랙리스트에 없는 다른 커널 모듈 종속성으로 모듈 로드를 막을 수는 없습니다. 따라서 `install` 행에서는 모듈 설치 대신 `/bin/false` 실행됩니다.

해시 기호로 시작하는 행은 파일이 더 읽기 쉬게 만드는 주석입니다.

참고

커널 모듈의 이름을 입력 할 때, `.ko.xz` 확장자는 이름의 끝에 추가하지 마십시오. 커널 모듈 이름에는 파일 확장명이 없습니다. 그러나 해당 파일의 확장자가 있습니다.

3. 재구성하기 전에 현재 초기 `ramdisk` 이미지의 백업 복사본을 만듭니다.

```
# cp /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r).bak.$(date +%-m-%d-%H%M%S).img
```

위의 명령은 새로운 버전에 예기치 않은 문제가 발생했을 경우를 대비해 백업 `initramfs` 이미지를 생성합니다.

- 또는 커널 모듈을 블랙리스트에 지정하는 커널 버전에 해당하는 기타 초기 `ramdisk` 이미지의 백업 복사본을 만듭니다.

```
# cp /boot/initramfs-<SOME_VERSION>.img /boot/initramfs-<SOME_VERSION>.img.bak.$(date +%-m-%d-%H%M%S)
```

4. 변경 사항을 반영하기 위해 새로운 초기 `ramdisk` 이미지를 생성합니다.

```
# dracut -f -v
```

- 현재 실행중인 것과 다른 커널 버전의 초기 `ramdisk` 이미지를 구축 할 경우 대상 `initramfs` 과 커널 버전을 모두 지정합니다.

```
# dracut -f -v / boot / initramfs- < TARGET_VERSION > .img \
< CORRESPONDING_TARGET_KERNEL_VERSION >
```

5. 시스템을 다시 시작합니다.

```
$ reboot
```

중요

- 이 단계에서 설명하는 변경 사항은 시스템을 다시 시작해도 **지속됩니다**. 주요 커널 모듈을 올바르게 블랙리스트에 지정하지 않으면 시스템이 불안정 해 지거나 시스템이 작동하지 못하게 할 수 있습니다.

관련 정보

- `dracut` 유틸리티의 자세한 내용은 `dracut(8)man` 페이지를 참조하십시오.

제 4 장 커널 명령 줄 매개 변수 설정

커널 명령 줄 매개 변수는 시스템 시작시 **Red Hat Enterprise Linux** 커널의 특정 측면의 동작을 변경하는 수단의 하나입니다. 시스템 관리자는 시스템 시작시에 설정되는 옵션을 완전히 제어 할 수 있습니다. 특정 커널의 동작은 시스템 시작시에만 구성 할 수 있으므로이 변경하는 방법을 이해하는 것이 관리 기술의 핵심입니다.

중요

커널 명령 줄 매개 변수를 변경하여 시스템의 동작을 변경하는 옵션은 시스템에 악영향을 미칠 가능성이 있습니다. 따라서 프로덕션 환경에 배포하기 전에 변경을 테스트해야합니다. 자세한 지침은 **Red Hat** 지원팀에 문의하십시오.

4.1 커널 명령 줄 매개 변수 설정

커널 명령 줄 매개 변수는 다음 시스템 시작 시간 설정에 사용됩니다.

- **Red Hat Enterprise Linux** 커널
- 초기 **RAM** 디스크
- 사용자 영역 기능

커널 부팅 시간 매개 변수는 종종 기본값을 덮어 쓰고 특정 하드웨어 설정을 설정하는 데 사용됩니다.

기본적으로 **GRUB2** 부트 로더를 사용하는 시스템의 커널 명령 줄 매개 변수는 모든 커널 부팅 항목의 `/boot/grub2/grubenv` 파일의 `kernelopts` 변수로 정의됩니다.

참고

IBM Z는 `zipl` 부트 로더 환경 변수에 대응하고 있지 않기 때문에, 커널 명령 줄 매개 변수는 부트 항목 설정 파일에 저장됩니다. 따라서 `kernelopts` 환경 변수는 사용할 수 없습니다.

관련 정보

- 변경 가능한 커널 명령 줄 매개 변수에 대한 자세한 내용은 **man** 페이지 `kernel-command-line(7)`, `bootparam(7)` 및 `dracut.cmdline (7)`을 참조하십시오.

4.2. grubby 란?

grubby 부트 로더 고유의 설정 파일을 조작하는 유ти리티입니다.

grubby 기본 부팅 항목을 변경하거나 GRUB2 메뉴 항목에서 인수 추가/제거에도 사용할 수 있습니다.

자세한 내용은 **man** 페이지의 **grubby(8)**을 참조하십시오.

4.3. 부트 항목의 개요

부트 항목(**boot entry**)은 설정 파일에 포함 된 특정 커널 버전에 관련된 옵션의 집합입니다. 사실, 부트 항목은 시스템 커널이 설치되어있는 수만큼 있습니다. 부팅 항목의 설정 파일 **/boot/loader/entries/directory**에 다음과 같이됩니다.

```
6f9cc9cb7d7845d49698c9537337cedc-4.18.0-5.el8.x86_64.conf
```

위의 파일 이름은 **/etc/machine-id** 파일에 저장된 컴퓨터 **ID** 와 커널 버전으로 구성됩니다.

부트 항목의 설정 파일은 커널 버전 초기 **ramdisk** 이미지 및 **kernelopts** 환경 변수 (커널 명령 줄 매개 변수를 포함)에 대한 정보가 포함되어 있습니다. 부팅 항목의 설정 내용은 다음에서 확인할 수 있습니다.

```
title Red Hat Enterprise Linux (4.18.0-74.el8.x86_64) 8.0 (Ootpa)
version 4.18.0-74.el8.x86_64
linux /vmlinuz-4.18.0-74.el8.x86_64
initrd /initramfs-4.18.0-74.el8.x86_64.img $tuned_initrd
options $kernelopts $tuned_params
id rhel-20190227183418-4.18.0-74.el8.x86_64
grub_users $grub_users
grub_arg --unrestricted
grub_class kernel
```

kernelopts 환경 변수는 **/boot/grub2/grubenv** 파일에서 정의됩니다.

4.4 커널 명령 줄 매개 변수 설정

본 섹션에서는 AMD64 및 Intel 64 아키텍처 64 비트 ARM 아키텍처 및 IBM Power Systems의 little-endian 변형에서 커널 명령 줄 매개 변수를 변경하는 방법을 설명합니다.

4.4.1. 모든 부트 항목에서 커널 명령 줄 매개 변수 변경

이 단계에서는 시스템에 있는 모든 부트 항목의 커널 명령 줄 매개 변수를 변경하는 방법을 설명합니다.

전제 조건

- 커널 명령 줄 매개 변수의 개요

단계

1. vim 편집기에서 /etc/default/grub 파일을 엽니다.

```
# vim /etc/default/grub
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root
rd.lvm.lv=rhel/swap rhgb quiet"
GRUB_DISABLE_RECOVERY="true"
GRUB_ENABLE_BLSCFG=true
```

2. GRUB_CMDLINE_LINUX 행에서 매개 변수를 추가, 편집 또는 삭제합니다.

3. GRUB2 설정 파일을 업데이트합니다.

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

4. 시스템을 다시 시작하여 변경 사항을 적용합니다.

그 결과, 부트 로더를 다시 설정 지정된 커널 명령 줄 매개 변수가 적용됩니다.

관련 정보

GRUB2 설정 파일을 사용하여 커널 매개 변수를 수정하는 방법은 "[메뉴 항목 편집](#)" 을 참조하십시오.

4.4.2 하나의 부트 항목에서 커널 명령 줄 매개 변수 변경

이 단계에서는 시스템의 부트 항목의 커널 명령 줄 매개 변수를 변경하는 방법을 설명합니다.

전제 조건

- [커널 명령 줄 매개 변수](#)의 개요
- [grubby\(8\) man](#) 페이지

단계

- 매개 변수를 추가하려면 다음 명령을 실행합니다.

```
# grubby --update-kernel = / boot / vmlinuz-$ (uname -r) --args = "< NEW_PARAMETER >"
```

- 매개 변수를 제거하려면 다음 명령을 사용합니다.

```
# grubby --update-kernel = / boot / vmlinuz-$ (uname -r) --remove-args = "< PARAMETER_TO_REMOVE >"
```

참고

기본적으로 각 커널 부팅 항목에 options 매개 변수가 있고, 이것은 kernelopts 변수로 설정됩니다. 이 변수는 /boot/grub2/grubenv 설정 파일에 정의됩니다.

중요

grubby 유ти리티를 사용하여 특정 부트 항목을 수정하면 수정 한 kernelopts 내용은 관련 커널 부팅 항목에 포함되어 에서 설정 한 값을 덮어 쓰게됩니다. ./boot/loader/entries/<RELEVANT_KERNEL_BOOT_ENTRY.conf>/boot/grub2/grubenvkernelopts

관련 정보

grubby 다른 예는 "[grubby 도구](#)" 를 참조하십시오.

제 5 장 런타임시 커널 매개 변수 설정

시스템 관리자는 런타임시에 Red Hat Enterprise Linux 커널의 동작을 다수 변경할 수 있습니다. 본 절에서는 `sysctl` 명령을 사용하여 `/etc/sysctl.d/` 및 `/proc/sys/` 디렉토리의 설정 파일을 수정하여 런타임시에 커널 매개 변수를 설정하는 방법을 설명합니다.

5.1 커널 매개 변수란?

커널 매개 변수(**kernel parameters**)는 시스템이 실행되는 동안 조정할 수 있는 조정 가능한 값입니다. 변경 사항을 적용하는 경우에도 커널을 다시 시작하거나 다시 컴파일 할 필요가 없습니다.

다음을 사용하여 커널 매개 변수에 대응할 수 있습니다.

+ * `sysctl` 명령 * `/proc/sys/` 디렉토리에 마운트 된 가상 파일 시스템 * `/etc/sysctl.d/` 디렉토리의 설정 파일

조정 가능한 매개 변수는 커널 서브 시스템에서 클래스로 분할됩니다. Red Hat Enterprise Linux에는 다음의 조절 가능한 클래스가 있습니다.

표 5.1 `sysctl` 클래스 표

조정 가능한 클래스(Tunable class)	서브시스템 (Subsystem)
abi(Application Binary Interface)	실행 도메인 및 특성 (Execution domains and personalities) * Linux는 각 프로세스마다 다른 실행 도메인 또는 특성을 지원함.
crypto	암호화 인터페이스(Cryptographic interfaces)
debug	커널 디버깅 인터페이스(Kernel debugging interfaces)
dev	장치별 정보(Device-specific information)
fs	글로벌 및 특정 파일 시스템 조정(Global and specific file system tunables)
kernel	글로벌 커널 조정(Global kernel tunables)

조정 가능한 클래스(Tunable class)	서브시스템 (Subsystem)
net	네트워크 조정(Network tunables)
sunrpc	SUN 원격 프로시저 호출(Sun Remote Procedure Call (NFS))
user	사용자 네임 스페이스 제한 (User Namespace limits)
vm	메모리, 버퍼 및 캐시 조정 및 관리 (Tuning and management of memory, buffers, and cache)

관련 정보

- sysctl 자세한 내용은 `sysctl(8)man` 페이지를 참조하십시오.
- `/etc/sysctl.d/`자세한 내용은 `man` 페이지의 `sysctl.d(5)`을 참조하십시오.

5.2 런타임 커널 매개 변수 설정

중요

프로덕션 시스템에서 커널 매개 변수를 설정하려면 신중한 계획이 필요합니다. 계획이 결여 된 변경은 커널이 불안정 해지고 시스템을 다시 시작해야 할 수 있습니다. 커널 값을 변경하기 전에 유효한 옵션을 사용하고 있는지 확인하십시오.

5.2.1. sysctl 커널 매개 변수의 임시 설정

다음 단계에서는 `sysctl` 명령을 사용하여 런타임에 커널 매개 변수를 일시적으로 설정하는 방법을 설명합니다. 이 명령은 조정 가능한 변수 목록보기 및 필터링에 편리합니다.

전제 조건

- [커널 매개 변수의 개요](#)
- root 권한

단계

- 모든 매개 변수와 그 값을 나열하려면 다음을 수행합니다.

```
# sysctl -a
```

참고

sysctl -a 명령은 런타임 및 시스템 시작시 조정할 수 있는 커널 매개 변수를 표시합니다.

- 매개 변수를 일시적으로 설정하려면 다음 예와 같이 명령을 실행합니다.

```
# sysctl <TUNABLE_CLASS>.<PARAMETER>=<TARGET_VALUE>
```

위의 예제 명령은 시스템이 실행되는 동안 매개 변수 값을 변경합니다. 이 변경은 재부팅없이 바로 적용됩니다.

참고

변경은 시스템 재부팅 후 기본값으로 돌아갑니다.

관련 정보

- sysctl 자세한 내용은 man 페이지의 `sysctl(8)`을 참조하십시오.
- 커널 매개 변수를 영구적으로 변경하려면 `sysctl` 명령을 사용하여 `/etc/sysctl.conf` 파일에 값을 기록하거나 [/etc/sysctl.d/디렉토리](#)의 설정 파일에 수동으로 변경합니다.

5.2.2. sysctl 커널 매개 변수를 영구적으로 설정

다음 단계에서는 `sysctl` 명령을 사용하여 커널 매개 변수를 영구적으로 설정하는 방법을 설명합니다.

전제 조건

- [커널 매개 변수의 개요](#)

- root 권한

단계

1. 모든 매개 변수를 나열하려면 다음을 수행합니다.

```
# sysctl -a
```

이 명령은 런타임시에 설정할 수있는 커널 매개 변수를 표시합니다.

2. 매개 변수를 영구적으로 설정하려면 다음 명령을 실행합니다.

```
# sysctl -w <TUNABLE_CLASS>.<PARAMETER>=<TARGET_VALUE> >> /etc/sysctl.conf
```

샘플 명령은 가변 값을 변경하여 `/etc/sysctl.conf` 파일에 씁니다. 그러면 커널 매개 변수의 기본값을 덮어 씁니다. 변경은 재부팅없이 즉시 영구적으로 반영됩니다.

참고

커널 매개 변수를 영구적으로 변경하려면 `/etc/sysctl.d/` 디렉토리의 설정 파일에 수동으로 변경하십시오.

관련 정보

- `sysctl` 자세한 내용은 `man` 페이지의 `sysctl(8)`도시 `sysctl.conf (5)`를 참조하십시오.
- 커널 매개 변수에 영구적 인 변경을위한 `/etc/sysctl.d/` 디렉토리에 설정 파일을 사용하는 것에 대한 정보는 [/etc/sysctl.d/ 설정 파일에 커널 매개 변수 조정](#) 섹션을 참조하십시오.

5.2.3. `/etc/sysctl.d/` 설정 파일에 커널 매개 변수 조정

다음 절차에서는 `/etc/sysctl.d/` 디렉토리의 설정 파일을 수동으로 수정하여 커널 매개 변수를 영구적으로 설정하는 방법을 설명합니다.

전제 조건

- 커널 매개 변수의 개요

- root 권한

단계

1. /etc/sysctl.d/ 새로운 설정 파일을 만듭니다.

```
# vim /etc/sysctl.d/<some_file.conf>
```

2. 다음과 같이 한 줄마다 커널 매개 변수를 포함합니다.

```
<TUNABLE_CLASS>.〈PARAMETER〉=<TARGET_VALUE>
<TUNABLE_CLASS>.〈PARAMETER〉=<TARGET_VALUE>
```

3. 설정을 저장합니다.

4. 시스템을 다시 시작하여 변경 사항을 적용합니다.

- 또한 다시 시작하지 않고 변경 사항을 적용하려면 다음을 수행합니다.

```
# sysctl -p /etc/sysctl.d/<some_file.conf>
```

이 명령은 이전에 만든 설정 파일에서 값을 읽을 수 있습니다.

관련 정보

- sysctl 자세한 내용은 man 페이지의 sysctl(8)을 참조하십시오.

- /etc/sysctl.d/자세한 내용은 man 페이지의 sysctl.d(5)을 참조하십시오.

5.2.4 /proc/sys/에 커널 매개 변수의 임시 설정

다음 단계에서는 가상 파일 시스템 /proc/sys/ 디렉토리의 파일을 사용하여 커널 매개 변수를 일시적으로 설정하는 방법을 설명합니다.

전제 조건

- [커널 매개 변수의 개요](#)
- root 권한

단계

1. 설정 커널 매개 변수를 식별합니다.

```
# ls -l /proc/sys/<TUNABLE_CLASS>/
```

명령에서 반환 된 쓰기 가능한 파일은 커널 설정에 사용할 수 있습니다. 읽기 전용 권한을 가진 사용자는 현재 설정에 대한 피드백을 제공합니다.

2. 커널 매개 변수에 대상의 값을 할당합니다.

```
# echo <TARGET_VALUE> > /proc/sys/<TUNABLE_CLASS>/<PARAMETER>
```

이 명령은 시스템이 다시 시작하면 설정 변경이 사라집니다.

3. 필요에 따라 새로 구성된 설정 한 커널 매개 변수의 값을 확인합니다.

```
# cat /proc/sys/<TUNABLE_CLASS>/<PARAMETER>
```

관련 정보

커널 매개 변수를 영구적으로 변경하려면 [sysctl 명령](#) 을 사용하거나 [/etc/sysctl.d/디렉토리](#)의 설정 파일에 수동으로 변경합니다.

5.3 가상화 환경에서 커널 패닉의 매개 변수 비활성화

Red Hat Enterprise Linux 8 (RHEL 8) 가상화 환경을 설정하는 경우, 가상화 환경은 시스템 장애를 필요로하지 않는 잘못된 소프트 잠금(soft lockup)을 발생시킬 가능성이 있기 때문에 softlockup_panic 및 nmi_watchdog 커널 매개 변수를 활성화하지 않아야합니다.

다음 섹션에서는 이 조언의 이유를 다음과 같이 요약하여 설명합니다.

- 소프트 잠금(soft lockup)의 원인
- 소프트 잠금(soft lockup)시 시스템의 동작을 제어하는 커널 매개 변수의 설명
- 가상화 환경에서 소프트 잠금(soft lockup)이 어떻게 발생하는지에 대한 설명

5.3.1. 잠금 기능이란?

잠금 기능은 일반적으로 작업이 일정하지 않고 **CPU**가 커널 영역에서 실행하는 경우에 버그에 의해 발생하는 상황입니다. 또한이 작업은 다른 작업이 그 특정 **CPU**에서 실행하는 것을 허용하지 않습니다. 이렇게하면 경고가 시스템 콘솔을 통해 사용자에게 표시됩니다. 이 문제는 소프트웨어 잠금 발생 (soft lockup firing)라고도 합니다.

관련 정보

잠금 기능에 관련된 기술적인 이유 로그 메시지의 예 및 기타 자세한 내용은 기술 자료 문서 "[CPU 잠금 기능](#)"을 참조하십시오.

5.3.2. 커널 패닉을 제어하는 매개 변수

소프트 롤이 탐지되면 시스템의 동작을 제어하는 다음의 커널 매개 변수를 설정할 수 있습니다.

softlockup_panic

잠금 기능이 검출 된 때 커널 패닉을 발생시키는 여부를 제어합니다.

타입	값	효과
정수(Integer)	0	소프트 잠금시 커널이 패닉 상태에 빠지지 않음
정수(Integer)	1	소프트 잠금에 대한 커널 패닉

RHEL 8에서는 이 값의 기본값은 0입니다.

패닉을 발생시키기 위해 시스템에서 처음으로 하드 록업을 감지해야합니다. 검색은 `nmi_watchdog` 매개 변수로 제어됩니다.

nmi_watchdog

잠금 감지 메커니즘 (`watchdogs`)가 활성화 여부를 제어합니다. 이 매개 변수는 정수형입니다.

값	효과
0	잠금 감지기 비활성화
1	잠금 감지기 활성화

하드 록업 검출기(**hard lockup detector**)는 각 CPU에 인터럽트에 응답하는 기능을 모니터링합니다.

watchdog_thresh

워치 독 시간 타이머 `hrtimer`, NMI 이벤트 및 소프트/하드 잠금 임계 값의 빈도를 제어합니다.

기본 임계 값(Default threshold)	소프트웨어 잠금 임계 값(Soft lockup threshold)
10 초	$2 * \text{watchdog_thresh}$

이 매개 변수를 “0”으로 설정하면 잠금 감지를 비활성화합니다.

관련 정보

- `nmi_watchdog` 그리고 `softlockup_panic` 대한 자세한 내용은 ["Softlockup detector and hardlockup detector"](#)를 참조하십시오.
- `watchdog_thresh` 자세한 내용은 ["Kernel sysctl"](#)를 참조하십시오.

5.3.3 가상화 환경에서 잘못된 소프트웨어 잠금

["잠금 기능은"](#)에서 설명되는 물리적 호스트에서 소프트웨어 잠금의 발생은 일반적으로 커널 또는 하드웨어의 버그를 보여줍니다. 가상화 환경의 게스트 운영 체제에서 같은 현상이 발생하면 잘못된 경고가 표시 될 수 있습니다.

호스트에 많은 작업 부하가 발생하거나 메모리와 같은 특정 리소스에 대한 높은 경합이 발생하면 일반적으로 가짜 소프트 락업이 발생합니다. 호스트가 게스트 **CPU**를 20초 이상 예약 할 수 있기 때문입니다. 그런 다음 게스트 **CPU**가 호스트에서 다시 실행되도록 예약하면 시간

초과가 발생하여 예정된 타이머를 트리거합니다. 타이머에는 워치 독 `hrtimer`도 포함되어 있어 게스트 **CPU**의 소프트 락업을 보고 할 수 있습니다.

가상화 된 환경에서의 소프트 락업이 허위 일 수 있으므로 게스트 **CPU**에서 소프트 락업이 보고 될 때 시스템 패닉을 유발하는 커널 매개 변수를 사용하지 않아야 합니다.

중요

게스트의 소프트 잠금을 이해하려면 호스트가 게스트를 작업으로 예약(**schedule**) 한 다음 게스트가 자체 작업을 예약(**schedule**)해야 합니다.

관련 정보

- 소프트 록업의 정의와 그 기능에 관련된 기술은 "[잠금 기능과](#)" 을 참조하십시오.
- RHEL 8 가상화 환경의 구성 요소와 그 상호 작용은 "[RHEL 8 가상 머신 구성 요소 및 그 상호 작용](#)" 을 참조하십시오.

5.4 데이터베이스 서버의 커널 매개 변수 조정

특정 데이터베이스 응용 프로그램의 성능에 영향을 미칠 수 있는 커널 매개 변수 세트에는 여러 가지가 있습니다. 다음 섹션에서는 데이터베이스 서버 및 데이터베이스의 효율적인 운영을 보장하기 위해 구성하는 커널 매개 변수를 설명합니다.

5.4.1 데이터베이스 서버의 개요

데이터베이스 서버는 일정량의 주 메모리와 데이터베이스 (**DB**) 응용 프로그램이 설치된 하드웨어 장치입니다. 이 **DB** 애플리케이션은 캐시 된 데이터를 일반적으로 작고 비싼 주 메모리에서 **DB** 파일 (**database**)로 쓰는 수단으로 서비스를 제공합니다. 이러한 서비스는 네트워크의 여러 클라이언트에 제공됩니다. 머신의 메인 메모리 및 스토리지가 허용하는 만큼 **DB** 서버가 존재할 수 있습니다.

Red Hat Enterprise Linux 8 다음 데이터베이스 응용 프로그램을 제공합니다.

- Maria DB 10.3
- MySQL 8.0

- PostgreSQL 10
- PostgreSQL 9.6

5.4.2. 데이터베이스 애플리케이션의 성능에 영향을 미치는 매개 변수

다음 커널 매개 변수는 데이터베이스 응용 프로그램의 성능에 영향을 줍니다.

fs.aio-max-nr

시스템이 서버에서 처리 할 수 있는 최대 비동기 I/O 작업 수를 정의합니다.

참고

`fs.aio-max-nr` 매개 변수를 높이면 `aio` 한도를 늘리는 것 외에는 추가 변경이 없습니다.

fs.file-max

시스템이 모든 인스턴스에서 지원하는 최대 파일 핸들 수 (임시 파일 이름 또는 파일을 열기 위해 할당 된 ID)를 정의합니다.

커널은 응용 프로그램이 파일 핸들을 요청할 때마다 파일 핸들을 동적으로 할당합니다. 그러나 커널은 이러한 파일 핸들이 응용 프로그램에 의해 해제 될 때 해제하지 않습니다. 커널은 대신 이러한 파일 핸들을 재활용합니다. 이는 현재 사용되는 파일 핸들 수가 적더라도 시간이 지남에 따라 할당 된 파일 핸들의 총 수가 증가 함을 의미합니다.

kernel.shmall

시스템 전체에서 사용할 수 있는 총 공유 메모리 페이지 수를 정의합니다. 전체 주 메모리를 사용하려면 `kernel.shmall` 매개 변수 값이 \leq 총 주 메모리 크기여야 합니다.

kernel.shmmmax

Linux 프로세스가 가상 주소 공간에 할당 할 수 있는 단일 공유 메모리 세그먼트의 최대 크기 (바이트)를 정의합니다.

kernel.shmmnni

데이터베이스 서버가 처리 할 수 있는 공유 메모리 세그먼트의 최대 개수를 정의합니다.

net.ipv4.ip_local_port_range

특정 포트 번호없이 데이터베이스 서버에 연결하는 프로그램에 시스템이 사용할 수 있는 포트 범위를 정의합니다.

net.core.rmem_default

TCP (Transmission Control Protocol)를 통해 기본 수신 소켓 메모리를 정의합니다.

net.core.rmem_max

TCP (Transmission Control Protocol)에 의한 최대 수신 소켓 메모리를 정의합니다.

net.core.wmem_default

TCP (Transmission Control Protocol)에 의한 기본 전송 소켓 메모리를 정의합니다.

net.core.wmem_max

TCP (Transmission Control Protocol)에 의한 최대 전송 소켓 메모리를 정의합니다.

vm.dirty_bytes / vm.dirty_ratio

더티 데이터를 생성하는 프로세스가 `write()` 함수에서 시작하는 더티 가능한 메모리 비율 (바이트 단위)에서 임계 값을 정의합니다.

참고

한 번에 지정할 수 있는 것은, `vm.dirty_bytes` 또는 `vm.dirty_ratio`의 **하나**입니다.

vm.dirty_background_bytes / vm.dirty_background_ratio

커널이 더티 데이터를 하드 디스크에 활성화 쓰려고하는 더티 가능한 메모리의 비율 (바이트 단위)에서 임계 값을 정의합니다.

참고

한 번에 지정할 수 있는 것은, `vm.dirty_background_bytes` 또는 `vm.dirty_background_ratio`의 **하나**입니다.

vm.dirty_writeback_centisecs

하드 디스크에 더티 데이터를 기록하는 커널 스레드의 시작을 정기적으로 간격을 정의합니다.

이 커널 매개 변수는 100 분의 1 초 단위로 측정됩니다.

vm.dirty_expire_centisecs

더티 데이터가 하드 디스크에 기록 될 정도로 오래 될 때까지의 시간을 정의합니다.

이 커널 매개 변수는 100 분의 1 초 단위로 측정됩니다.

관련 정보

- 더티 데이터의 라이트 백 그 기능 및 관련 커널 매개 변수에 대한 설명은 "[더티 페이지의 라이트 백은 어떻게 작동합니까?](#)"를 참조하십시오.

제 6 장 커널 로깅 사용

로그 파일은 시스템(커널, 서비스 및 실행중인 응용 프로그램 등)에 대한 메시지가 포함된 파일입니다. Red Hat Enterprise Linux에서 로깅 시스템은 내장 **syslog** 프로토콜을 기반으로합니다. 다양한 유ти리티가 시스템을 사용하여 이벤트를 기록하고 로그 파일에 정리합니다. 이 파일은 운영 체제의 감사 및 문제 해결에 도움이됩니다.

6.1 커널 링 버퍼(kernel ring buffer)란?

콘솔은 시스템이 시작되는 동안 시스템 부팅시에 대한 중요한 정보를 다수 제공합니다. 먼저 출력된 메시지가 손실되지 않도록 커널 링 버퍼라는 것이 사용되고 있습니다. 이 버퍼는 커널 코드의 `printf()` 함수에 의해 생성된 부트 메시지 등 모든 메시지를 저장합니다. 다음은 커널 링 버퍼에서 메시지 `syslog` 서비스 등의 영구 저장소의 로그 파일에 로드되어 저장됩니다.

상기 버퍼는 고정된 크기의 순환 데이터 구조이며, 커널에 하드 코딩되어 있습니다. 사용자는 `dmesg` 명령 또는 `/var/log/boot.1og` 파일 통해 커널 링 버퍼에 저장되어있는 데이터를 볼 수 있습니다. 링 버퍼가 가득 차면 새 데이터는 이전 데이터를 덮어 씁니다.

관련 정보

- `syslog` 자세한 내용은 `man` 페이지의 `syslog(2)`을 참조하십시오.
- `dmesg`로 시작 로그 메시지를 확인하거나 제어하는 방법은 `man` 페이지의 `dmesg(1)`을 참조하십시오.

6.2 로그 수준 및 커널 로깅에 `printf`의 역할

커널이보고하는 각 메시지에는 메시지의 중요도를 정의하는 관련 로그 수준이 있습니다. 커널 링 버퍼는 "[커널 링 버퍼란?](#)"에 설명 된대로 모든 로그 레벨의 커널 메시지를 수집합니다. 버퍼에서 콘솔에 출력되는 메시지를 정의하는 것은 `kernel.printf` 매개 변수입니다.

로그 수준 값은 다음 순서로 분류됩니다.

0 – 커널 비상(emergency). 시스템을 사용할 수 없습니다.

1 – 커널 경고(alert). 즉시 조치를 취해야합니다.

2 – 커널의 상태가 중요합니다.

3 – 일반 커널 오류 조건.

4 – 일반적인 커널 경고 조건.

5 – 정상이지만 중요한 상태의 커널 통지.

6 – 커널 정보 메시지.

7 – 커널 디버그 수준 메시지.

기본적 `kernel.printk` 으로 RHEL 8 에는 다음 네 가지 값이 포함되어 있습니다.

```
# sysctl kernel.printk  
kernel.printk = 7      4      1      7
```

네 가지 값은 다음을 정의합니다.

1. 값(**value**). 콘솔 로그 수준. 콘솔에 인쇄되는 메시지의 최저 우선 순위를 정의합니다.
2. 값(**value**). 명시적인 로그 수준이 첨부되지 않은 메시지의 기본 로그 수준.
3. 값(**value**). 콘솔 로그 수준에 가능한 최저 수준의 로그 수준 구성 설정합니다.
4. 값(**value**). 부팅시 콘솔 로그 수준의 기본값을 설정합니다.

위의 각 값은 오류 메시지를 처리하기 위한 다른 규칙을 정의합니다.

참고

같은 특정 커널 명령 줄 매개 변수, `quiet` 또는 `debug` 기본 변경 `kernel.printk` 값을.

관련 정보

- `kernel.printk` 및 로그 수준에 대한 자세한 내용은 `syslog(2)` 매뉴얼 페이지를 참조하십시오.

제 7 장 kdump 설치 및 구성

7.1. kdump 란?

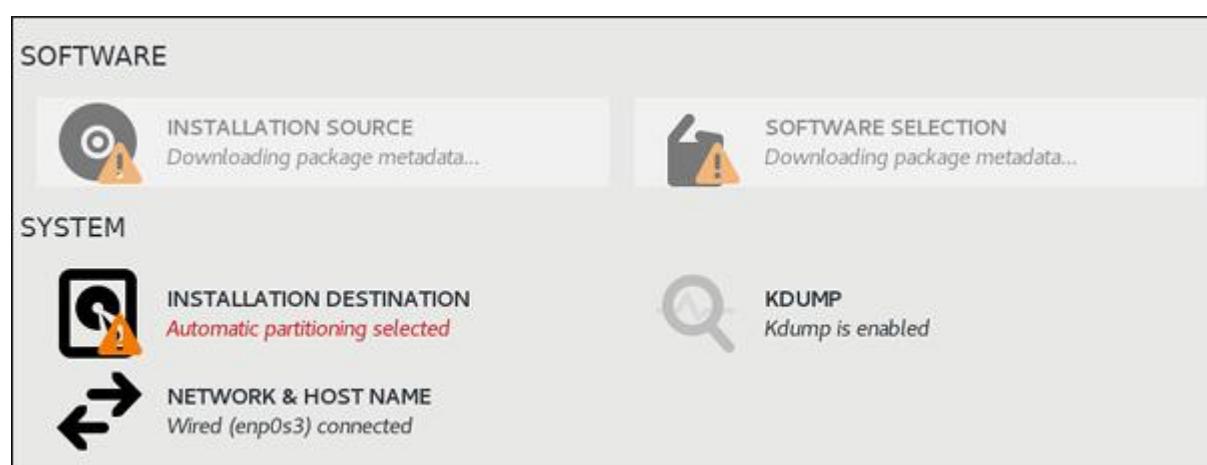
kdump 크래시 덤프 메커니즘을 제공하는 서비스입니다. 이 서비스를 사용하면 나중에 분석 할 수 있도록 시스템 메모리의 내용을 저장할 수 있습니다. 시스템 호출을 kdump 사용하여 재부팅하지 않고 kexec 두 번째 커널 (캡쳐 커널)로 부팅합니다. 그런 다음 충돌 한 커널 메모리의 내용 (crash dump 또는 vmcore) 을 캡처하여 저장합니다. 두 번째 커널은 시스템 메모리의 예약 된 부분에 있습니다.

중요

커널 크래시 덤프는 시스템 장애(심각한 버그)가 발생했을 때 사용할 수 있는 유일한 정보 일 수 있습니다. 따라서 미션 크리티컬 환경에서 kdump 를 확실하게 실행하는 것이 중요합니다. Red Hat은 시스템 관리자가 일반 커널 업데이트주기 kexec-tools 를 정기적으로 업데이트하여 테스트하는 것이 좋습니다. 이것은 새로운 커널 기능이 구현되어있는 경우에 특히 중요합니다.

7.2. kdump 설치

대부분의 경우 kdump 서비스는 새로운 Red Hat Enterprise Linux 설치시 기본적으로 설치되고 활성화됩니다. **아나콘다** 설치 프로그램에 대한 화면을 제공하는 kdump 그래픽 또는 텍스트 인터페이스를 사용하여 대화식 설치를 수행 할 때 구성. 설치 프로그램 화면은 제목이 Kdump 있고 기본 Installation Summary 화면에서 사용할 수 있으며 제한된 구성 만 허용합니다. kdump 활성화 여부와 예약된 메모리 양만 선택할 수 있습니다 .



사용자 정의 킥 스타트 설치와 같은 일부 설치 옵션 `kdump`은 기본적으로 설치 또는 활성화되지 않습니다. 시스템에 해당하는 경우 아래 절차에 따라 설치하십시오 `kdump`.

전제 조건

- 활성 Red Hat Enterprise Linux 서브스크립션
- 시스템의 CPU 아키텍처 용 **kexec-tools** 패키지가 있는 저장소
- `kdump` 요구 사항을 충족

단계

1. 다음 명령을 실행하여 `kdump`이 시스템에 설치되어 있는지 확인합니다.

```
$ rpm -q kexec-tools
```

이 패키지가 설치되어있는 경우는 다음을 출력합니다.

```
kexec-tools-2.0.17-11.el8.x86_64
```

이 패키지가 설치되어 있지 않은 경우는 다음을 출력합니다

```
package kexec-tools is not installed
```

2. `kdump` 및 필요한 패키지를 설치합니다.

```
# yum install kexec-tools
```

중요

Red Hat Enterprise Linux 7.4 (kernel-3.10.0-693.el7) 이후부터 `kdump`는 Intel IOMMU 드라이버가 지원되하고 있습니다. 이전 버전의 Red Hat Enterprise Linux 7.3 (ernel-3.10.0-514 [.XYZ].el7) 또는 그 이전 버전에서는 Intel IOMMU 지원을 비활성화하는 것이 좋습니다되어 있습니다. 해제하지 않으면 `kdump` 커널이 응답하지 않을 가능성이 높습니다.

관련 정보

- kdump 메모리 요구 사항의 자세한 내용은 "[kdump 메모리 요구 사항](#)"을 참조하십시오.

7.3 명령 줄에서 kdump 설정

7.3.1 메모리 사용량 설정

kdump 기능에 예약 된 메모리는 시스템 시작시에 항상 예약되어 있습니다. 메모리의 양은 시스템의 **GRUB (Grand Unified Bootloader) 2** 설정에서 지정합니다. 다음 단계에서는 명령 줄에서 kdump 예약 한 메모리의 설정 방법을 설명합니다.

전제 조건

- kdump [요구 사항](#)을 충족.

단계

1. **root** 권한으로 /etc/default/grub 파일을 편집합니다.

2. **crashkernel=** 옵션을 원하는 값으로 설정합니다.

예를 들어, 128 MB 의 메모리를 예약하려면 다음을 사용합니다.

```
crashkernel=128M
```

또는 설치되어있는 메모리 양에 따라 예약 메모리 크기를 변수로 설정할 수 있습니다. 변수에 대한 메모리 예약 구문입니다

`crashkernel=<range1>:<size1>,<range2>:<size2>`. 다음에 예를 나타냅니다.

```
crashkernel=512M-2G:64M,2G-:128M
```

위의 예에서는 시스템 메모리의 합계가 512 MB 이상 2 GB 미만의 경우 64 MB의 메모리를 예약합니다. 메모리 크기가 2 GB를 초과하는 경우에는 128 MB가 kdump용으로 예약됩니다.

- 예약 된 메모리의 오프셋.

`crashkernel` 예약 매우 빠르기 때문에 특정 고정 오프셋에서 메모리를 예약해야 하는 시스템도 있습니다. 또한 특별한 용도에 일부 공간을 예약하고자 하는 시스템도 있습니다. 오프셋이 설정되면 예약 메모리는 거기에서 시작됩니다. 예약 메모리를 오프셋하려면 다음 구문을 사용합니다.

```
crashkernel=128M@16M
```

위의 예는 `kdump`이 16 MB(물리 주소 0x01000000)에서 시작 128 MB의 메모리를 예약하고 있음을 보여줍니다. 오프셋 매개 변수가 0으로 설정되어 있는 경우, 또는 완전히 생략되는 경우 `kdump` 자동으로 예약 메모리를 오프셋합니다. 위와 같이 변수 메모리 예약을 설정하는 경우에도 이 구문을 사용합니다. 이 경우 오프셋은 항상 마지막에 지정됩니다 (예 : `crashkernel=512M-2G:64M,2G-:128 _M@16 _M`).

3. 다음 명령을 사용하여 GRUB2 설정 파일을 업데이트합니다.

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

참고

`kdump`에 대한 메모리를 구성하는 다른 방법은 `grub2-editenv`를 사용하여 `crashkernel = <some_value>` 매개 변수를 `kernelopts` 변수에 추가하여 모든 부팅 항목을 업데이트하는 것입니다. 또는 `grubby` 유틸리티를 사용하여 한 항목의 커널 명령 줄 매개 변수를 업데이트 할 수 있습니다.

관련 정보

- `crashkernel` 옵션은 여러 가지 방법으로 정의 할 수 있습니다. "["kdump 메모리 요구 사항"](#)"에 설명 된 지침에 따라 `auto` 값을 지정하면 시스템의 총 메모리에 따라 예약 메모리의 자동 설정이 가능하게됩니다.
- 부팅 항목, `kernelopts`, `grub2-editenv` 및 `grubby` 작업 방법은 [4 장 커널 명령 줄 매개 변수 설정](#)을 참조하십시오.

7.3.2 kdump 대상 설정

커널 충돌이 캡처되는 코어 덤프가 로컬 파일 시스템의 파일로 저장하거나 장치에 직접 쓰거나 NFS(Network File System) 또는 SSH(Secure Shell) 프로토콜을 사용하여 네트워크를 통해 전송할 수 있습니다. 이 옵션은 동시에 설정할 수 없습니다. 기본 동작은 **vmcore** 파일을 로컬 파일 시스템의 /var/pkcs/디렉토리에 저장됩니다.

전제 조건

- kdump [요구 사항](#) 을 충족.

단계

코어 덤프를 저장할 로컬 디렉토리를 변경하려면 root에서 다음과 같이 /etc/kdump.conf 설정 파일을 편집합니다.

1. #path /var/crash 를 앞에 해시 기호 ("#")를 제거합니다.
2. 값을 대상 디렉토리 경로로 바꿉니다. 다음에 예를 나타냅니다.

```
path /usr/local/cores
```

중요

Red Hat Enterprise Linux 8에서 경로 지정문을 사용하여 kdump 대상으로 정의 된 디렉토리가 kdump systemd 서비스 시작시에 존재해야합니다. 이것이 존재하지 않는 경우 서비스가 실패합니다. 이 동작은 서비스 시작시 디렉토리가 없으면 자동으로 생성 된 **Red hat Enterprise Linux**의 이전 릴리스와는 다릅니다.

다른 파티션에 파일을 쓰려면 root 권한으로 다음과 같이 /etc/kdump.conf 설정 파일을 편집합니다.

1. 필요에 따라 #ext4 를 앞에 해시 기호 ("#")를 제거합니다.
 - 장치 이름 (#ext4 /dev/vg/lv_kdump 행)
 - 파일 시스템 레이블 (#ext4 LABEL=/boot 행)
 - UUID (#ext4 UUID=03138356-5e61-4ab3-b58e-27507ac41937 행)

2. 파일 시스템 유형과 장치 이름, 레이블 UUID를 원하는 값으로 변경합니다. 다음에 예를 나타냅니다.

```
ext4 UUID = 03138356-5e61-4ab3-b58e-27507ac41937
```

중요

LABEL= UUID=를 사용하여 저장 장치를 지정하는 것이 좋습니다. /dev/sda3 등의 디스크 장치 이름은 재부팅시 일관성은 보장되지 않습니다.

중요

IBM Z 하드웨어의 **Direct Access Storage Device (DASD)**로 덤프하는 경우 처리하기 전에 /etc/dasd.conf 덤프 장치가 제대로 지정되어 있어야합니다.

덤프 장치에 직접 쓰려면 다음을 수행합니다.

1. #raw /dev/vg/lv_kdump 줄 앞에 해시 기호 ("#")를 제거합니다.
2. 값을 대상 장치 이름으로 바꿉니다. 다음에 예를 나타냅니다.

```
raw /dev/sdb1
```

NFS 프로토콜을 사용하여 원격 컴퓨터에 덤프를 저장하려면 다음을 수행합니다:

1. #nfs my.server.com:/export/tmp 줄 앞에 해시 기호 ("#")를 제거합니다.
2. 값을 올바른 호스트 이름 및 디렉토리 경로로 바꿉니다. 다음에 예를 나타냅니다.

```
nfs penguin.example.com:/export/cores
```

SSH 프로토콜을 사용하여 원격 컴퓨터에 덤프를 저장하려면 다음을 수행합니다.

1. #ssh user@my.server.com 줄 앞에 해시 기호 ("#")를 제거합니다.
2. 값을 올바른 사용자 이름과 호스트 이름으로 대체합니다.

3. SSH 키 설정에 포함됩니다.

- `#sshkey /root/.ssh/kdump_id_rsa` 줄 앞에 해시 기호 ("#")를 제거합니다.
- 값을 덤프 대상 서버에 올바른 키의 위치로 변경합니다. 다음에 예를 나타냅니다.

```
ssh john@penguin.example.com  
sshkey /root/.ssh/mykey
```

관련 정보

- 지원하는 덤프 대상과 호환되지 않는 덤프 대상 유형별 목록은 "[지원 kdump 덤프 대상](#)"을 참조하십시오.
- SSH 서버를 구성하고 키 기반 인증을 구성하는 방법은 [Red Hat Enterprise Linux 의 기본 시스템 설정 구성](#)을 참조하십시오.

7.3.3 코어 컬렉터 설정

`kdump`는 `core collector`로 지정된 프로그램을 사용하여 `vmcore`를 캡처합니다. 현재 완벽하게 지원하는 `core collector`는 `makedumpfile` 유ти리티뿐입니다. 여기에는 몇 가지 설정 가능한 옵션이 컬렉션 프로세스에 영향을 줍니다. 예를 들어, 수집 된 데이터의 범위와 생성 된 `vmcore`를 압축할지 여부 등입니다.

`core collector`를 활성화하고 설정하려면 다음 단계를 수행합니다.

전제 조건

- `kdump` [요구 사항](#)을 충족.

단계

1. `root` 권한으로, `/etc/kdump.conf` 설정 파일을 편집하여 `#core_collector makedumpfile -l --message-level 1 -d 31` 줄 앞에 해시 기호 ("#")를 제거합니다.
2. `-c` 매개 변수를 추가합니다. 다음에 예를 나타냅니다.

```
core_collector makedumpfile -c
```

위의 명령을 사용하면 덤프 파일의 압축을 사용할 수 있습니다.

- d 값매개 변수를 추가합니다. 다음에 예를 나타냅니다.

```
core_collector makedumpfile -d 17 -c
```

위의 명령은 덤프에서 “0” 및 빈 페이지를 모두 삭제합니다. 이 값은 비트 마스크를 나타냅니다. 여기에서는 각 비트가 특정 유형의 메모리 페이지와 관련된 해당 유형의 페이지가 수집되는지 여부를 결정합니다. 각 비트의 설명은 ["지원되는 kdump 필터 수준"](#)을 참조하십시오.

관련 정보

- 사용 가능한 옵션의 전체 목록은 `makedumpfile(8)man` 페이지를 참조하십시오.

7.3.4. kdump 기본 장애 응답 설정

기본적으로 kdump 이 ["kdump 대상 설정"](#)에 지정된 위치에서 `vmcore` 덤프 파일을 생성하지 못하면 시스템이 재부팅되고 프로세스에서 덤프가 손실됩니다. 이 동작을 변경하려면 아래 절차를 따르십시오.

전제 조건

- kdump [요구 사항](#) 을 충족.

단계

- `root` 권한으로, `/etc/kdump.conf` 설정 파일 `#default shell`의 시작 부분에 해시 기호 ("#")를 제거합니다.
- ["지원하는 기본 장애 응답"](#)의 설명대로 값을 원하는 액션으로 바꿉니다. 다음에 예를 나타냅니다.

```
default poweroff
```

7.3.5. kdump 서비스의 활성화 및 비활성화

시스템 시작시 kdump 서비스를 시작하려면 다음을 수행합니다.

전제 조건

- kdump [요구 사항](#) 을 충족하고 있다.
- 모든 [설정](#) 이 필요에 따라 설정되어 있다.

단계

1. kdump 서비스를 활성화하려면 다음 명령을 실행합니다.

```
# systemctl enable kdump.service
```

이로 인해 `multi-user.target` 서비스가 활성화됩니다.

2. 현재 세션에서 서비스를 시작하려면 다음 명령을 사용합니다.

```
# systemctl start kdump.service
```

3. kdump 서비스를 중지하려면 다음 명령을 실행합니다.

```
# systemctl stop kdump.service
```

4. kdump 서비스를 해제하려면 다음 명령을 실행합니다.

```
# systemctl disable kdump.service
```

관련 정보

- `systemd` 세부 일반적인 서비스 설정은 Red Hat Enterprise Linux의 "기본 시스템 설정"을 참조하십시오.

7.4 Web 콘솔에서 kdump 설정

다음 섹션에서는 Red Hat Enterprise Linux 웹콘솔을 통해 `kdump`를 설정하고 테스트하는 방법에 대한 개요를 제공합니다. 웹콘솔은 Red Hat Enterprise Linux 8의 기본 설치에 포함되어 있으며, 시스템 시작시 `kdump` 서비스를 활성화하거나 비활성화 할 수 있습니다. 또한, `kdump` 예약 메모리를 설정하거나 비 압축 또는 압축 형식으로 `vmcore`의 저장 위치를 선택 할 수 있습니다.

전제 조건

- 자세한 내용은 "Red Hat Enterprise Linux web console"을 참조하십시오.

7.4.1. Web 콘솔에서 kdump 메모리 사용량과 대상 위치를 설정

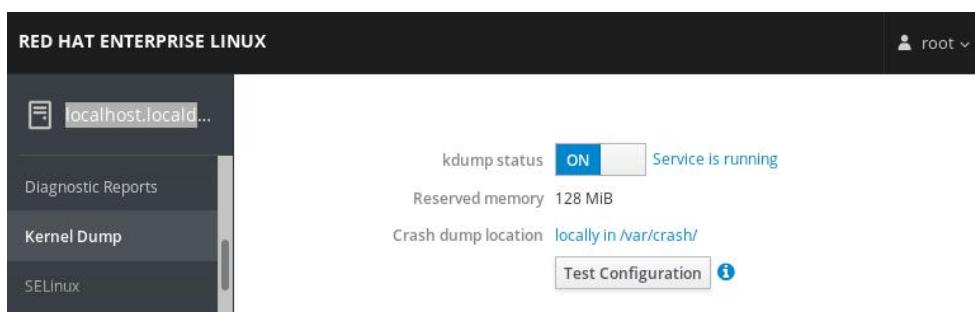
다음 단계는 Red Hat Enterprise Linux 웹콘솔 인터페이스의 Kernel Dump 탭을 사용하여 `kdump` 커널 용으로 예약 된 메모리 용량을 설정하는 방법을 설명합니다. 이 단계에서는 `vmcore` 덤프 파일의 대상 위치를 지정하는 방법과 설정을 테스트하는 방법을 설명합니다.

전제 조건

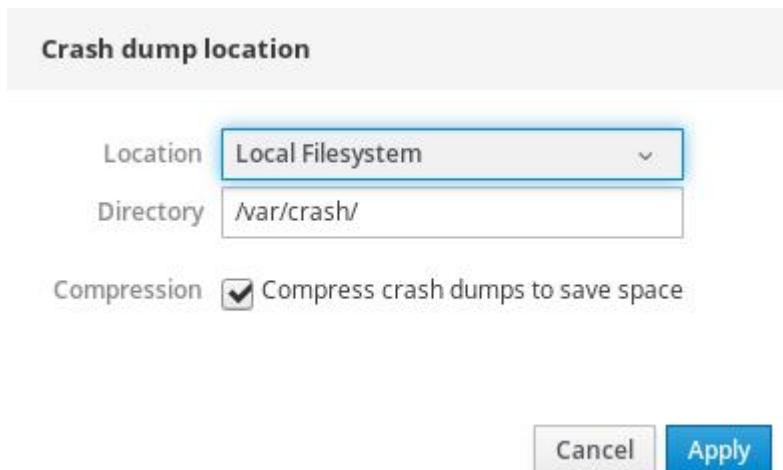
- Web 콘솔 작업 개요

단계

1. Kernel Dump 탭을 열고 `kdump` 서비스를 시작합니다.
2. 명령 줄에서 `kdump`의 메모리 사용량을 설정합니다.
3. Crash dump location 옵션 옆에있는 링크를 클릭합니다.



4. 드롭 다운 메뉴에서 **Local Filesystems** 옵션을 선택하고 덤프를 저장할 디렉토리를 지정합니다.



또는 드롭 다운에서 **Remote over SSH** 옵션을 선택하고 **SSH** 프로토콜을 사용하여 **vmcore** 을 원격 시스템으로 전송합니다.

Server, **ssh key**, **Directory** 의 각 필드에 원격 컴퓨터의 주소, **ssh** 키의 위치 및 대상 디렉토리를 입력합니다.

또는 드롭 다운에서 **Remote over NFS** 옵션을 선택하고 **Mount** 필드에 정보를 입력하여 **NFS** 프로토콜을 사용하여 **vmcore** 을 원격 컴퓨터에 보낼 수 있습니다.

참고

Compression 체크 박스에 체크 표시를하고 **vmcore** 파일의 크기를 줄일 수 있습니다.

5. 커널 크래시를 이용하여 설정을 테스트 하십시오.

The screenshot shows a status interface for kdump. It displays "kdump status" as "ON" with a green progress bar labeled "Service is running". Below this, it shows "Reserved memory" as "128 MiB" and "Crash dump location" as "locally in /var/crash/". There is a "Test Configuration" button with an info icon.

경고

이 단계에서는 커널의 실행을 중단하고 시스템 크래시나 데이터 손실이 발생 할 수 있습니다.

관련 정보

- 현재 지원되는 **kdump** 대상의 전체 목록은 "지원되는 **kdump** 덤프 대상"을 참조하십시오.
- SSH 서버를 구성하고 키 기반 인증을 구성하는 방법은 Red Hat Enterprise Linux 의 "[기본 시스템 설정 구성](#)"을 참조하십시오.

7.5. 지원하는 **kdump** 설정 및 덤프 대상

7.5.1. **kdump** 메모리 요구 사항

위해서는 **kdump** 커널 크래시 덤프를 캡처하고 추가 분석을 위해 저장 할 수 있도록, 시스템 메모리의 일부가 영구적으로 캡처 커널에 예약해야합니다. 예약되면 시스템 메모리의이 부분을 기본 커널에서 사용할 수 없습니다.

메모리 요구 사항은 특정 시스템 매개 변수에 따라 다릅니다. 주요 요인 중 하나는 시스템의 하드웨어 아키텍처입니다. 정확한 시스템 아키텍처 (예 :**x86_64** 라고도하는 Intel 64 및 AMD64)를 찾아 표준 출력으로 인쇄하려면 다음 명령을 사용하십시오.

```
$ uname -m
```

아래 표에는에 대한 메모리 크기를 자동으로 예약하기위한 최소 메모리 요구 사항 목록이 포함되어 있습니다 **kdump**. 크기는 시스템 아키텍처 및 사용 가능한 총 실제 메모리에 따라 달라집니다.

표 7.1. **kdump에 필요한 예약 메모리의 최소량**

아키텍처	사용 가능한 메모리	최소 예약 메모리
	1GB ~ 64GB	160MB 의 RAM.
AMD64 및 Intel 64 (x86_64)	64GB ~ 1TB	256MB 의 RAM

아키텍처	사용 가능한 메모리	최소 예약 메모리
	1TB 이상	512MB 의 RAM
64 비트 ARM 아키텍처 (arm64)	2GB 이상	512MB 의 RAM
	2GB ~ 4GB	384MB 의 RAM.
	4GB ~ 16GB	512MB 의 RAM
	16GB ~ 64GB	1GB 의 RAM.
	64GB ~ 128GB	2GB 의 RAM.
	128GB 이상	4GB 의 RAM.
	4GB ~ 64GB	160MB 의 RAM.
IBM Power Systems (ppc64le)	64GB ~ 1TB	256MB 의 RAM
	1TB 이상	512MB 의 RAM
IBM Z (s390x)	4GB ~ 64GB	160MB 의 RAM.
	64GB ~ 1TB	256MB 의 RAM

많은 시스템에서 `kdump` 필요한 메모리 양을 예측하여 자동으로 예약 할 수 있습니다. 이 동작은 기본적으로 활성화되어 있지만 [사용 가능한 메모리 총량](#)이 일정 이상인 시스템에서만 작동합니다. 이것은 시스템의 아키텍처에 따라 다릅니다.

중요

시스템 메모리 총량에 근거 예약 메모리의 자동 설정은 최선형 예측입니다. 실제로 필요한 메모리는 I/O 장치 등의 다른 요소에 따라 다를 수 있습니다. 메모리가 충분하지 않은 경우 커널 패닉이 발생했을 때 디버그 커널이 캡처 커널로 부팅 할 수 없게 될 가능성이 있습니다. 이 문제를 해결하려면 크래시 커널 메모리를 충분한 크기로 합니다.

관련 정보

- 명령 줄에서 메모리 설정을 변경하는 방법은 ["메모리 사용량 설정"](#)을 참조하십시오.
- Web 콘솔에서 예약 된 메모리의 양을 설정하는 방법은 ["Web 콘솔에서 kdump 메모리 사용량과 대상 위치를 설정"](#)을 참조하십시오.

- Red Hat Enterprise Linux 의 다양한 기술 기능과 제한은 [기술 기능과 제한 표](#) 를 참조하십시오.

7.5.2 메모리 자동 예약 최소 임계 값

일부 시스템에서는 부트 로더 설정 파일에서 `crashkernel=auto` 매개 변수를 사용하거나 그래픽 설정 유ти리티에서 옵션을 사용하여 `kdump` 용 메모리를 자동으로 할당 할 수 있습니다. 그러나 자동 예약 기능하려면 전체 메모리의 특정 양의 메모리를 사용할 수 있어야합니다. 필요한 용량은 시스템의 아키텍처에 따라 다릅니다.

다음 표는 자동 메모리 할당 임계 값의 목록입니다. 시스템의 메모리가 테이블의 지정보다 낮은 경우, 메모리는 [수동으로 예약](#) 해야합니다.

표 7.2 자동 메모리 예약 필요한 최소 메모리 크기

아키텍처	필요한 메모리
AMD64 및 Intel 64 (<code>x86_64</code>)	2GB
IBM Power Systems (<code>ppc64le</code>)	2GB
IBM Z (<code>s390x</code>)	4GB

관련 정보

- 명령 줄에서 이러한 설정을 수동으로 변경하는 방법은 ["메모리 사용량 설정"](#) 을 참조하십시오.
- Web 콘솔을 통해 예약 메모리의 양을 수동으로 변경하는 방법은 ["Web 콘솔에서 kdump 메모리 사용량과 대상 위치를 설정"](#) 을 참조하십시오.

7.5.3. 지원하는 kdump 덤프 대상

커널 크래시가 캡처되면 `vmcore` 덤프 파일은 장치에 직접 쓰거나 로컬 파일시스템에서 파일로 저장되거나 네트워크를 통해 전송됩니다. 다음 표는 현재 지원 덤프 대상 또는 `kdump` 명시적으로 지원하지 않는 덤프 대상의 전체 목록을 보여줍니다.

표 7.3 지원하는 kdump 덤프 대상

타입	지원하는 덤프 대상	지원하지 않는 덤프 대상
Raw 장치	로컬에서 첨부 된 모든 raw 디스크와 파티션	
로컬 파일 시스템	직접 연결된 디스크 드라이브, 하드웨어 RAID 논리 드라이브 LVM 장치, mdraid 어레이의 ext2, ext 3, ext4, 및 xfs 파일 시스템.	auto 유형 (자동 파일 시스템 감지) 을 포함하여 표에서 명시적으로 자명되지 않은 로컬 파일시스템.
원격 디렉토리	IPv4 를 통한 NFS 또는 SSH 프로토콜을 사용하여 엑세스 한 원격 디렉토리.	NFS 프로토콜을 사용하여 액세스 한 rootfs 파일 시스템의 원격 디렉토리.
iSCSI 하드웨어 및 소프트웨어 이니시에이터를 통해 프로토콜을 사용하여 액세스 한 원격 디렉토리 .	be2iscsi 하드웨어에서 iSCSI 프로토콜을 사용하여 액세스하는 원격 디렉토리.	다중 경로 기반의 스토리지
		IPv6 기반으로 액세스하는 원격 디렉토리
		SMB 또는 CIFS 를 사용하여 액세스하는 원격 디렉토리.
		FCoE(Fibre Channel over Ethernet) 프로토콜을 사용하여 액세스하는 원격 디렉토리.
		무선 네트워크 인터페이스를 통해 액세스하는 원격 디렉토리

관련 정보

- 명령 줄에서 대상 유형을 설정하는 방법은 "[kdump 대상 설정](#)" 을 참조하십시오.
- Web 콘솔에서 대상을 설정하는 방법은 "[Web 콘솔에서 kdump 메모리 사용량과 대상 위치를 설정](#)" 을 참조하십시오.

7.5.4. 지원하는 kdump 필터 수준

덤프 파일의 크기를 축소하기 위해 kdump 는 makedumpfile 코어 컬렉터를 사용하여 데이터를 압축하고 필요에 따라 불필요한 정보를 생략합니다. 다음 표는 makedumpfile 유틸리티에서 현재 지원하는 필터 수준의 전체 목록을 보여줍니다.

표 7.4 지원하는 필터 수준

옵션	설명
1	제로 페이지(zero pages)
2	캐시 페이지(Cache pages)
4	캐시 전용(Cache private)
8	사용자 페이지(User pages)
16	빈 페이지(Free pages)

참고

`makedumpfile` 명령은 투명한 대형 페이지 및 `hugefbfs` 페이지의 삭제에 대응하고 있습니다. 이러한 유형의 **hugepages User Page** 모두를 생각하는 -8 수준을 사용하여 삭제합니다.

관련 정보

- 명령 줄에서 코어 콜렉터를 설정하는 방법은 "[코어 콜렉터 설정](#)"을 참조하십시오.

7.5.5. 지원하는 기본 장애 응답

기본적으로 `kdump`이 코어 덤프를 작성할 수 없는 경우, 운영 체제가 다시 시작합니다. 그러나 코어 덤프를 주 대상으로 저장할 수 없는 경우 `kdump` 가 다른 작업을 수행하도록 설정할 수 있습니다. 다음 표는 현재 지원되는 모든 기본 작업 목록입니다.

표 7.5 지원하는 기본 동작

옵션	설명
<code>dump_to_rootfs</code>	<code>root</code> 파일 시스템에 코어 덤프의 저장을 시도합니다. 네트워크 덤프 대상과 병용하는 경우에 특히 유용한 옵션입니다. 네트워크 덤프 대상에 연결할 수 없는 경우 로컬 코어 덤프를 저장하도록 <code>kdump</code> 설정합니다. 시스템은 나중에 다시 시작합니다.
<code>reboot</code>	시스템을 다시 시작합니다. 코어 덤프는 손실됩니다.
<code>halt</code>	시스템을 중지합니다. 코어 덤프는 손실됩니다.

옵션	설명
poweroff	시스템의 전원을 끕니다. 코어 덤프는 손실됩니다.
shell	initramfs 내에서 shell 세션을 실행하여 사용자가 수동으로 코어 덤프를 기록 할 수 있도록합니다.

관련 정보

- 명령 줄에서 기본 실패 응답을 설정하는 방법에 대한 자세한 내용은 "["kdump 기본 장애 응답 설정"](#)"을 참조하십시오.

7.5.6. kdump 크기 추정(Estimating)

kdump 환경 계획 및 구축시에는 덤프 파일에 필요한 공간을 파악하고 작성해야합니다.

`makedumpfile --mem-usage` 명령은 제외 가능한 페이지에 대한 유용한 보고서를 생성하고 할당 할 덤프 레벨을 결정하는 데 사용할 수 있습니다. 시스템이 대표적인 부하에서 이 명령을 실행합니다. 그렇지 않으면, `makedumpfile --mem-usage`이 프로덕션 환경에서 예상되는 값보다 작은 값을 반환합니다.

```
[root@hostname ~]# makedumpfile --mem-usage /proc/kcore
```

TYPE	PAGES	EXCLUDABLE	DESCRIPTION
ZERO	501635	yes	Pages filled with zero
CACHE	51657	yes	Cache pages
CACHE_PRIVATE	5442	yes	Cache pages + private
USER	16301	yes	User process pages
FREE	77738211	yes	Free pages
KERN_DATA	1333192	no	Dumpable kernel data

중요

`makedumpfile --mem-usage` 명령은 **페이지** 단위를 출력합니다. 즉, 커널 페이지 크기에 대해 사용 중인 메모리의 크기를 계산해야합니다. 기본적으로 Red Hat Enterprise Linux 커널은 AMD64 및 Intel 64 아키텍처 4 KB 크기의 페이지를 사용하여 IBM POWER 아키텍처에는 64 KB 크기의 페이지를 사용합니다.

7.6. kdump 설정 테스트

다음 단계에서는 커널 덤프 프로세스가 실행 시스템이 프로덕션 환경에 들어가기 전에 유효한지 테스트하는 방법을 설명합니다.

경고

다음 명령은 커널 크래시를 발생시킵니다. 다음 단계에 따라 활성 프로덕션 시스템에서 부주의하게 사용하지 마십시오.

단계

1. `kdump` 를 [사용](#) 하여 시스템을 다시 시작합니다.
2. `kdump` 이 동작하고 있는지 확인합니다.

```
~]# systemctl is-active kdump  
Active
```

3. `Linux` 커널을 강제로 크래시 시킵니다.

```
echo 1 > /proc/sys/kernel/sysrq  
echo c > /proc/sysrq-trigger
```

경고

위의 명령은 커널이 크래시됩니다. 리부팅이 필요합니다.

다시 시작하면 `/etc/kdump.conf` 에서 [지정](#) 하는 장소 (기본적으로 `/var/crash/`)에 `address-YYYY-MM-DD-HH:MM:SS/vmcore` 파일이 생성됩니다.

참고

설정의 유효성을 확인하는 것 외에도, 이 액션을 이용하여 대표적인 부하가 실행 중일 때 크래시 덤프가 완료하는 데 걸리는 시간을 기록 할 수 있습니다.

7.7 코어 덤프 분석

시스템 충돌의 원인을 판별하기 위해 **충돌** 디버거를 사용하면 GNU 디버거 (GDB)와 매우 유사한 대화식 프롬프트를 제공 할 수 있습니다. 이 유ти리티를 사용하면 실행중인 Linux 시스템 뿐만 아니라 `kdump`, `netdump`, `diskdump` 또는 `xendump`에 의해 생성된 코어 덤프를 대화식으로 분석할 수 있습니다. 또는 [Kdump Helper](#) 또는 [Kernel Oops Analyzer](#)를 사용할 수 있습니다.

7.7.1. crash 유ти리티 설치

다음 단계에서는 `crash` 분석 도구의 방법을 설명합니다.

단계

- 관련 `baseos` 저장소와 `appstream` 저장소를 사용합니다.

```
# subscription-manager repos --enable baseos repository  
# subscription-manager repos --enable appstream repository
```

- `crash` 패키지를 설치합니다.

```
# yum install crash
```

- `kernel-debuginfo` 패키지를 설치합니다.

```
# yum install kernel-debuginfo
```

패키지는 실행중인 커널에 대응하고 덤프 분석에 필요한 데이터를 제공합니다.

관련 정보

- `subscription-manager` 유ти리티를 사용하여 저장소를 조작하는 방법은 ["기본 시스템 설정 구성"](#)을 참조하십시오.

7.7.2. crash 유ти리티 종료

다음 단계에서는 시스템 충돌의 원인을 분석하기 위해 `crash` 유ти리티를 시작하는 방법을 설명합니다.

전제 조건

현재 실행중인 커널을 확인합니다 (4.18.0-5.el8.x86_64 등).

단계

1. `crash` 유ти리티를 시작하려면 두 가지 필요한 매개 변수를 명령에 전달해야합니다.
 - debug-info (압축 해제 된 vmlinuz 이미지) (특정 `kernel-debuginfo` 패키지에 포함된 `/usr/lib/debug/lib/modules/4.18.0-5.el8.x86_64/vmlinuz` 등)
 - 실제 vmcore 파일 `./var/crash/127.0.0.1-2018-10-06-14:05:33/vmcore`

그 결과 `crash` 명령 다음과 같습니다.

```
# crash /usr/lib/debug/lib/modules/4.18.0-5.el8.x86_64/vmlinuz \
var/crash/127.0.0.1-2018-10-06-14:05:33/vmcore
```

`kdump`에서 취득한 것과 같은 `<kernel>` 버전을 사용합니다.

예 7.1 `crash` 유ти리티 실행

다음의 예는 4.18.0-5.el8.x86_64 커널을 사용하여 2018 년 10 월 6 일 (14:05 PM)에 생성 된 코어 덤프 분석을 보여줍니다.

```
...
WARNING: kernel relocated [202MB]: patching 90160 gdb minimal_symbol values

        KERNEL: /usr/lib/debug/lib/modules/4.18.0-5.el8.x86_64/vmlinuz
        DUMPFILE: /var/crash/127.0.0.1-2018-10-06-14:05:33/vmcore  [PARTIAL DUMP]
          CPUS: 2
          DATE: Sat Oct  6 14:05:16 2018
         UPTIME: 01:03:57
    LOAD AVERAGE: 0.00, 0.00, 0.00
      TASKS: 586
    NODENAME: localhost.localdomain
    RELEASE: 4.18.0-5.el8.x86_64
   VERSION: #1 SMP Wed Aug 29 11:51:55 UTC 2018
    MACHINE: x86_64 (2904 Mhz)
    MEMORY: 2.9 GB
    PANIC: "sysrq: SysRq : Trigger a crash"
      PID: 10635
    COMMAND: "bash"
    TASK: fffff8d6c84271800 [THREAD_INFO: fffff8d6c84271800]
```

```
CPU: 1  
STATE: TASK_RUNNING (SYSRQ)  
  
crash>
```

2. 대화 형 프롬프트를 종료하고 `crash` 종료하려면 `crash` 또는 `q`를 사용합니다.

예 7.2 `crash` 유ти리티 종료

```
crash> exit  
~]#
```

참고

`crash` 명령은 라이브 시스템을 디버깅하는 강력한 도구로 사용할 수 있습니다. 그러나 시스템이 손상되지 않도록주의하십시오.

7.7.3. `crash` 유ти리티의 메시지 버퍼 역추적 및 기타 표시기의 표시

다음 단계는 `crash` 유ти리티를 사용하여 커널 메시지 버퍼 다시 추적 프로세스 상태, 가상 메모리 정보 오픈 파일 등 다양한 표시기를 표시하는 방법을 설명합니다.

메시지 버퍼보기

커널 메시지 버퍼를 보려면 다음 예에 표시된대로 대화식 프롬프트에서 `log` 명령을 실행 합니다.

예 7.3 커널 메시지 버퍼보기

```
crash> log  
... several lines omitted ...  
EIP: 0060:[<c068124f>] EFLAGS: 00010096 CPU: 2  
EIP is at sysrq_handle_crash+0xf/0x20  
EAX: 00000063 EBX: 00000063 ECX: c09e1c8c EDX: 00000000  
ESI: c0a09ca0 EDI: 00000286 EBP: 00000000 ESP: ef4dbf24  
DS: 007b ES: 007b FS: 00d8 GS: 00e0 SS: 0068  
Process bash (pid: 5591, ti=ef4da000 task=f196d560 task.ti=ef4da000)  
Stack:  
c068146b c0960891 c0968653 00000003 00000000 00000002 efade5c0 c06814d0  
<0> ffffffb c068150f b7776000 f2600c40 c0569ec4 ef4dbf9c 00000002 b7776000  
<0> efade5c0 00000002 b7776000 c0569e60 c051de50 ef4dbf9c f196d560 ef4dbfb4  
Call Trace:
```

```

[<c068146b>] ? __handle_sysrq+0xfb/0x160
[<c06814d0>] ? write_sysrq_trigger+0x0/0x50
[<c068150f>] ? write_sysrq_trigger+0x3f/0x50
[<c0569ec4>] ? proc_reg_write+0x64/0xa0
[<c0569e60>] ? proc_reg_write+0x0/0xa0
[<c051de50>] ? vfs_write+0xa0/0x190
[<c051e8d1>] ? sys_write+0x41/0x70
[<c0409adc>] ? syscall_call+0x7/0xb
Code: a0 c0 01 0f b6 41 03 19 d2 f7 d2 83 e2 03 83 e0 cf c1 e2 04 09 d0 88 41 03 f3 c3 90 c7 05
c8 1b 9e c0 01 00 00 00 0f ae f8 89 f6 <c6> 05 00 00 00 00 01 c3 89 f6 8d bc 27 00 00 00 00 00 8d 50
d0 83
EIP: [<c068124f>] sysrq_handle_crash+0xf/0x20 SS:ESP 0068:ef4dbf24
CR2: 0000000000000000

```

명령 사용에 대한 자세한 내용 `help log` 을 실행합니다.

참고

커널 메시지 버퍼는 시스템 장애에 대한 가장 중요한 정보가 포함되어 있습니다. 따라서 이것은 항상 먼저 `vmcore-dmesg.txt` 파일로 덤프됩니다. 이것은 예를 들어, 대상 위치에 공간이 없기 때문에 `vmcore` 전체 파일 가져오기 시도가 실패 할 때 유용합니다. 기본적으로 `vmcore-dmesg.txt` 는 `/var/pkcs/directory/`입니다.

7.7.3.1. 역추적보기(backtrace)

커널 스택 트레이스을 표시하려면 `bt` 명령을 사용합니다.

예 7.4 커널 스택 추적 표시

```

crash> bt
PID: 5591  TASK: f196d560  CPU: 2  COMMAND: "bash"
#0 [ef4dbdcc] crash_kexec at c0494922
#1 [ef4dbe20] oops_end at c080e402
#2 [ef4dbe34] no_context at c043089d
#3 [ef4dbe58] bad_area at c0430b26
#4 [ef4dbe6c] do_page_fault at c080fb9b
#5 [ef4dbe4] error_code (via page_fault) at c080d809
    EAX: 00000063  EBX: 00000063  ECX: c09e1c8c  EDX: 00000000  EBP: 00000000
    DS: 007b      ESI: c0a09ca0  ES: 007b      EDI: 00000286  GS: 00e0
    CS: 0060      EIP: c068124f  ERR: ffffffff  EFLAGS: 00010096
#6 [ef4dbf18] sysrq_handle_crash at c068124f
#7 [ef4dbf24] __handle_sysrq at c0681469
#8 [ef4dbf48] write_sysrq_trigger at c068150a
#9 [ef4dbf54] proc_reg_write at c0569ec2
#10 [ef4dbf74] vfs_write at c051de4e
#11 [ef4dbf94] sys_write at c051e8cc
#12 [ef4dbfb0] system_call at c0409ad5
    EAX: ffffffd8  EBX: 00000001  ECX: b7776000  EDX: 00000002
    DS: 007b      ESI: 00000002  ES: 007b      EDI: b7776000
    SS: 007b      ESP: bfcb2088  EBP: bfcb20b4  GS: 0033
    CS: 0073      EIP: 00edc416  ERR: 00000004  EFLAGS: 00000246

```

`bt <pid>`를 입력하여 특정 프로세스의 백 트레이스를 표시하거나 `help bt` 실행하여 `bt` 사용에 대한 자세한 내용을 표시합니다.

7.7.3.2 프로세스의 상태 표시

시스템의 프로세스의 상태를 표시하려면 `ps` 명령을 사용합니다.

예 7.5 시스템의 프로세스의 상태 표시

```
crash> ps
  PID  PPID CPU TASK ST %MEM VSZ RSS COMM
> 0 0 0 c09dc560 RU 0.0 0 0 [swapper]
> 0 0 1 f7072030 RU 0.0 0 0 [swapper]
0 0 2 f70a3a90 RU 0.0 0 0 [swapper]
> 0 0 3 f70ac560 RU 0.0 0 0 [swapper]
1 0 1 f705ba90 IN 0.0 2828 1424 init
... several lines omitted ...
5566 1 1 f2592560 IN 0.0 12876 784 auditd
5567 1 2 ef427560 IN 0.0 12876 784 auditd
5587 5132 0 f196d030 IN 0.0 11064 3184 sshd
> 5591 5587 2 f196d560 RU 0.0 5084 1648 bash
```

`ps <pid>`를 사용하여 단일 프로세스의 상태를 표시합니다. `ps` 자세한 사용 방법은 `help ps`를 사용합니다.

7.7.3.3 가상 메모리 정보보기

기본 가상 메모리 정보를 표시하려면 대화식 프롬프트에서 `vm` 명령을 입력합니다.

예 7.6 현재 컨텍스트의 가상 메모리 정보보기

```
crash> vm
PID: 5591 TASK: f196d560 CPU: 2 COMMAND: "bash"
 MM PGD RSS TOTAL_VM
f19b5900 ef9c6000 1648k 5084k
 VMA START END FLAGS FILE
f1bb0310 242000 260000 8000875 /lib/ld-2.12.so
f26af0b8 260000 261000 8100871 /lib/ld-2.12.so
efbc275c 261000 262000 8100873 /lib/ld-2.12.so
efbc2a18 268000 3ed000 8000075 /lib/libc-2.12.so
efbc23d8 3ed000 3ee000 8000070 /lib/libc-2.12.so
efbc2888 3ee000 3f0000 8100071 /lib/libc-2.12.so
efbc2cd4 3f0000 3f1000 8100073 /lib/libc-2.12.so
efbc243c 3f1000 3f4000 100073
efbc28ec 3f6000 3f9000 8000075 /lib/libdl-2.12.so
```

```
efbc2568 3f9000 3fa000 8100071 /lib/libdl-2.12.so
efbc2f2c 3fa000 3fb000 8100073 /lib/libdl-2.12.so
f26af888 7e6000 7fc000 8000075 /lib/libtinfo.so.5.7
f26aff2c 7fc000 7ff000 8100073 /lib/libtinfo.so.5.7
efbc211c d83000 d8f000 8000075 /lib/libnss_files-2.12.so
efbc2504 d8f000 d90000 8100071 /lib/libnss_files-2.12.so
efbc2950 d90000 d91000 8100073 /lib/libnss_files-2.12.so
f26afe00 edc000 edd000 4040075
f1bb0a18 8047000 8118000 8001875 /bin/bash
f1bb01e4 8118000 811d000 8101873 /bin/bash
f1bb0c70 811d000 8122000 100073
f26afae0 9fd9000 9ffa000 100073
... several lines omitted ...
```

`vm <pid>`를 실행하여 하나의 프로세스 정보를 표시하거나 `help vm` 실행하여 `vm` 사용 방법을 표시합니다.

7.7.3.4 오픈 파일보기

오픈 파일의 정보를 표시하려면 `files` 명령을 실행합니다.

예 7.7 현재 컨텍스트의 오픈 파일에 대한 정보 표시

```
crash> files
PID: 5591  TASK: f196d560  CPU: 2  COMMAND: "bash"
ROOT: /  CWD: /root
FD   FILE    DENTRY   INODE   TYPE   PATH
 0  f734f640  eedc2c6c  eecd6048  CHR    /pts/0
 1  efade5c0  eee14090  f00431d4  REG    /proc/sysrq-trigger
 2  f734f640  eedc2c6c  eecd6048  CHR    /pts/0
10  f734f640  eedc2c6c  eecd6048  CHR    /pts/0
255 f734f640  eedc2c6c  eecd6048  CHR    /pts/0
```

`files <pid>`를 실행하여 선택한 하나의 프로세스에 의해 열린 파일을 표시하거나 `help files` 실행하여 `files` 사용 방법을 표시합니다.

7.7.4. Kernel Oops Analyzer 사용

Kernel Oops Analyzer는 지식 기반의 알려진 문제 `oops` 메시지를 비교하여 크래시 덤프를 분석하는 도구입니다.

전제 조건

- [Red Hat Labs](#)의 절차에 따라 `oops` 메시지를 보호하고 Kernel Oops Analyzer에 입력합니다.

단계

1. [Kernel Oops Analyzer](#) 링크를 따라 도구에 액세스합니다.
2. 찾아보기 버튼을 눌러 oops 메시지를 찾아보십시오.

Data Input

File Input

Text Input

Choose and upload the **kernel oops log** generated from a vmcore.

Browse... No file selected.

Maximum file size for uploaded kernel oops log is 10 MB.

Detect

3. DETECT 버튼을 클릭하여 makedumpfile에서 정보를 기반으로 기존의 솔루션과 oops 메시지를 비교합니다.

관련 정보

- **kdump.conf (5)** - 사용 가능한 옵션의 전체 문서를 포함한 /etc/kdump.conf 설정 파일의 man 페이지들.
- **zipl.conf (5)** : /etc/zipl.conf 설정 파일의 man 페이지입니다.
- **zipl (8)** - IBM System z 용 zipl 부트 로더 유ти리티 man 페이지.
- **makedumpfile (8)** - makedumpfile 코어 수집가 man 페이지.
- **kexec (8)** - kexec man 페이지.
- **crash (8)** - crash 유ти리티 man 페이지.
- `/usr/share/doc/kexec-tools/kexec-kdump-howto.txt` - kdump 와 kexec 설치 및 사용의 개요.
- kexec로 kdump 설정 자세한 내용은 "[Red Hat 기술 자료 문서](#)"를 참조하십시오.
- 지원되는 kdump 대상의 자세한 내용은 "[Red Hat 기술 자료 문서](#)"를 참조하십시오.

제 8 장 커널 라이브 패치로 패치

Red Hat Enterprise Linux 커널의 라이브 패치 솔루션을 사용하여 시스템을 다시 시작하거나 프로세스를 다시 시작하지 않고 실행중인 커널에 패치 할 수 있습니다.

이 솔루션으로 시스템 관리자는 다음을 수행 할 수 있습니다.

- 중요한 보안 패치를 커널에 즉시 적용 할 수 있습니다.
- 장시간 실행되는 작업의 완료되거나 사용자 로그 오프 또는 예정된 가동 중지 시간을 기다릴 필요가 없다.
- 시스템 가동 시간을 더 많이 확보하고 보안과 안정성을 잃지 않습니다.

커널 라이브 패치 솔루션을 사용하여 모든 크리티컬 또는 중요한 CVE 가 해결되는 것은 아닙니다. 우리의 목표는 보안 관련 패치에 필요한 재부팅을 줄이고 완전히 제거하지는 않는 것입니다. 라이브 패치 범위의 자세한 내용은 "[RHEL 7 라이브 커널 패치 \(kpatch\)을 지원하고 있습니까?](#)"를 참조하십시오.

경고

커널의 라이브 패치와 다른 커널 서브 구성 요소 사이에 일부 비 호환성이 존재합니다. 커널 라이브 패치를 사용하기 전에 ["kpatch 제한"](#)을 주의 깊게 확인하십시오.

8.1. kpatch 제한

kpatch 기능은 일반 커널 업그레이드 메커니즘은 없습니다. 시스템을 신속하게 다시 시작 할 수없는 경우 등 간단한 보안 및 버그 수정 업데이트를 적용하는 경우에 사용합니다.

패치 읽거나 읽은 후에는 SystemTap 도구 또는 kprobe 도구를 사용하지 마십시오. 이러한 프로브가 제거 될 때까지 패치를 적용 할 수 없게 될 가능성이 있습니다.

8.2 타사 라이브 패치 지원(third-party live patching)

kpatch 유틸리티는 Red Hat 저장소 배포 한 RPM 모듈을 포함한 Red Hat 이 지원하는 유일한 커널 라이브 패치 유틸리티입니다. Red Hat 은 Red Hat 제공하지 않은 라이브 커널 패치는 지원하지 않습니다.

타사 라이브 패치에서 발생하는 문제에 대한 지원이 필요한 경우, 근본 원인 확인이 필요한 모든 조사의 시작 부분에 라이브 패치 벤더와 함께 사례를 열 것을 권장합니다. 이를 통해 공급업체가 허용한 경우 소스 코드를 공급할 수 있으며, 공급업체가 **Red Hat** 지원팀으로 조사를 확대하기 전에 근본 원인 결정에 도움을 제공할 수 있습니다.

타사 라이브 패치를 실행하는 시스템의 경우, **Red Hat**은 **Red Hat**이 함께하고 지원하는 소프트웨어의 복제를 요구할 권리가 있습니다. 이것이 가능하지 않은 경우, **Red Hat**은 동일한 문제가 발생하는지 여부를 확인하기 위해 라이브 패치를 적용하지 않고 사용자의 테스트 환경에서 유사한 시스템 및 워크로드 배포를 구합니다.

타사 소프트웨어 지원 정책의 자세한 내용은 "["Red Hat 글로벌 지원 서비스는 타사의 소프트웨어, 드라이버, 그리고 인증되지 않은 하드웨어 및 하이퍼 바이저 또는 게스트 운영 체제에 대해 어떤 지원을 제공합니다. 있습니까?"](#)"를 참조하십시오.

8.3 커널 라이브 패치에 액세스

라이브 커널 패치 기능은 RPM 패키지로 제공되는 커널 모듈 (`kmod`)로 구현됩니다.

모든 고객은 일반 채널에서 제공되는 커널 라이브 패치에 액세스 할 수 있습니다. 그러나 연장 지원 서비스에 가입하지 않은 고객은 다음의 마이너 릴리즈를 사용할 수 있게되면 현재의 마이너 릴리즈에 대한 새로운 패치에 대한 액세스를 잃게됩니다. 예를 들어, 스탠다드 서브스크립션을 구매 한 고객은 RHEL 8.3 커널이 나올 때까지 RHEL 8.2 커널 라이브 패치만 할 수 있습니다.

8.4 커널 라이브 패치의 구성 요소

커널 라이브 패치의 구성 요소는 다음과 같습니다.

커널 패치 모듈(kernel patch module)

- 커널 라이브 패치의 전달 메커니즘
- 패치가 적용된 커널에 구축 한 커널 모듈.
- 패치 모듈은 커널 서브 시스템에 등록하고 대체 원래 기능 정보를 제공합니다. 또한 대체되는 기능과 일치하는 포인터도 포함됩니다. 커널 패치 모듈은 RPM 으로 제공됩니다.

- 명명 규칙 `kpatch_<kernel version>_<kpatch version>_<kpatch release>`입니다. 이름 "kernel version"부분의 마침표와 하이픈을 밑줄로 대체합니다.

kpatch 유ти리티

패치 모듈을 관리하기위한 명령줄 유ти리티.

kpatch 서비스

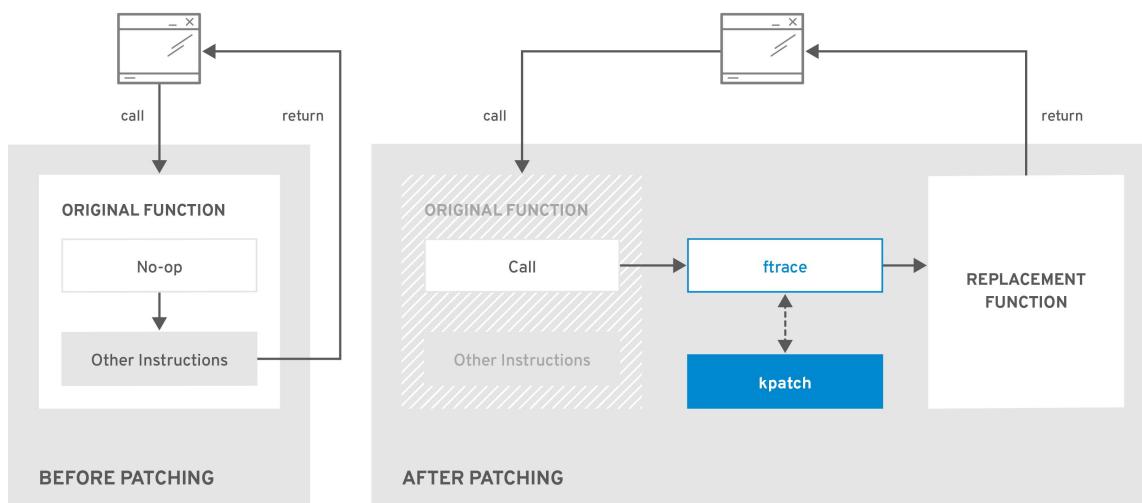
`multiuser.target`에서 필요한 `systemd` 서비스입니다. 이 대상은 시스템 부팅시 커널 패치를 로드합니다.

8.5 커널 라이브 패치의 작동 방식

`kpatch` 커널 패치 솔루션은 `livepatch` 커널 서브 시스템을 사용하여 기존 기능의 대상을 새로운 기능으로 변경합니다. 라이브 커널 패치가 시스템에 적용되면 다음과 같은 일이 발생합니다.

1. 커널 패치 모듈은 `/var/lib/kpatch/` 디렉토리에 복사하고 다음 부팅시에 `systemd`를 통해 커널에 다시 적용으로 등록됩니다.
2. 실행중인 커널에 `kpatch` 모듈이로드 된 새로운 코드의 메모리 위치를 지정하는 포인터를 사용하여 새로운 기능이 `ftrace` 메커니즘에 등록됩니다.
3. 패치되었을 기능에 커널이 접근하면 `ftrace` 메커니즘에 리디렉션됩니다. 이렇게하면 원래의 기능을 방지하고 패치 버전의 기능에 커널을 재 지정합니다.

그림 8.1 커널 라이브 패치의 동작 방식



8.6 커널 라이브 패치 활성화

커널 패치 모듈은 패치되는 커널 버전에 따라 RPM 패키지로 제공됩니다. 각 RPM 패키지는 시간이 지남에 따라 누적 업데이트됩니다.

다음 섹션에서는 특정 커널에 대한 향후 모든 누적 패치 업데이트를 받는 방법을 설명합니다.

경고

Red Hat은 Red Hat이 지원하는 시스템에 적용된 타사 라이브 패치를 지원하지 않습니다.

8.6.1 라이브 패치 스트림에 등록

이 단계에서는 특정 라이브 패치 패키지를 설치하는 방법을 설명합니다. 이렇게하면 지정된 커널 라이브 패치 스트림에 가입하여 향후 커널에 대한 누적 라이브 패치 업데이트를 모두 받을 수 있습니다.

경고

라이브 패치는 누적되는 형태이므로 특정 커널에 배치되는 개별 패치를 선택할 수 없습니다.

전제 조건

- root 권한

단계

- 필요에 따라 커널 버전을 확인합니다.

```
# uname -r  
4.18.0-94.el8.x86_64
```

- 커널의 버전과 일치하는 라이브 패치 패키지를 검색합니다.

```
# yum search $(uname -r)
```

- 라이브 패치 패키지를 설치합니다.

```
# yum install "kpatch-patch = $(uname -r)"
```

위의 명령은 특정 커널에만 최신 누적 패치를 설치하고 적용합니다.

패키지 버전이 1-1 이상이면 라이브 패치 패키지에 패치 모듈이 포함되어 있습니다. 이 경우 라이브 패치 패키지를 설치할 때 커널에 패치가 자동으로 적용됩니다.

커널 패치 모듈은 향후 다시 시작할 때 `systemd` 시스템 및 서비스 관리자에 의해 로드된 `/var/lib/kpatch/디렉토리`에 설치됩니다.

참고

지정된 커널에 사용 가능한 라이브 패치가 없는 경우는 빈 라이브 패치 패키지가 설치됩니다. 빈 라이브 패치 패키지는 `kpatch_version-kpatch_release 0-0` (예 : `kpatch-patch-4_18_0-94-0-0.e18.x86_64.rpm`)가 포함되어 있습니다. 빈 **RPM** 설치하면 시스템이 지정된 커널에 대한 향후 모든 라이브 패치를 구독합니다.

4. 필요에 따라 커널 패치 있는지 확인합니다.

```
# kpatch list
Loaded patch modules:
kpatch_4_18_0_94_1_1 [enabled]

Installed patch modules:
kpatch_4_18_0_94_1_1 (4.18.0-94.e18.x86_64)
...
```

이 출력은 커널 패치 모듈이 커널에 로드되어 있는지를 보여줍니다. 즉, 커널은 현재 `kpatch-patch-4_18_0-94-1-1.e18.x86_64.rpm` 패키지의 최신 수정 사항이 적용되어 있습니다.

관련 정보

- `kpatch` 명령 줄 유틸리티에 대한 자세한 내용은 `man` 페이지의 `kpatch(1)`을 참조하십시오.
- Red Hat Enterprise Linux 8 소프트웨어 패키지의 자세한 내용은 "[기본 시스템 설정 구성](#)"의 관련 섹션을 참조하십시오.

8.7 커널 패치 모듈 업데이트

커널 패치 모듈이 제공되며 **RPM** 패키지를 통해 적용되어 있기 때문에, 누계 커널 패치 모듈 업데이트는 다른 **RPM** 패키지의 업데이트와 비슷합니다.

전제 조건

- **root** 권한
- "[라이브 패치 스트림에 등록](#)"에 설명 된대로 시스템이 라이브 패치 스트림에 등록되어 있습니다.

단계

1. 현재 커널의 새 누적 버전을 업데이트합니다.

```
# yum update "kpatch-patch = $ (uname -r)"
```

위의 명령은 현재 실행중인 커널에 사용 가능한 업데이트를 자동으로 설치하고 적용합니다. 여기에는 새로 출시된 누적 라이브 패치가 포함되어 있습니다.

- 또는 설치 한 모든 커널 패치 모듈을 업데이트합니다.

```
# yum update "kpatch-patch *"
```

참고

시스템이 동일한 커널로 재부팅되면 `kpatch.service` `systemd` 서비스는 커널을 자동으로 다시 패치됩니다.

관련 정보

- 소프트웨어 패키지의 업데이트에 대한 자세한 정보는 Red Hat Enterprise Linux 8에서 ["기본 시스템 설정을 구성"](#)을 참조하십시오.

8.8 커널 라이브 패치 비활성화

시스템 관리자가 Red Hat Enterprise Linux 커널 라이브 패치 솔루션과 관련하여 예상치 못한 부정적인 영향을 경험 한 경우, 메커니즘을 비활성화 할 수 있습니다. 다음 섹션에서는 라이브 패치 솔루션을 비활성화하는 방법에 대해 설명합니다.

중요

현재 **Red Hat**은 시스템의 재부팅없이 라이브 패치를 되돌릴 수는 방법은 지원하지 않습니다. 문의 사항이 있으시면 지원팀에 문의하십시오.

8.8.1 라이브 패치 패키지 제거

다음 단계는 라이브 패치 패키지를 제거하여 **Red Hat Enterprise Linux** 커널의 라이브 패치 솔루션을 해제하는 방법을 설명합니다.

전제 조건

- **root** 권한
- 라이브 패치 패키지가 설치되어 있다.

단계

1. 라이브 패치 패키지를 선택합니다.

```
# yum list installed | grep kpatch-patch
kpatch-patch-4_18_0-94.x86_64 1-1.el8 @@ commandline
...
```

위의 출력 예는 설치 한 라이브 패치 패키지를 나열합니다.

2. 라이브 패치 패키지를 제거합니다.

```
# yum remove kpatch-patch-4_18_0-94.x86_64
```

라이브 패치 패키지가 제거되면 커널은 다음 재부팅까지 패치되었을 남아 있지만 커널 패치 모듈은 디스크에서 삭제됩니다. 다음 재부팅 후이 일치하는 커널 패치가 적용되지 않습니다.

3. 시스템을 다시 시작합니다.

4. 라이브 패치 패키지가 삭제 된 것을 확인합니다.

```
# yum list installed | grep kpatch-patch
```

패키지가 성공적으로 제거 된 경우이 명령은 아무것도 출력되지 않습니다.

- 필요에 따라 커널의 라이브 패치 솔루션이 비활성화되어 있는지 확인합니다.

```
# kpatch list  
Loaded patch modules:
```

이 출력 예제에서는 현재로드 된 패치 모듈이 없기 때문에 커널에 패치가 적용되지 않고, 라이브 패치 솔루션이 활성화되지 않은 것으로 표시되어 있습니다.

관련 정보

- kpatch** 명령 줄 유틸리티에 대한 자세한 내용은 **man** 페이지의 **kpatch(1)**을 참조하십시오.
- 소프트웨어 패키지의 사용 방법은 ['기본 시스템 설정 구성'](#)의 관련 섹션을 참조하십시오.

8.8.2 커널 패치 모듈 제거

다음 단계에서는 Red Hat Enterprise Linux 커널 라이브 패치 솔루션이 후속 부팅 커널 패치 모듈을 적용하지 않도록하는 방법을 설명합니다.

전제 조건

- root** 권한
- 라이브 패치 패키지가 설치되어있다.
- 커널 패치 모듈이 설치된로드되어있다.

단계

- 커널 패치 모듈을 선택합니다.

```
# kpatch list  
Loaded patch modules:  
kpatch_4_18_0_94_1_1 [enabled]  
  
Installed patch modules:  
kpatch_4_18_0_94_1_1 (4.18.0-94.el8.x86_64)  
...
```

2. 선택한 커널 패치 모듈을 제거합니다.

```
# kpatch uninstall kpatch_4_18_0_94_1_1  
Uninstalling kpatch_4_18_0_94_1_1 (4.18.0-94.el8.x86_64)
```

- 제거 된 커널 모듈이 로드되어 있는지에 주의하십시오.

```
# kpatch list  
Loaded patch modules:  
kpatch_4_18_0_94_1_1 [enabled]  
  
Installed patch modules:  
<NO_RESULT>
```

선택한 모듈을 제거하면 다음에 다시 부팅 할 때까지 커널이 패치된 상태로 유지되지만 커널 패치 모듈은 디스크에서 제거됩니다.

3. 시스템을 다시 시작합니다.

4. 필요한 커널 패치 모듈이 제거되어 있는지 확인합니다.

```
# kpatch list  
Loaded patch modules:  
...
```

위의 예제 출력에는 로드되거나 설치된 커널 패치 모듈이 표시되지 않으므로 커널이 패치되지 않고 커널 라이브 패치 솔루션이 활성화되지 않습니다.

관련 정보

- `kpatch` 명령 줄 유틸리티에 대한 자세한 내용은 `man` 페이지의 `kpatch(1)`을 참조하십시오.

8.8.3. `kpatch.service` 비활성화

다음 단계에서는 **Red Hat Enterprise Linux** 커널 라이브 패치 솔루션이 후속 부팅 커널 패치 모듈을 전체적으로 적용되지 않도록 하는 방법을 설명합니다.

전제 조건

- root 권한
- 라이브 패치 패키지가 설치되어있음.
- 커널 패치 모듈이 설치되고 로드되어 있음.

단계

1. `kpatch.service` 가 활성화되어 있는지 확인합니다.

```
# systemctl is-enabled kpatch.service  
Enabled
```

2. `kpatch.service` 를 비활성화합니다.

```
# systemctl disable kpatch.service  
Removed /etc/systemd/system/multi-user.target.wants/kpatch.service.
```

```
# kpatch list  
Loaded patch modules:  
kpatch_4_18_0_94_1_1 [enabled]  
  
Installed patch modules:  
kpatch_4_18_0_94_1_1 (4.18.0-94.el8.x86_64)
```

3. 시스템을 다시 시작합니다.

4. 필요에 따라 `kpatch.service` 상태를 확인합니다.

```
# systemctl status kpatch.service  
● kpatch.service - "Apply kpatch kernel patches"  
  Loaded: loaded (/usr/lib/systemd/system/kpatch.service; disabled; vendor preset: disabled)  
  Active: inactive (dead)
```

출력 예는 `kpatch.service` 가 비활성화되어 실행되고 있지 않음을 나타냅니다. 따라서 커널 라이브 패치 솔루션이 활성화되지 않습니다.

5. 커널 패치 모듈이 언로드된 것을 확인합니다.

```
# kpatch list
```

```
Loaded patch modules:  
<NO_RESULT>  
  
Installed patch modules:  
kpatch_4_18_0_94_1_1 (4.18.0-94.el8.x86_64)
```

위의 출력 예는 커널 패치 모듈이 여전히 설치되어 있지만 커널이 패치되지 않았음을 보여줍니다.

관련 정보

- kpatch 명령 줄 유틸리티에 대한 자세한 내용은 man 페이지의 `kpatch(1)`을 참조하십시오.
- `systemd` 시스템 및 서비스 관리자, 장치 설정, 그 위치 및 `systemd` 장치 유형의 전체 목록의 자세한 내용은 "[기본 시스템 설정 구성](#)"의 관련 섹션을 참조하십시오.

제 9 장 응용 프로그램 제한 설정

시스템 관리자로서 컨트롤 그룹(control group) 커널 기능을 사용하여 프로세스의 하드웨어 리소스를 제한하거나 우선 순위를 지정하거나 격리하여 시스템의 애플리케이션이 안정적이고 메모리가 부족하지 않도록 하십시오.

9.1. 컨트롤 그룹(control group)이란?

컨트롤 그룹은 Linux 커널 기능으로 프로세스를 계층적으로 정렬된 그룹(cgroups)으로 구성 할수 있습니다. 계층 구조(control group tree)는 cgroups 가상 파일 시스템 구조를 제공하여 정의됩니다. 기본적으로 /sys/fs/cgroup 디렉토리에 마운트됩니다. 이것은 /sys/fs/cgroup에 하위 디렉토리를 만들고 제거하여 수동으로 수행됩니다. systemd 시스템 및 서비스 관리자를 사용할 수 있습니다.

자원 컨트롤러 (커널 구성 요소)은 이러한 프로세스의 시스템 자원 (CPU time, 메모리, 네트워크 대역폭 및 다양한 조합 등)을 제한, 우선 순위 지정 또는 할당하여 cgroup 내 프로세스의 동작을 변경합니다.

Cgroups 부가기능은 프로세스 통합으로 응용 프로그램과 사용자 사이의 하드웨어 자원의 분할을 가능하게합니다. 그 결과, 사용자 환경의 전반적인 효율성, 안정성 및 보안이 향상됩니다.

9.1.1. 컨트롤 그룹 버전 1(Control group version 1)

컨트롤 그룹 버전 1(cgroups-v1)은 리소스 별 컨트롤러 계층을 제공합니다. 즉, CPU, 메모리, I/O 등과 같은 각 리소스에는 자체 제어 그룹 계층이 있습니다. 하나의 컨트롤러가 각각의 리소스를 관리 할 때 다른 컨트롤러와 조정할 수 있는 방식으로 다른 컨트롤 그룹 계층을 결합 할 수 있습니다. 그러나 두 컨트롤러는 서로 다른 프로세스 계층에 속할 수 있으며, 적절한 조정을 허용하지 않습니다.

cgroups-v1 컨트롤러는 오랜 기간에 걸쳐 개발되었었으므로 제어 파일의 동작과 이름이 일정하지 않습니다.

이 절은 Devconf.cz 2019 프레젠테이션을 기반으로했습니다. [1]

9.1.2. 컨트롤 그룹 2(Control group version 2)

계층 구조 유연성에서 비롯된 컨트롤러 조정 문제로 인해 **control groups version 2** 가 개발되었습니다.

Control groups version 2 (cgroups-v2)에서 모든 리소스 컨트롤러가 마운트되는 단일 컨트롤 그룹 계층 구조를 제공합니다.

제어 파일 동작 및 이름은 다른 컨트롤러간에 일관됩니다.

이 절은 Devconf.cz 2019 프레젠테이션을 기반으로했습니다. [2]

경고

Red Hat Enterprise Linux 8에서 제한된 수의 리소스 컨트롤러가 있는 테크니컬 프리뷰로 cgroups-v2 사용할 수 있습니다. 관련 리소스 컨트롤러의 자세한 내용은 [cgroups-v2 릴리즈 노트](#)를 참조하십시오.

관련 정보

- 자원 컨트롤러의 자세한 내용은 "[What are kernel resource controllers](#)" 섹션 및 `cgroups (7)man` 페이지를 참조하십시오.
- `cgroups` 계층과 `cgroups` 버전의 자세한 내용은 `cgroups(7)man` 페이지를 참조하십시오.

9.2 Linux 커널 리소스 컨트롤러란?

본 절에서는 Linux 커널의 자원 컨트롤러의 개념을 설명하고 Red Hat Enterprise Linux 8에서 컨트롤 그룹 버전 1(`cgroups-v1`) 및 컨트롤 그룹 버전 2(`cgroups-v2`)에서 지원하는 컨트롤러를 나열합니다.

`cgroup` 서브 시스템이라고도 하는 자원 제어기는 CPU 시간, 메모리, 네트워크 대역폭 또는 디스크 I/O와 같은 단일 자원을 나타냅니다. Linux 커널은 `systemd` 시스템 및 서비스 관리자가 자동으로 마운트하는 다양한 리소스 컨트롤러를 제공합니다. `/proc/cgroups` 항목에서 현재 마운트 된 자원 제어기 목록을 찾으십시오.

`cgroups-v1`는 다음의 컨트롤러를 사용할 수 있습니다.

- `blkio`-블록 장치에 대한 입출력 액세스 제한을 설정합니다.

- `cpu`-CPU 스케줄러를 사용하여 제어 그룹 작업에 CPU에 대한 액세스 권한을 제공합니다. `cputacct` 컨트롤러와 함께 동일한 마운트에 마운트됩니다.
- `cputacct`-제어 그룹의 작업에서 사용하는 CPU 리소스에 대한 자동 보고서를 만듭니다. `cpu` 동일한 마운트에 컨트롤러와 함께 마운트됩니다.
- `cpuset`-멀티 코어 시스템 및 메모리 노드의 개별 CPU를 제어 그룹의 작업에 할당합니다.
- `devices`-제어 그룹의 작업을 위한 장치에 대한 액세스 권한을 부여하거나 거부합니다.
- `freezer`-제어 그룹에서 작업을 일시 중단하거나 다시 시작합니다.
- `memory`-제어 그룹의 작업에 의한 메모리 사용 제한을 설정하고 해당 작업에 사용되는 메모리 리소스에 대한 자동 보고서를 생성합니다.
- `NET_cls`- 특정 제어 그룹 작업에서 발생하는 패킷을 식별 할 수 있도록하기 위해 Linux 트래픽 컨트롤러 (`tc` 명령)를 사용하는 클래스 식별자 (`classic`)에서 네트워크 패킷에 태그를 추가합니다. `net_cls` 서브 시스템 `net_filter`(`iptables`)하지만이 태그를 사용하여 이러한 패킷에 대한 작업을 수행 할 수 있습니다. `net_filter` 방화벽 식별자 (`fwid`)에서 네트워크 소켓을 태그합니다. 그러면 Linux 방화벽 (`iptables` 명령)가 특정 제어 그룹 작업에서 발생하는 패킷을 식별 할 수 있습니다.
- `net_prio`-네트워크 트래픽의 우선 순위를 설정합니다.
- `pids`-제어 그룹의 프로세스 및 해당 자식 수에 대한 제한을 설정합니다.
- `perf_event`- 모니터링 할 수 `cgroups` 와 `perf` 도구입니다.
- `rdma`-제어 그룹의 원격 직접 메모리 액세스(RDMA)/InfiniBand 특정 자원에 대한 제한을 설정합니다.
- `hugetlb`-큰 가상 메모리 페이지를 사용하고 이 페이지에서 리소스 제한을 적용 할 수 있습니다.

`cgroups-v2` 는에서 사용할 수 있는 컨트롤러는 다음과 같습니다.

- `io-cgroups-v1` 의 `blkio`에 대한 후속
- `memory-cgroups-v1` 의 `memory`에 대한 후속
- `pids-cgroups-v1` 의 `pids` 와 동일
- `rdma-cgroups-v1` 의 `rdma` 와 동일
- `cpu-cgroups-v1` 의 `cpu` 와 `cputacct`에 대한 후속

중요

주어진 자원 제어기는 cgroups-v1 계층 구조나 cgroups-v2 계층 구조로 동시에 사용될 수는 없습니다.

추가 자료

- 일반적인 자원 제어기에 대한 자세한 정보는 `cgroups(7)` 매뉴얼 페이지를 참조하십시오.
- 특정 자원 제어기에 대한 자세한 설명은 `/usr/share/doc/kernel-doc-<kernel_version>/Documentation/cgroups-v1/` 디렉토리의 문서를 참조하십시오 .
- 자세한 내용은 대한 `cgroups-v2` 의 참조 `cgroups(7)` 매뉴얼 페이지를 참조하십시오.

9.3. 네임스페이스(namespace)란?

이 섹션에서는 네임스페이스의 개념, 제어 그룹과의 연결 및 리소스 관리에 대해 설명합니다 .

네임스페이스는 `/proc/self/ns/cgroup` 인터페이스를 통해 격리된 시스템 리소스를 가상으로 볼 수 있는 커널 기능입니다 . 시스템 자원에서 프로세스를 분리함으로써 프로세스가 상호 작용할 수 있는 것을 지정하고 제어 할 수 있습니다.

목적은 전역 네임스페이스에서 `cgroup`으로 권한있는 데이터가 유출되는 것을 방지하고 컨테이너 마이그레이션과 같은 다른 기능을 활성화하는 것입니다.

다음 네임스페이스가 지원됩니다.

Mount

- 마운트 네임스페이스는 파일 시스템 마운트 포인트를 분리하여 각 프로세스가 작동 할 수 있는 고유한 파일 시스템 공간을 갖도록합니다.

UTS

- 호스트 이름과 NIS 도메인 이름

IPC

- System V IPC 는 POSIX 메시지 큐

PID

- 프로세스 ID

Network

- 네트워크 장치, 스택, 포트 등.

User

- 사용자 및 그룹 ID

Control groups

- cgroup 의 격리

관련 정보

- 네임스페이스의 자세한 내용은 `namespace(7)` 및 `cgroup_namespaces (7)man` 페이지를 참조하십시오.
- cgroups 자세한 내용은 "[컨트롤 그룹은](#)" 을 참조하십시오.

9.4 가상 파일 시스템을 통한 컨트롤 그룹 사용

다음 섹션에서는 `/sys/fs/` 가상 파일 시스템을 사용하여 컨트롤 그룹 (`cgroup`)의 생성, 수정 및 삭제와 관련된 작업의 개요를 설명합니다.

9.4.1. cgroups-v1을 통해 응용 프로그램에 대한 메모리 제한 설정

이 단계에서는 `/sys/fs/` 가상 파일 시스템을 사용하여 컨트롤 그룹 버전 1(`cgroups-v1`)에서 응용 프로그램의 메모리 제한을 설정하는 방법을 설명합니다.

전제 조건

- 제한하는 응용 프로그램
- root 권한
- 컨트롤 그룹의 기본적인 개요

단계

1. 메모리 리소스 컨트롤러 디렉토리에 서브 디렉토리를 작성합니다.

```
# mkdir /sys/fs/cgroup/memory/example/
```

위의 디렉토리는 특정 프로세스를 배치 할 특정 메모리 제한을 프로세스에 적용 할 수 있는 컨트롤 그룹입니다.

2. 필요에 따라 새로 만든 컨트롤 그룹을 확인합니다.

```
# ll /sys/fs/cgroup/memory/example/
-rw-r--r--. 1 root root 0 Apr 25 16:34 cgroup.clone_children
--w--w--w-. 1 root root 0 Apr 25 16:34 cgroup.event_control
-rw-r--r--. 1 root root 0 Apr 25 16:42 cgroup.procs
...
```

이 출력은 example 컨트롤 그룹이 부모 리소스 컨트롤러로부터 상속된 파일을 보여줍니다. 기본적으로 새로 만든 컨트롤 그룹은 시스템 메모리에 대한 액세스를 제한없이 상속합니다.

3. 컨트롤 그룹의 메모리 제한을 설정합니다.

```
# echo 700000 > /sys/fs/cgroup/memory/example/memory.limit_in_bytes
```

이 예제 명령은 메모리 제한을 700 킬로바이트로 설정합니다.

4. 제한 설정 확인:

```
# cat /sys/fs/cgroup/memory/example/memory.limit_in_bytes
696320
```

이 출력은 메모리 제한을 4096 바이트의 배수 (커널 페이지 크기)로 표시합니다.

5. 응용 프로그램의 PID 를 컨트롤 그룹에 추가합니다.

```
# echo 23453 > /sys/fs/cgroup/memory/example/cgroup.procs
```

이 예제 명령은 원하는 응용 프로그램이 컨트롤 그룹에서 설정 한 메모리 제한을 초과하지 않도록합니다. PID 는 시스템의 기존 프로세스에서 검색됩니다. 여기에서 PID23453 가상의 ID 입니다.

6. 응용 프로그램이 지정된 컨트롤 그룹에서 실행되고 있는지 확인합니다.

```
# ps -o cgroup 23453
CGROUP
11:memory:/example,5:devices:/system.slice/example.service,4:pids:/system.slice/example.service,1:name=systemd:/system.slice/example.service
```

위의 샘플 출력은 원하는 응용 프로그램의 프로세스 example 컨트롤 그룹에서 실행되는 응용 프로그램 프로세스에 메모리 제한이 적용되는 것을 보여줍니다.

관련 정보

- 자원 컨트롤러의 자세한 내용은 "[Linux 커널 리소스 컨트롤러는](#)" 섹션 및 cgroups (7)man 페이지를 참조하십시오.
- /sys/fs/자세한 내용은 sysfs(5)man 페이지를 참조하십시오.

[Linux Control Group v2-소개, Waiconf Long 의 Devconf.cz 2019](#) 프레젠테이션 [1]

[Linux Control Group v2-소개, Waiconf Long 의 Devconf.cz 2019](#) 프레젠테이션 [2]

제 10 장 BPF Compiler Collection 으로 시스템 성능 분석

시스템 관리자는 BCC(BPF Compiler Collection) 라이브러리를 사용하여 Linux 운영 체제의 성능을 분석하고 정보를 수집할 수 있는 도구를 만들 수 있으며, 다른 인터페이스를 통해 정보를 얻기 어려울 수 있다.

중요

BCC 라이브러리는 AMD 및 Intel 64 비트 아키텍처에서만 완벽하게 지원하고 있습니다. 다른 모든 아키텍처에서는 BCC는 기술 프리뷰로 남아 있습니다. 자세한 내용은 "[기술 프리뷰 가능 지원 범위](#)"를 참조하십시오.

10.1. BCC

BPF 컴파일러 컬렉션 (BCC)은 eBPF (extended Berkeley Packet Filter) 프로그램의 작성을 용이하게하는 라이브러리입니다. 이러한 주요 유ти리티는 오버 헤드와 보안상의 문제가 발생하지 않고 OS 성능 및 네트워크 성능을 분석하는 것입니다.

BCC는 사용자가 eBPF에 대한 심층적 인 기술적 세부 사항을 알 필요가 없으며 사전 작성된 eBPF 프로그램이 포함 된 `bcc-tools` 패키지와 같이 즉시 사용 가능한 많은 시작점을 제공합니다.

참고

EBPF 프로그램은 디스크 I/O TCP 연결 프로세스 생성 등의 이벤트에서 트리거됩니다. 프로그램이 커널의 안전 가상 머신에서 실행하기 위해 커널이 크래시하고, 루프하거나 응답하지 않을 경우는 거의 없습니다.

관련 정보

- BCC 대한 자세한 내용은 `/usr/share/doc/bcc/README.md` 파일을 참조하십시오.

10.2. BCC 설치

이 섹션에서는 **BCC (BPF Compiler Collection)** 라이브러리를 포함 `bcc-tools` 패키지를 설치하는 방법을 설명합니다.

전제 조건

- 활성 [Red Hat Enterprise Linux 서브스크립션](#)
- `bcc-tools` 패키지를 포함하는 [활성화된 저장소](#)
- `yum` [패키지 매니저](#)
- [업데이트 된 커널](#)

단계

1. `bcc-tools` 를 설치합니다.

```
# yum install bcc-tools
```

설치가 완료되면 도구는 `/usr/share/bcc/tools/` 디렉토리에 있습니다.

2. 필요한 도구를 확인합니다.

```
# ll /usr/share/bcc/tools/
...
-rwxr-xr-x. 1 root root 4198 Dec 14 17:53 dcsnoop
-rwxr-xr-x. 1 root root 3931 Dec 14 17:53 dcstat
-rwxr-xr-x. 1 root root 20040 Dec 14 17:53 deadlock_detector
-rw-r--r--. 1 root root 7105 Dec 14 17:53 deadlock_detector.c
drwxr-xr-x. 3 root root 8192 Mar 11 10:28 doc
-rwxr-xr-x. 1 root root 7588 Dec 14 17:53 execsnoop
-rwxr-xr-x. 1 root root 6373 Dec 14 17:53 ext4dist
-rwxr-xr-x. 1 root root 10401 Dec 14 17:53 ext4slower
...
```

`doc` 위 목록의 디렉토리에는 각 도구에 대한 설명서가 포함되어 있습니다.

10.3. bcc-tools로 성능 분석

이 섹션에서는 **BPF 컴파일러 컬렉션 (BCC)** 라이브러리에서 특정 미리 만들어진 프로그램을 사용하여 이벤트에 대해 시스템 성능을 효율적이고 안전하게 분석하는 방법을 설명합니다. **BCC** 라이브러리에서 미리 만들어진 프로그램 세트는 추가 프로그램 작성 예제로 사용할 수 있습니다.

전제 조건

- [BCC의 개요](#)
- [설치되어있는 BCC 라이브러리](#)
- root 권한

execsnoop를 사용하여 시스템 프로세스의 검증

1. 하나의 터미널에서 execsnoop 프로그램을 실행합니다.

```
# /usr/share/bcc/tools/execsnoop
```

2. 다른 터미널에서 다음과 같이 실행합니다.

```
$ ls /usr/share/bcc/tools/doc/
```

그러면 ls 명령의 단명 프로세스가 생성됩니다.

3. execsnoop를 실행하는 터미널은 다음과 같은 출력을 표시합니다.

PCOMM	PID	PPID	RET	ARGS
ls	8382	8287	0	/usr/bin/ls --color=auto /usr/share/bcc/tools/doc/
sed	8385	8383	0	/usr/bin/sed s/^ *[0-9]+\+ */
...				

execsnoop 프로그램은 새로운 프로세스마다 출력 행을 출력하기 위해 시스템 리소스를 소모합니다. 또한 ls 같은 매우 단기간에 실행되는 프로그램의 프로세스를 검색합니다. 또한, 대부분의 모니터링 도구는 등록하지 않습니다.

위의 결과는 부모 프로세스 이름 (ls) 프로세스 ID (5076) 부모 프로세스 ID (2931) exec() 시스템 호출의 반환 값 (0)를 보여줍니다. 이것은 프로그램 코드를 새 프로세스에 로드합니다. 마지막으로, 출력은 인수 (/usr/bin/ls --color=auto /usr/share/bcc/tools/doc/)에서 시작한 프로그램의 위치를 표시합니다.

execsnoop 세부 사항, 예 및 옵션을 확인하려면 [/usr/share/bcc/tools/doc/execsnoop_example.txt](#) 파일을 참조하십시오.

`exec()` 자세한 내용은 `exec(3)man` 페이지를 참조하십시오.

opensnoop 를 사용하여 명령어가 오픈하는 파일 추적

- 하나의 터미널에서 `opensnoop` 프로그램을 실행합니다.

```
# /usr/share/bcc/tools/opensnoop -n uname
```

위의 출력은 `uname` 명령의 프로세스에 의해서만 열리는 파일의 내용이 출력됩니다.

- 다른 터미널에서 다음을 수행합니다.

```
$ uname
```

위의 명령은 특정 파일을 엽니다. 이 파일은 다음 단계에서 캡처됩니다.

- `opensnoop` 를 실행하는 터미널은 다음과 같은 출력을 표시합니다.

PID	COMM	FD	ERR	PATH
8596	uname	3	0	/etc/ld.so.cache
8596	uname	3	0	/lib64/libc.so.6
8596	uname	3	0	/usr/lib/locale/locale-archive
...				

`opensnoop` 프로그램은 전체 시스템에서 `open()` 시스템 호출을 모니터링하고 `uname` 이 열려고 하는 파일에 대해 아웃풋을 출력합니다.

위의 결과는 프로세스 ID (`PID`) 프로세스 이름 (`COMM`) 및 파일 디스크립터 (`FD`)를 표시합니다. `open()`이 반환 값은 열려있는 파일을 참조하십시오. 마지막으로, 출력은 오류 (`ERR`) 란과 `open()`이 열려고 하는 파일 위치 (`PATH`)가 표시됩니다.

명령이 존재하지 않는 파일을 로드하려고하면 `FD` 칼럼은 `-1` 둘려주고, `ERR` 칼럼은 관련된 오류에 해당하는 값을 출력합니다. 그 결과 `opennoop`은 제대로 작동하지 않는 응용 프로그램을 식별에 도움을 줄 수 있습니다.

`opensnoop` 세부 사항, 예 및 옵션을 확인하려면 `/usr/share/bcc/tools/doc/opensnoop_example.txt` 파일을 참조하십시오.

`open()` 자세한 내용은 `open(2)man` 페이지를 참조하십시오.

디스크 I/O 작업을 조사하기 위한 `biotop` 사용

- 하나의 터미널에서 `biotop` 프로그램을 실행합니다.

```
# /usr/share/bcc/tools/biotop 30
```

이 명령을 사용하면 디스크에서 I/O 작업을 수행하는 최상위 프로세스를 모니터링 할 수 있습니다. 이 인수는 명령이 30 초 요약을 생성하도록 합니다.

참고

인수를 지정하지 않으면 기본적으로 1 초마다 출력 화면이 업데이트됩니다.

- 다른 터미널에서 다음을 실행합니다.

```
# dd if=/dev/vda of=/dev/zero
```

위의 명령은 로컬 하드 디스크 장치에서 내용을 읽고 출력을 `/dev/zero` 파일에 씁니다. 이 단계에서는 `biotop`을 설명하기 위해 특정 I/O 트래픽을 생성합니다.

- `biotop`을 실행하는 터미널은 다음과 같은 출력을 표시합니다.

PID	COMM	D	MAJ	MIN	DISK	I/O	Kbytes	AVGms
9568	dd		R	252	0	vda	16294	14440636.0
48	kswapd0		W	252	0	vda	1763	120696.0
7571	gnome-shell		R	252	0	vda	834	83612.0
1891	gnome-shell		R	252	0	vda	1379	19792.0
7515	Xorg		R	252	0	vda	280	9940.0
7579	llvmpipe-1		R	252	0	vda	228	6928.0
9515	gnome-control-c		R	252	0	vda	62	6444.0
8112	gnome-terminal-		R	252	0	vda	67	2572.0
7807	gnome-software		R	252	0	vda	31	2336.0
9578	awk		R	252	0	vda	17	2228.0
7578	llvmpipe-0		R	252	0	vda	156	2204.0
9581	pgrep		R	252	0	vda	58	1748.0
7531	InputThread		R	252	0	vda	30	1200.0
7504	gdbus		R	252	0	vda	3	1164.0
1983	llvmpipe-1		R	252	0	vda	39	724.0
1982	llvmpipe-0		R	252	0	vda	36	652.0
	...							

결과는 프로세스 ID 가 9568 인 dd 프로세스가 vda 디스크에서 16,294 개의 읽기 작업을 수행했음을 보여줍니다. 읽기 작업은 총 14,440,636KB 에 이였으며 평균 I/O 시간은 3.69ms 입니다.

biotop 세부 사항, 예 및 옵션을 확인하려면 `/usr/share/bcc/tools/doc/biotop_example.txt` 파일을 참조하십시오.

dd 자세한 내용은 dd(1)man 페이지를 참조하십시오.

`xfsslower`를 사용하여 예상외로 느린 파일 시스템 작업(operation) 확인하기

1. 하나의 터미널에서 xfsslower 프로그램을 실행합니다.

```
# /usr/share/bcc/tools/xfsslower 1
```

위의 명령은 XFS 파일 시스템이 읽기, 쓰기, 열기, 또는 동기화 (`fsync`) 작업을 수행하는 데 걸린 시간을 측정합니다. 1 인수를 지정하면 1ms 보다 느린 작업(operation)만 표시됩니다.

참고

인수를 지정하지 않으면 xfsslower 기본적으로 10 ms 보다 느린 작업을 표시합니다.

- ## 2. 다른 터미널에서 다음을 수행합니다.

```
$ vim text
```

위의 명령은 vim 편집기에서 XFS 파일 시스템과의 특정 상호작용을 시작하기 위해 텍스트 파일을 작성합니다

3. xfsslower를 실행하는 터미널은 이전 단계에서 파일을 저장한 경우와 동일한 내용을 보여줍니다.

TTIME	COMM	PID	T	BYTES	OFF_KB	LAT(ms)	FILENAME
13:07:14	b'bash'	4754	R	256	0	7.11	b'vim'
13:07:14	b'vim'	4754	R	832	0	4.03	b'libgpm.so.2.1.0'
13:07:14	b'vim'	4754	R	32	20	1.04	b'libgpm.so.2.1.0'
13:07:14	b'vim'	4754	R	1982	0	2.30	b'vimrc'
13:07:14	b'vim'	4754	R	1393	0	2.52	b'getscriptPlugin.vim'
13:07:45	b'vim'	4754	S	0	0	6.71	b'text'
13:07:45	b'pool'	2588	R	16	0	5.58	b'text'

위의 각 행은 파일 시스템의 작업을 나타내며 특정 임계값보다 시간이 걸렸습니다. `xfsslower`는 파일시스템의 작동이 예상보다 느려지는 문제를 확인할때 유용하게 사용됩니다.

T 칼럼은 작업 유형 (**R**ead / **W**rite / **S**ync)을 나타내고, OFF_KB는 KB의 파일 오프셋입니다. `FILENAME` 프로세스 (`COMM`)이 읽기, 쓰기, 또는 동기화를 시도하는 파일입니다.

`xfsslower` 세부 사항, 예 및 옵션은 `/usr/share/bcc/tools/doc/xfsslower_example.txt` 파일을 참조하십시오.

`fsync` 자세한 내용은 `man` 페이지의 `fsync(2)`을 참조하십시오.

법적 고지

저작권 © 2020 Red Hat, Inc.

이 문서의 텍스트와 그림은 **Creative Commons Attribution-Share Alike 3.0 Unported** 라이센스 ("CC-BY-SA")에 따라 **Red Hat**에 의해 라이센스가 부여됩니다. CC-BY-SA에 대한 설명은 <http://creativecommons.org/licenses/by-sa/3.0/>에서 확인할 수 있습니다. CC-BY-SA에 따라 이 문서 또는 해당 문서를 배포 할 경우 원본 버전의 **URL**을 제공해야합니다.

Red Hat은 이 문서의 라이센스 제공자로서 관련 법률이 허용하는 한도 내에서 **CC-BY-SA**의 섹션 **4d**를 시행 할 권리를 포기하며 이를 주장하지 않을 것에 동의합니다.

Red Hat, Red Hat Enterprise Linux, Shadowman 로고, **Red Hat** 로고, **JBoss, OpenShift, Fedora, Infinity** 로고 및 **RHCE**는 미국 및 기타 국가에 등록 된 **Red Hat, Inc.**의 상표입니다.

Linux®는 미국 및 기타 국가에서 **Linus Torvalds**의 등록 상표입니다.

Java®는 **Oracle** 및/또는 그 계열사의 등록 상표입니다.

XFS®는 미국 및/또는 기타 국가에서 **Silicon Graphics International Corp.** 또는 그 자회사의 상표입니다.

MySQL®은 미국, 유럽 연합 및 기타 국가에서 **MySQL AB**의 등록 상표입니다.

Node.js®는 **Joyent**의 공식 상표입니다. **Red Hat**은 공식 **Joyent Node.js** 오픈 소스 또는 상업용 프로젝트와 공식적으로 관련되거나 보증하지 않습니다.

OpenStack® Word Mark 및 **OpenStack** 로고는 미국 및 기타 국가에서 **OpenStack Foundation**의 등록 상표/서비스 마크 또는 상표/서비스 마크이며 **OpenStack Foundation**의 허가하에 사용됩니다. 우리는 **OpenStack Foundation** 또는 **OpenStack** 커뮤니티와 제휴, 보증 또는 후원하지 않습니다.

다른 모든 상표는 해당 소유자의 자산입니다.