목차

DRBD 사용자 안내서

출처 : https://docs.linbit.com/docs/users-guide-9.0/

DRBD 소개

DRBD 기본 사항

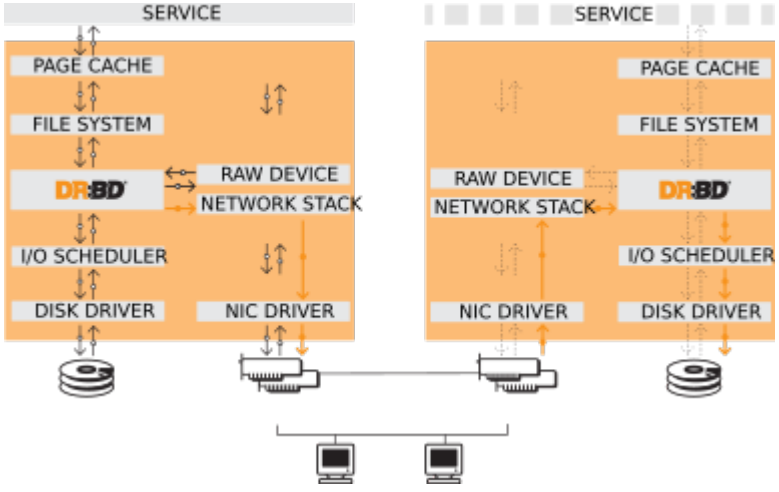
DRBD는 호스트간에 블록 장치 (하드 디스크, 파티션, 논리 볼륨 등)의 컨텐츠를 미러링하는 소프트웨어 기반의 공유 기능이없는 복제 된 스토리지 솔루션입니다. DRBD는 데이터를 미러링합니다.

- 실시간으로. 응용 프로그램이 장치의 데이터를 수정하는 동안 복제가 계속 발생합니다.
- 투명하게. 응용 프로그램은 데이터가 여러 호스트에 저장되어 있다는 것을 인식 할 필요가 없습니다.
- 동기 또는 비동기. 동기식 미러링을 사용하면 응용 프로그램은 모든 (연결된) 호스트에서 쓰기가 수행 된 후 쓰기 완료를 알립니다. 비동기 미러링을 사용하면 응용 프로그램이 로컬에서 완료 될 때 쓰기 완료를 통보 받게됩니다. 일반적으로 쓰기가 완료되면 다른 호스트로 전파됩니다.

커널 모듈

DRBD의 핵심 기능은 Linux 커널 모듈을 통해 구현됩니다. 특히, DRBD는 가상 블록 장치 용 드라이버를 구성하므로 DRBD는 시스템의 I / O 스택 맨 아래에 있습니다. 이 때문에 DRBD는 매우 유연하고 다양한 기능을 제공하므로 모든 응용 프로그램에 고 가용성을 추가하는 데 적합한 복제 솔루션입니다.

DRBD는 정의에 따라 Linux 커널 아키텍처가 위임 한대로 계층 위의 계층을 불가지론합니다. 따라서 DRBD가 갖고 있지 않은 상위 계층에 기능을 기적적으로 추가하는 것은 불가능합니다. 예를 들어 DRBD는 파일 시스템 손상을 자동 감지하거나 ext3 또는 XFS와 같은 파일 시스템에 활성 - 활성 클러스터링 기능을 추가 할 수 없습니다.



DRBD 특징

Single-primary mode : 단일 기본 모드 : 싱글 마스터

단일 기본 모드에서 자원은 주어진 시간에 단 하나의 클러스터 구성원에 대한 기본 역할에 있습니다. 언제든지 하나의 클러스터 노드 만 데이터를 조작하도록 보장되므로이 모드는 기존 파일 시스템 (ext3, ext4, XFS 등)과 함께 사용할 수 있습니다.

단일 기본 모드에서 DRBD를 배포하는 것은 고 가용성 (장애 조치 가능) 클러스터에 대한 표준 접근 방식입니다.

Dual-primary mode : 이중 기본 모드 : 멀티 마스터

이중 기본 모드에서 자원은 주어진 시간에 두 개의 클러스터 노드에서 주 역할을 수행합니다. 데이터에 대한 동시 액세스가 가능하기 때문에이 모드에서는 분산 잠금 관리자를 사용하는 공유 클러스터 파일 시스템을 사용해야합니다. 예로는 GFS 및 OCFS2가 있습니다.

듀얼 기본 모드에서 DRBD를 배포하는 것은 두 노드에서 동시에 데이터 액세스가 필요한 클러스터의로드 균형 조정에 적합한 방법입니다 (예 : 실시간 마이그레이션이 필요한 가상화 환경 이 모드는 기본적으로 비활성화되어 있으므로 DRBD의 구성 파일에서 명시 적으로 활성화해야합니다.

특정 자원에 대한 이중 기본 모드 활성화에 대한 정보는 이중 기본 모드 활성화를 참조하십시오 .

현재 DRBD-9.0 버전을 이중 기본 모드로 실행하는 것은 권장되지 않습니다 (테스트 부족으로 인해). DRBD-9.1에서는 동시에 2 개 이상의 마스터노드를 가질 수 있습니다.

복제 모드

DRBD는 3 가지 복제 모드를 지원하여 3 가지 복제 동시성을 허용합니다.

프로토콜 A : Async

비동기 복제 프로토콜. 기본 노드에서의 로컬 쓰기 작업은 로컬 디스크 쓰기가 완료되고 복제 패킷이 로컬 TCP 송신 버퍼에 배치되는 즉시 완료된 것으로 간주됩니다. 강제 장애 조치 (failover)가 발생할 경우 데이터 손실이 발생할 수 있습니다. 대기 노드의 데이터는 장애 극복 후 일관성이 있습니다. 그러나 충돌 이전에 수행 된 가장 최근 업데이트가 손실 될 수 있습니다. 프로토콜 A는 장거리 복제 시나리오에서 가장 자주 사용됩니다. DRBD 프록시와 함께 사용하면 효과적인 재해 복구 솔루션이됩니다. 자세한 내용은 DRBD 프록시를 통한 장거리 복제를 참조하십시오.

프로토콜 B : Semi-Async

메모리 동기 (반 동기) 복제 프로토콜 기본 디스크 쓰기가 발생하고 복제 패킷이 피어 노드에 도달하면 즉시 기본 노드의 로컬 쓰기 작업이 완료된 것으로 간주됩니다. 일반적으로 강제 장애 조치 (failover)의

경우 쓰기가 손실되지 않습니다. 그러나 두 노드에서 동시에 전원이 중단 되고 동시에 주 데이터 저장소가 비가 역적으로 파괴되는 경우 주 노드에서 완료된 최신 쓰기가 손실 될 수 있습니다.

프로토콜 C : Sync

동기 복제 프로토콜. 기본 노드의 로컬 쓰기 작업은 로컬 디스크 쓰기와 원격 디스크 쓰기가 모두 확인 된 후에 만 완료된 것으로 간주됩니다. 따라서 단일 노드의 손실로 인해 데이터 손실이 발생하지 않습니다. 물론 모든 노드 (저장소 하위 시스템)가 동시에 비가역 적으로 파괴되는 경우이 복제 프로토콜을 사용하더라도 데이터 손실이 불가피합니다.

지금까지 DRBD 설정에서 가장 일반적으로 사용되는 복제 프로토콜은 프로토콜 C입니다.

복제 프로토콜의 선택은 배포의 두 가지 요소 인 보호 및 대기 시간에 영향을줍니다 . 대조적으로 처리량은 선택된 복제 프로토콜과 크게 독립적입니다.

DRBD 설치

수세 리눅스 엔터프라이즈 서버

SLES 고 가용성 확장 (HAE)에는 DRBD가 포함됩니다. SLES에서 DRBD는 일반적으로 YaST2의 소프트웨어 설치 구성 요소를 통해 설치됩니다. 고 가용성 패키지 선택 사항과 함께 번들로 제공됩니다. 명령 줄 설치를 선호하는 사용자는 단순히 다음을 발행 할 수 있습니다.

```
# yast -i drbd
```

또는

```
# zypper install drbd
```

CentOS

CentOS는 릴리스 5 이후 DRBD 8을 가졌습니다. DRBD 9의 경우 EPEL, ELRepo 및 유사한 소스를 살펴봐야합니다. DRBD는 다음을 사용하여 설치할 수 있습니다 yum(이 기능을 사용하려면 올바른 저장소가 활성화되어 있어야 함).

```
# yum install drbd kmod-drbd
```

우분투 리눅스

Ubuntu LTS의 경우 LINBIT은 https://launchpad.net/~linbit/+archive/ubuntu/linbit-drbd9-stack에 PPA 저장소를 제공합니다 . 자세한 내용은 런치 패드 PPA 저장소 추가 를 참조하십시오.

```
# apt-get install drbd-utils python-drbdmanage drbd-dkms
```

DRBD 설정하기

- 단순 DRBD 설정

```
global {
  usage-count yes;
}
common {
  net {
    protocol C;
  }
}
```

- 단일 볼륨 DRBD 설정

```
resource r0 {
  on alice {
    device    /dev/drbd1;
    disk      /dev/sda7;
    address   10.1.1.31:7789;
    meta-disk internal;
  }
  on bob {
    device    /dev/drbd1;
    disk      /dev/sda7;
    address   10.1.1.32:7789;
    meta-disk internal;
  }
}
```

- 다중 볼륨 DRBD 설정

```
resource r0 {
  volume 0 {
    device    /dev/drbd1;
    disk      /dev/sda7;
    meta-disk internal;
  }
  volume 1 {
    device    /dev/drbd2;
    disk      /dev/sda8;
    meta-disk internal;
  }
```

```
  on alice {
    address   10.1.1.31:7789;
  }
  on bob {
    address   10.1.1.32:7789;
  }
}
```

global섹션

이 섹션은 구성에서 한 번만 허용됩니다. 일반적으로 /etc/drbd.d/global_common.conf 파일에 있습니다. 단일 파일 구성에서는 구성 파일의 맨 위로 이동해야합니다. 이 섹션에서 사용할 수있는 몇 가지 옵션 중 하나만이 대부분의 사용자와 관련이 있습니다.

usage-count

DRBD 프로젝트는 다양한 DRBD 버전 사용에 대한 통계를 보관합니다. 이는 시스템에 새 DRBD 버전이 설치 될 때마다 HTTP 서버에 문의하여 수행됩니다. 설정을 통해이 기능을 비활성화 할 수 있습니다 usage-count no;. 기본값은 usage-count ask; DRBD를 업그레이드 할 때마다 메시지를 표시하는 것입니다.

DRBD의 사용 통계는 공개되어 있습니다 (http://usage.drbd.org 참조).

common섹션

이 절에서는 모든 자원에서 상속 된 구성 설정을 정의하는 약식 방법을 제공합니다. 일반적으로 /etc/drbd.d/global_common.conf에 설정되며 자원별로 옵션을 정의 할 수 있습니다.

common섹션을 포함시키는 것은 엄격하게 요구되는 것은 아니지만 둘 이상의 리소스를 사용하는 경우 강력히 권장됩니다. 그렇지 않으면 구성이 반복적으로 사용되는 옵션에 의해 빠르게 복잡하게됩니다.

위의 예에서 우리는 common섹션에 net { protocol C; }를 선언하였습니다. 이 선언은 모든 자원이 이 옵션을 상속 받아 동일하게 적용됨을 뜻합니다. 다만 resource에 동일한 옵션을 선언하는 경우는 그 옵션값이 우선합니다. 사용 가능한 다른 동기화 프로토콜은 복제 모드를 참조하십시오 .

resource섹션

일반적으로 /etc/drbd.d/<resource>.res 파일에 설정됩니다. /etc/drbd.conf내에 리소스 파일의 확장자가 *.res 로 선언되어있습니다. 정의한 모든 DRBD 자원은 구성에서 자원 이름을 지정하여 이름을 지정해야합니다. 규약은 문자, 숫자 및 밑줄 만 사용하는 것입니다.

모든 리소스 구성에는 적어도 두 개의 on host 하위 섹션 (모든 클러스터 노드에 하나씩)이 있어야합니다. 다른 모든 구성 설정은 common 섹션 에서 상속되거나(존재하는 경우) DRBD의 기본 설정에서 파생됩니다.

또한 모든 호스트에서 동일한 값을 갖는 옵션을 resource섹션 에서 직접 지정할 수 있습니다 . 따라서 다음과 같이 예제 구성을 더 압축 할 수 있습니다.

```
resource r0 {
```

```
  device    /dev/drbd1;
  disk      /dev/sda7;
  meta-disk internal;
  on alice {
    address   10.1.1.31:7789;
  }
  on bob {
    address   10.1.1.32:7789;
  }
}
```

초기 동기화 구성 방안

initial device synchronization : 마스터 동기화

마스터 역할의 노드를 기준으로 다른 노드쪽으로 복제동기화를 수행하는 방법입니다. 초기에 복제해야 하는 데이터의 양이 많지 않거나 시간적으로 여유가 있을때 수행할 수 있는 방법입니다.

마스터 노드에서 아래 명령어를 수행합니다.

```
# drbdadm primary --force <resource>
```

Using truck based replication : 트럭 복제 동기화

복제해야할 데이터 양이 많거나 시간적 여유가 없는경우, 또는 지리적으로 먼곳과 동기화를 해야할때 사용할 수 있는 방법입니다. 먼저 마스터 노드의 데이터를 동기화 설정을 진행하여 해당 데이터를 통째로 원격지로 복사한 후 그 이후 시점부터 동기화를 진행하는 방법입니다.

먼저 마스터 노드에서 아래의 명령어를 수행합니다.

```
# drbdadm new-current-uuid --clear-bitmap <resource>/<volume>
```

또는

```
# drbdsetup new-current-uuid --clear-bitmap <minor>
```

이후 해당 장치를 복제하십시요. 복제하는 방법은 RAID1 볼륨의 경우 디스크 1개를 대체하거나 또는 디스크 이미지를 dd로 복제하는 방법이 있습니다. (file-copy 방식은 정상 작동하지 않습니다.)

복제가 완료되면 아래 명령어를 다시 수행합니다.

```
# drbdadm new-current-uuid <resource>
```

이번에는 옵션에 –clear-bitmap 이라는 옵션이 빠져있는것에 주의하시기 바랍니다. 즉 복제를 위한 정보를 초기화하고 해당 시점으로 새롭게 생성하는것이라고 보면 되겠습니다.

복제본을 원격지 노드에 구성합니다. 원격지 노드에 복제본을 구성한 후 메타데이터 안의 노드 정보를 수정하여야 합니다. 아래의 명령어를 수행하여 리소스 r0의 볼륨 0에 대해서 2번 노드를 1번 노드로 정보를 수정합니다.

```
V=r0/0
NODE_FROM=2
NODE_TO=1

drbdadm -- --force dump-md $V > /tmp/md_orig.txt
sed -e "s/node-id $NODE_FROM/node-id $NODE_TO/" \
    -e "s/^peer.$NODE_FROM. /peer-NEW /" \
    -e "s/^peer.$NODE_TO. /peer[$NODE_FROM] /" \
    -e "s/^peer-NEW /peer[$NODE_TO] /" \
    < /tmp/md_orig.txt > /tmp/md.txt

drbdmeta --force $(drbdadm sh-minor $V) v09 $(drbdadm sh-ll-dev $V) internal
restore-md /tmp/md.txt
```

여기까지 완료되면 이제 원격지 노드를 활성화 합니다.

```
# drbdadm up <resource>
```

이제 각 노드가 연결되면 전체동기화를 수행하는것이 아니라 새로운 시점을 기준으로 변경된 블럭만 처리하게 됩니다.

4개 노드 연결 설정

일반적으로 4개 노드 연결 설정은 아래와 같습니다.

```
resource r0 {
  device      /dev/drbd0;
  disk        /dev/vg/r0;
  meta-disk   internal;

  on store1 {
    address   10.1.10.1:7100;
    node-id   1;
  }
  on store2 {
    address   10.1.10.2:7100;
    node-id   2;
  }
```

```
  on store3 {
    address   10.1.10.3:7100;
    node-id   3;
  }
  on store4 {
    address   10.1.10.4:7100;
    node-id   4;
  }

  # All connections involving store1
  connection {
    host store1  port 7012;
    host store2  port 7021;
  }
  connection {
    host store1  port 7013;
    host store3  port 7031;
  }
  connection {
    host store1  port 7014;
    host store4  port 7041;
  }

  # All remaining connections involving store2
  connection {
    host store2  port 7023;
    host store3  port 7032;
  }
  connection {
    host store2  port 7024;
    host store4  port 7042;
  }

  # All remaining connections involving store3
  connection {
    host store3  port 7034;
    host store4  port 7043;
  }

  # store4 already done.
}
```

노드 4대가 Full-mesh 구조로 연결되어야 하기 때문에 1-2, 1-3, 1-4, 2-3, 2-4, 3-4 연결 설정이 필요합니다.

위 구성 그대로 더욱 단순한 형태로 구성하면 아래와 같습니다.

```
resource r0 {
```

```
  device      /dev/drbd0;
  disk        /dev/vg/r0;
  meta-disk   internal;

  on store1 {
    address   10.1.10.1:7100;
    node-id   1;
  }
  on store2 {
    address   10.1.10.2:7100;
    node-id   2;
  }
  on store3 {
    address   10.1.10.3:7100;
    node-id   3;
  }
  on store4 {
    address   10.1.10.4:7100;
    node-id   4;
  }

  connection-mesh {
    hosts     store1 store2 store3 store4;
  }
}
```

각노드에 여러개의 NIC가 있고 각각 노드간 서로 다른 IP로 통신해야하는경우 아래처럼 구성할 수도 있습니다.

```
resource r0 {
  ...

  # store1 has crossover links like 10.99.1x.y
  connection {
    host store1  address 10.99.12.1 port 7012;
    host store2  address 10.99.12.2 port 7021;
  }
  connection {
    host store1  address 10.99.13.1  port 7013;
    host store3  address 10.99.13.3  port 7031;
  }
  connection {
    host store1  address 10.99.14.1  port 7014;
    host store4  address 10.99.14.4  port 7041;
  }

  # store2 has crossover links like 10.99.2x.y
  connection {
    host store2  address 10.99.23.2  port 7023;
    host store3  address 10.99.23.3  port 7032;
```

```
  }
  connection {
    host store2  address 10.99.24.2  port 7024;
    host store4  address 10.99.24.4  port 7042;
  }

  # store3 has crossover links like 10.99.3x.y
  connection {
    host store3  address 10.99.34.3  port 7034;
    host store4  address 10.99.34.4  port 7043;
  }
}
```

리소스 초기화

위와 같이 설정파일(/etc/drbd.d/*.conf)을 통하여 리소스를 정의한 후에 해당 리소스를 초기화 시켜야 한다. 리소스를 초기화하기 위해 아래 명령어로 메타데이터를 생성한다

메타데이타 생성

```
# drbdadm create-md <resource>
```

DRBD 상태 확인

DRBD의 상태를 살펴볼 수 있는 편리한 방법 중 하나는 drbd-overview 유틸리티입니다

```
nina# drbd-overview
  0:r0/0  Connected(*) Seco(*)/Prim(nina) UpTo(*)/Disk(nono)
            /mnt ext3 1008M 18M 940M 2%
  1:r1/0  Connected(*) Secondary(*)       UpTo(*)/Disk(nono)
  5:r2/0  Connected(*) Seco(*)/Prim(nini) UpTo(*)/Disk(nono)
  6:r2/1  Connected(*) Seco(*)/Prim(nini) UpTo(*)/Disk(nono)
```

이 결과는

- r0 볼륨 0은 호스트 nina(로컬 호스트)의 Primary 이고 ext3 파일 시스템으로 /mnt에 마운트 됩니다.
- 호스트 nono는 모든 리소스에 대해 디스크없는 상태 입니다.(DRBD 클라이언트)
- 다른 모든 호스트는 모든 자원에 대해 UpToDate 입니다.
- r1 은 모든 호스트에서 세컨더리 입니다. 사용되지 않음.
- r2 는 호스트 nini에서 사용됩니다.