스토리지 기술정보

- centos6.x drbd설치
- Ceph 스토리지 구축
- DRBD 기술노트
- Glusterfs rpm 설치
- RHEL환경에서 VDO 사용하기

centos6.x drbd설치

Centos6에서 DRBD rpm 으로 설치

```
[root@localhost ~]# rpm -ivh http://elrepo.org/elrepo-release-6-5.el6.elrepo.noarch.rpm
[root@localhost ~]# yum install *drbd84* -y
[root@localhost ~]# modprobe drbd
[root@localhost ~]# lsmod | grep drbd
drbd 297925 0
libcrc32c 841 1 drbd
```

Ceph 스토리지 구축

ceph 소개

1. 분산 객체 스토리지를 구성하는 OSS
2. 서버 구성은 OSD, Monitor, Manager, MDS 서버가 필요
3. 논리적으로 구성한 Storage pool안에서 데이터를 개체로 저장. Crush알고리즘을 사용해서 배치그룹을 계산하고 저장

구성정보

1. Component 종류
   1. ceph-mon(모니터노드) : 클러스터 상태를 체크하고, 데몬과 클라이언트간 인증관리 담당 / HA구성시 3대 필요
   2. ceph-mgr(관리노드) : 스토리지 활용도 / 현재상태 및 메트릭 추적 (dashboard 및 RestAPI 제공) / HA구성시 2대 필요(Active / Standby)
   3. ceph-osd(객체스토리지 데몬) : 데이터를 저장하고 복제 / 부하분산 역할을 수행 (OSD디스크 1TB당 메모리 1G이상으로 구성을 권고), HA구성시 최소 3대 필요
   4. ceph-mds : CEPH FS를 대신해서 메타 데이터를 관리하는 서버. = Block Devices / Object Storage에서는 MDS를 사용하지 않음
2. Component Hardware Spec

Component	Hardward	Spec
osd	CPU	OSD당 2 Core
osd	MEM	데몬당 4GB이상
osd	DISK	최소 1TB이상,(SSD 권장) 단일 디스크에서 여러 OSD 실행은 비권장 단일 디스크에서 osd+mon+mds 실행방식 비권장 OSD용 디스크는 OS와 분리해서 사용(성능저하 이슈)
osd	NIC	10G이상
mon	CPU	2코어 이상
mon	MEM	데몬당 24GB이상
mon	DISK	데몬당 60GB
mds	CPU	2코어 이상
mds	MEM	데몬당 2GB이상
mds	DISK	데몬당 1MB 이상
mds	NIC	1Gb 이상

* OSD에 RAID구성시 성능저하가 발생할 수 있으므로 BMT를 통해 성능 비교 권고

시스템 이해

1. OSD Backend
   1. Bluestore
      - Ceph 12.2이후 부터 default storage
      - 저장장치를 직접 엑세스 해서 데이터를 관리 - XFS같은 파일시스템을 사용하지 않음
      - RocksDB를 통한 메타데이터 관리
      - 전체 데이터 및 메타데이터 checksum 수행 - 무결성 유지
      - inline압축 - 디스크에 저장하기 전에 선택적으로 압축수행
      - 데이터 관리 계층화 - journal을 별도 장치에 기록할 수 있어 성능향상 가능.
      - CoW을 사용하기 때문에 기존보다 향상된 IO
   2. Filestore
      - Ceph에 개체를 저장하는 방식.
      - 일부 메타데이터에 대해 LevelDB를 사용해 key/value로 저장

    - 파일시스템을 btrfs / ext4에서 사용시 알려진 결함이 있어 데이터가 손실될 수 있음 (XFS는 영향없음)
2. Pool
    1. 개체를 저장하기 위해 사용하는 논리 파티션
        - Recovery : 데이터 손실없이 사용할 수 있도록 설정하는 OSD
        - PG : Pool에 대한 배치 그룹 수 (일반적으로 OSD당 100개의의 PG을 사용)
        - Cursh Rule : 데이터를 Pool에 저장할때 Crush Rule에 의해 결정
        - Snapshot : 특정 Pool의 스냅샷 생성
    2. Pool을 사용하기 위해서는 어플리케이션과 연결되어 있어야 하며, RBD에서 사용할 경우 RBD도구를 사용해서 초기화가 필요 (cephfs / rbd / rgw 중 택1)
3. CephFS
    1. 분산 개체 저장소인 RADOS를 기반으로 구축된 파일시스템
    2. 공유 디렉토리 및 HA를 제공
    3. CephFS는 데이터용와 메다데이터용으로 각각 2개이상의 RAODS Pool이 필요
        - 메타데이터 pool에서 데이터가 손실되면 전체파일 시스템 액세스가 불가능
        - 메타 데이터 pool에 SSD 사용
        - 데이터 풀은 파일시스템을 생성하고, 기본적으로 모든 inode 정보를 저장하는 위치
4. NFSExport
    1. NFS-Ganesha NFS를 이용해 CephFS 네임스페이스 export 가능

Ceph 설치하기 (ansible 기반의 ceph배포)

1. 설치 방법에는 cephadm / Rook / ansible을 이용한 설치방법이 존재,
    1. cephadm - 자체적으로 설치하는 binary container 혹은 python3이 필요
    2. Rook - kuernetes에서 ceph를 설치하거나 기존 ceph를 k8s로 join할때 Rook을 이용
    3. ceph-deploy은 최신버전에서 사용되지 않음
2. ceph-ansible을 설치하기 위한 python 패키지 설치

```
$ yum install -y python3 python3-pip sshpass
$ pip3 install --upgrade setuptools pip --ignore-installed
```

3. ceph-ansible 내려받기

```
$ git clone https://github.com/ceph/ceph-ansible.git -b "v6.0.13"
$ cd ceph-ansible
```

    - ceph-ansible 버전별 대응 버전

ceph-ansible	ceph	ansible
3.0	jewel / luminous	2.4
3.1	luminous / mimic	2.4
3.2	luminous / mimic	2.6
4.0	nautilus	2.9
5.0	octopus	2.9
6.0	pacific	2.9

4. dependency 패키지 설치

```
$ pip3 install -r requirements.txt
```

5. 배포를 위한 호스트파일 작성

```
$ vi hosts

[mons]
192.168.100.41

[osds]
192.168.100.41
192.168.100.42
```

```
[mdss]

[rgws]

[nfss]
192.168.100.41

[rbdmirrors]

[clients]
192.168.100.41

[mgrs]
192.168.100.41

[iscsigws]

[iscsi-gws]

[grafana-server]

[rgwloadbalancers]

[monitoring]
192.168.100.41

[all:vars]
ansible_become=true
ansible_user=root
ansible_ssh_pass=root
```

6. 환경변수 복사 (systemd 기반으로 구동시)

```
$ cp site.yml.sample site.yml
$ cp group_vars/all.yml.sample  group_vars/all.yml
$ cp group_vars/osds.yml.sample  group_vars/osds.yml
```

7. 환경변수 복사 (container 기반으로 구동시)

```
$ cp site-container.yml.sample  site.yml
$ cp group_vars/all.yml.sample  group_vars/all.yml
$ cp group_vars/osds.yml.sample  group_vars/osds.yml
```

8. config 설정 (systemd 기반으로 구동시)

```
$ vi group_vars/all.yml
...
osd_objectstore: bluestore
monitor_interface: ens3f0
public_network: 192.168.100.0/24
ntp_service_enabled: true
ntp_daemon_type: chronyd
...
##############
# DASHBOARD #
##############
dashboard_enabled: false
dashboard_protocol: http
dashboard_port: 8081
dashboard_admin_user: admin
dashboard_admin_password: adminpassword
containerized_deployment: false
...
configure_firewall: false
...
ceph_origin: repository
...
ceph_repository: community
...
```

```
ceph_stable_release: octopus
```

```
$ vi group_vars/osds.yml
...
devices:
  - /dev/sdb
...
```

```
$ vi roles/ceph-validate/tasks/main.yml
...
  #해당 name 전체 삭제
  - name: validate ceph_repository_community
    fail:
      msg: "ceph_stable_release must be 'pacific'"
    when:
      - ceph_origin == 'repository'
      - ceph_repository == 'community'
      - ceph_stable_release not in ['pacific']
...
```

Centos7에서 systemd 기반으로 구동시 dashboard가 호환되지 않아 false로 처리해야 함
ceph 릴리즈 버전 중 pacific 버전은 Centos7에서 nfs export가 되지 않아 octopus로 다운그레이드가 필요
config 설정 (ceph를 container로 구동시)

```
$ vi group_vars/all.yml
...
osd_objectstore: bluestore
monitor_interface: ens3f0
public_network: 192.168.100.0/24
ntp_service_enabled: true
ntp_daemon_type: chronyd
...
#############
# DASHBOARD #
#############
dashboard_enabled: false
containerized_deployment: true
...
```

```
$ vi group_vars/osds.yml
...
devices:
  - /dev/sdb
...
```

9. 배포

```
$ ansible-playbook  -i hosts  site.yml -b -v
```

10. cluster health check시 warn 발생시
    #Cluster 구성상태 모두 정상인데, health check warn으로 표시될 경우 조치방법 (ceph자체 버그로 의심)

```
$ ceph config set mon auth_allow_insecure_global_id_reclaim false
```

운영방법

1. ceph cluster상태 확인

```
$  ceph status
  cluster:
    id:     ca96d48d-1c9d-4168-9f21-ffda54a5cd9c
    health: HEALTH_OK

  services:
    mon: 2 daemons, quorum openstack-dev1,openstack-dev2 (age 87m)
    mgr: openstack-dev1(active, since 78m), standbys: openstack-dev2
    osd: 3 osds: 3 up (since 83m), 3 in (since 2h)

  data:
    pools:   5 pools, 105 pgs
    objects: 49 objects, 5.3 KiB
    usage:   41 MiB used, 300 GiB / 300 GiB avail
    pgs:     105 active+clean
```

2. ceph osd 상태 확인

```
$ ceph osd tree
ID  CLASS  WEIGHT   TYPE NAME             STATUS  REWEIGHT  PRI-AFF
-1         0.29306  root default
-5         0.09769      host dev1
2    hdd  0.09769          osd.2             up   1.00000  1.00000
-3         0.09769      host dev2
0    hdd  0.09769          osd.0             up   1.00000  1.00000
-7         0.09769      host dev3
1    hdd  0.09769          osd.1             up   1.00000  1.00000
```

3. ceph현재 latency 확인방법

```
$ ceph osd perf
osd  commit_latency(ms)  apply_latency(ms)
 2                   0                  0
 0                   0                  0
 1                   0                  0
```

commit은 시스템 call이 있기 때문에 일반적으로 100 ~ 600ms까지는 수용가능한 수준으로 판단
메모리내 적용된 파일을 파일시스템에 적용하는 시간 (ms단위, 실제 성능에 판단되는 시간)

4. nfs 오류시 로그 확인

```
$ cephadm logs --fsid <fsid> --name nfs.{{ clusteid }}.hostname
```

5. 파일시스템1. CephFS - Pool 관리

```
$ ceph osd lspools
```

1. Pool 생성
   #Pool 생성

```
$ ceph osd pool create {{ DATA_POOL_NAME }}
$ ceph osd pool create {{ METADATA_POOL_NAME }}
```

#CephFS는 데이터용과 메타데이터용 각각 2개이상의 RADOS풀 필요

2. 생성된 Pool을 애플리케이션에 연결 (cephfs로 연결)

```
$ ceph osd pool application enable {{ DATA_POOL_NAME }} cephfs
```

3. 파일시스템 생성

```
$ ceph fs new {{ FS_NAME }} {{ METADATA_POOL_NAME  }} {{DATANAME }}
```

4. NFS export

1. nfs module설정

```
$ ceph mgr module enable nfs
```

2. nfs ganesha 클러스터 생성

```
$ ceph nfs cluster create {{ clusterid }}
```

3. nfs export

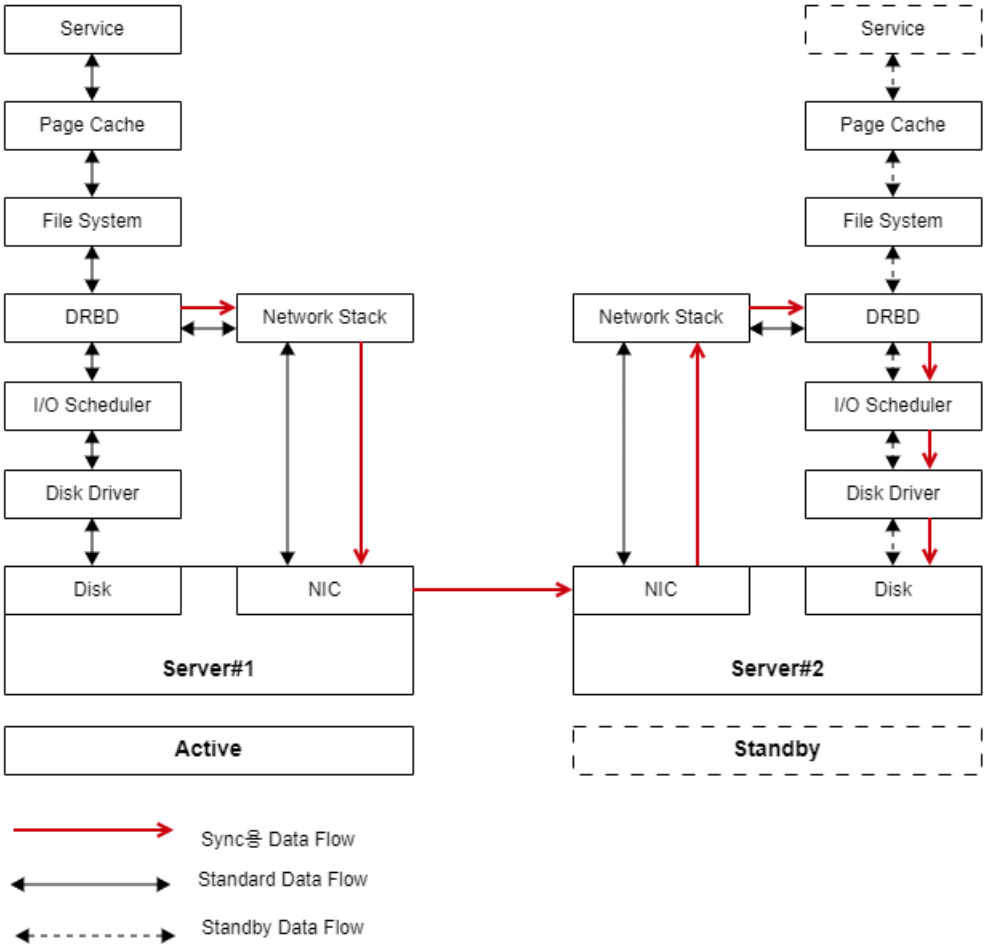
```
$ ceph nfs export create cephfs {{ NAME }} {{ clusterid }
```

reference

- https://docs.ceph.com/en/latest/architecture/
- *https://www.slideshare.net/jenshadlich/ceph-object-storage-at-spreadshirt-july-2015-ceph-berlin-meetup*

DRBD 기술노트

1. Opensource 정보 : DRBD / DRBDutils (https://www.linbit.com/en/drbd-oss-distribution/)
2. DRBD Stack (Active / Passive 구성임)
3. 작업 절차중 나오는 용어들은 Mantec 을 참고해도 좋을듯
   : https://mantech.jira.com/wiki/spaces/WDRBDV9/pages/170098908/1.
4. Mantech에 MCCS라는 솔루션이 DRBD을 기반으로 상품화되었음
5. DRBD Data Flow



DRBD설치하기

1. DRBD 모듈 설치 (8.x, 9.x 설치 방법 동일함)

```
> tar -zxvf drbd-8.4.5.tar.gz
> cd drbd-8.4.5
> make;make install
> echo "modprobe drbd" > /etc/sysconfig/modules/drbd.modules
> chmod +x /etc/sysconfig/modules/drbd.modules
> modprobe drbd
> lsmod | grep drbd
drbd                  568788  0
libcrc32c              12644  4 xfs,drbd,nf_nat,nf_conntrack
```

2. DRBD Util 설치 (8.x, 9.x 설치 방법 동일함)

```
> yum install -y libxslt libxml2
> tar -zxvf drbd-utils-8.9.1.tar.gz
> cd drbd-utils-8.9.1
> ./configure --prefix=/usr/local/drbd --sysconfdir=/etc/ --with-initscripttype=sysv
> ./configure --sysconfdir=/etc/ --with-initscripttype=systemd --with-pacemaker (Centos7)
> make; make install
> mkdir -p /usr/local/drbd/var/run
```

DRBD구성하기

1. Target 파티션 구성하기

```
>\# fdisk /dev/xvdb
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0xcce44d25.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.
\\Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)
\\WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
        switch off the mode (command 'c') and change display units to
        sectors (command 'u').
\\Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p (입력)
Partition number (1-4): 1 (입력)
First cylinder (1-2080500, default 1):
Using default value 1 (입력)
Last cylinder, +cylinders or +size{K,M,G} (1-2080500, default 2080500):
Using default value 2080500
\\Command (m for help): w (입력)
The partition table has been altered!
\\Calling ioctl() to re-read partition table.
Syncing disks.
```

2. DRBD구성하기

```
>\# cat /etc/drbd.d/drbd.res
resource drbd0
{
  startup {
    wfc-timeout 30;
    outdated-wfc-timeout 20;
    degr-wfc-timeout 30;
 }
\\  syncer {
    rate 1000M;
    verify-alg sha1;
 }
\\  on web1 {
    device drbd0;
    disk /dev/xvdb1;
    address {노드1번IP}:7789;
    meta-disk internal;
 }
\\  on web2 {
    device drbd0;
    disk /dev/xvdb1;
    address {노드2번IP}:7789;
    meta-disk internal;
 }
}
```

3. DRBD 메타 데이터 구성 후 서비스 시작

```
>\# drbdadm create-md drbd0
>\# /etc/init.d/drbd start
```

4. Disk Sync 시작 (1번 서버에서 수행) (DRBD 8.x 해당)

```
>\# drbdadm -- --overwrite-data-of-peer primary drbd0
```

5. drbd 실행 후 primary 지정 (1번서버에서 수행)

```
>\# drbdadm invalidate drbd0
>\# drbdadm  primary --force drbd0
```

6. 데이터 동기화 상태 확인DRBD 서비스 시작 (DRBD 8.x 해당)

```
>#  cat /proc/drbd
version: 8.4.5 (api:1/proto:86-101)
GIT-hash: 1d360bde0e095d495786eaeb2a1ac76888e4db96 build by root@web2, 2015-03-09 14:49:34
0: cs:SyncTarget ro:Secondary/Primary ds:Inconsistent/UpToDate C r-----
    ns:0 nr:491520 dw:487424 dr:0 al:0 bm:0 lo:16 pe:0 ua:16 ap:0 ep:1 wo:f oos:9986956
      [>.....................] sync'ed:  4.7% (9752/10228)M
       finish: 0:04:05 speed: 40,616 (40,616) want: 102,400 K/sec
```

7. 연결상태 확인 (DRBD 9.0)

```
$> drbdadm status drbd0
drbd0 role:Primary
  disk:UpToDate
  repotx-gitlab-dev02.tx.skp role:Secondary
    replication:SyncSource
      peer-disk:Inconsistent done:8.53
```

DRBD 기능 테스트

1. DRBD의 Primary 선언 (master 노드수행)

```
$> drbdadm primary drbd0
$> cat /proc/drbd    *Primary/Secondary으로 표기되어 있으면 됨
version: 8.4.5 (api:1/proto:86-101)
GIT-hash: 1d360bde0e095d495786eaeb2a1ac76888e4db96 build by root@techtx-base-dev02, 2019-04-26 17:54:37
0: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C r-----
    ns:0 nr:0 dw:0 dr:664 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:0
```

2. 파일시스템 생성 (master 노드수행)

```
>#  mkfs.ext4 /dev/drbd0
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
327680 inodes, 1310595 blocks
65529 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=1342177280
40 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736
\\Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
\\This filesystem will be automatically checked every 26 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

3. 마운트 후 파일 생성 (master 노드수행)

```
$> mount /dev/drbd0 /mnt/
$> df -h /mnt
Filesystem      Size  Used Avail Use% Mounted on
/dev/drbd0      4.8G   10M  4.6G   1% /mnt
```

4. 임의 파일 생성 (master 노드수행)

```
$> touch 123
$> ls -l 123
-rw-r--r-- 1 root root 0 Apr 29 13:04 123
```

5. 마운트 해제 및 DRBD의 secondary 선언 (master 노드수행)

```
$> umount /mnt
```

```
$> drbdadm secondary drbd0
$> cat /proc/drbd
version: 8.4.5 (api:1/proto:86-101)
GIT-hash: 1d360bde0e095d495786eaeb2a1ac76888e4db96 build by root@techtx-base-dev02, 2019-04-26 17:54:37
0: cs:Connected ro:Secondary/Secondary ds:UpToDate/UpToDate C r-----
   ns:1235836 nr:0 dw:1235836 dr:1409 al:307 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:0
```

6. DRBD의 Pimary 선언 (slave 노드 수행)

```
$> drbdadm primary drbd0
$> cat /proc/drbd
version: 8.4.5 (api:1/proto:86-101)
GIT-hash: 1d360bde0e095d495786eaeb2a1ac76888e4db96 build by root@techtx-base-dev05, 2019-04-26 17:54:29
0: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C r-----
   ns:4 nr:1235836 dw:1235840 dr:1017 al:1 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:0
```

7. 파일시스템 마운트 후 master노드에서 생성한 파일 확인 (slave 노드 수행)

```
># mount /dev/drbd0 /mnt/
># ls -l /mnt/
total 1000020
-rw-r--r-- 1 root root         0 Apr 29 13:04 123
```

DRBD장애로 인한 데이터 동기화 실패대응

1. crm_mon -1 명령어 수행을 통한 Active / Standby 정보 확인
2. Corocync / Pacemaker / DRBD 서비스 종료
3. DRBD 서비스 시작
4. 슬레이브 선언 (Standby 서버에서 수행)

```
$> drbdadm secondary all
$> drbdadm disconnect all
$> drbdadm -- --discard-my-data connect all
```

5. 마스터 선언 (Active 서버에서 수행)

```
$> drbdadm primary all
$> drbdadm disconnect all
$>drbdadm connect all
```

6. 데이터 강제 동기화 (마스터에서 작업 수행)

```
$> drbdadm invalidate drbd0
```

Glusterfs rpm 설치

Centos환경에서 GlusterFS설치 및 운용

1. 다운로드 : http://download.gluster.org/pub/gluster/glusterfs/3.3/3.3.1/CentOS/
2. 설치

```
$> yum install -y fuse
$> rpm -ivh glusterfs-3.3.1-1.el5.i386.rpm
$> rpm -ivh glusterfs-fuse-3.3.1-1.el5.i386.rpm
$> rpm -ivh glusterfs-devel-3.3.1-1.el5.i386.rpm
$> rpm -ivh glusterfs-server-3.3.1-1.el5.i386.rpm
```

3. gluster 데몬 실행으로 설정

```
/etc/rc.d/init.d/glusterd start
chkconfig glusterd on
\\* 관련경로
/etc/glusterfs/, /var/lib/glusterd/
```

4. 서버에 fuse 모듈 적용 및 확인하기

```
modprobe fuse
echo "modprobe fuse" >> /etc/rc.local
dmesg | grep -i fuse
```

fuse init (API version 7.10) <-- 해당 내용이 없으면 fuse 모듈 적재가 안된 부분임.

5. Storage Pool 구성
   1. 1. Management 서버에서 스토리지로 연결할 서버 등록(자신은 등록하지 않아도 됨)

```
gluster peer probe 서버IP
ef) gluster peer probe 192.168.150.19
```

6. Pool 확인

```
[root@localhost ]# gluster peer status
Number of Peers: 1
Hostname: 192.168.150.19
Uuid: 0bb2e93b-1ab1-4420-91c0-e5bcf203586b
State: Peer in Cluster (Connected)
```

7. Storage pool 제거

```
gluster peer detach 서버IP
\\ef) gluster peer detach 192.168.150.19
```

8. Volume 구성정보
   1. Volume 구성방식은 Distributed(분산저장, 기본)
      - 데이터가 들어오면 Brick중 한군데 서버에 저장이 됨
      - Distributed 모드에서 어느 하나 Brick이 중단될 경우, 중단된 Brick이 가지고 있는 데이터는 보이지 않게 됨.
   2. replicated(분산저장, replication 수 만큼 파일복사)
      - 데이터가 들어오면 Brick이 모두 데이터를 가지게 됨, 가장 안정적.
   3. stripe(단독저장) 로 구성가능
      1. 데이터가 들어오면 한 데이터를 Brick에 분산 저장됨
      2. Stripe 모드에서 어느 하나 Brick이 중단될 경우, 데이터가 깨짐
9. volume 생성 (Distributed 방식)

```
gluster volume create 볼륨이름 vmlnx001:/data vmlnx002:/data vmlnx003:/data vmlnx004:/data
\\ef) gluster volume create data 192.168.150.18:/data 192.168.150.19:/data
\\\\gluster> volume status data
Status of volume: data
Gluster process                                 Port    Online  Pid
\\------------------------------------------------------------------------------
```

```
\\Brick 192.168.150.18:/data                24011   Y       3245
Brick 192.168.150.19:/data                  24011   Y       2481
NFS Server on localhost                     38467   Y       3251
NFS Server on 192.168.150.19                38467   Y       2487
\\gluster> volume info data
Volume Name: data
Type: Distribute
Volume ID: 556d6065-f888-4198-8782-65bc03979a0b
Status: Started
Number of Bricks: 3
Transport-type: tcp
Bricks:
Brick1: 192.168.150.18:/data
Brick2: 192.168.150.19:/data
Brick3: 192.168.150.16:/data
```

10. volume 생성 (Replicate 방식)

```
gluster> volume create data replica 2 192.168.150.18:/data 192.168.150.19:/data
\\생성한 volume 확인
gluster> volume info all
Volume Name: data
Type: Replicate
Volume ID: c2ecd1b8-708e-47d4-8f15-adcd1b081987
Status: Started
Number of Bricks: 1 x 2 = 2
Transport-type: tcp
Bricks:
Brick1: 192.168.150.19:/data
Brick2: 192.168.150.18:/data
```

11. volume 생성 (stripe 방식)

```
gluster volume create data stripe 2 transport tcp 192.168.150.18:/data 192.168.150.19:/data
```

12. volume 생성시 기존에 생성된 volume일 경우, 생성이 실패 한다
    ef) gluster volume create data replica 3 192.168.150.18:/data 192.168.150.19:/data 192.168.150.16:/data
    \\/data or a prefix of it is already part of a volume
    - 이 경우, 등록한 노드 폴더에 들어가면, .glusterfs라는 폴더가 존재한다.

```
root@localhost data]# ls -al
합계 44
drwxr-xr-x  4 root root  4096 12월  6 16:00 .
drwxr-xr-x 24 root root  4096 12월  6 16:01 ..
drw-------  7 root root  4096 12월  6 15:17 .glusterfs
drwx------  2 root root 16384 12월  5 13:48 lost+found
```

    해결책은 해당 폴더 권한 수정 후 폴더 삭제하면 된다.

```
setfattr -x trusted.glusterfs.volume-id /data/
setfattr -x trusted.gfid /data/
rm -rf .glusterfs
```

13. 생성할 볼륨 활성화

```
gluster> volume start data
```

14. 생성한 볼륨 접근 허가 대역 지정

```
gluster volume set <volume_name> auth.allow <허가IP대역>
ef) gluster> volume set data  auth.allow 192.168.150.*
```

15. Volume 삭제 방법

```
gluster volume stop <volume_name>
```

```
gluster volume delete <volume_name>
\\ef)
gluster volume stop data
gluster volume delete data
```

16. 생성한 Volume 마운트

```
mount -t glusterfs <server_name>:<volume_name> <mount point>
mount -t glusterfs 192.168.150.18:/data /test/
\\[root@localhost ]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda2             4.9G  1.5G  3.2G  31% /
/dev/sda5             9.6G  151M  9.0G   2% /data
/dev/sda1              99M   12M   83M  13% /boot
tmpfs                1014M     0 1014M   0% /dev/shm
glusterfs#192.168.150.18:/data
                     9.6G  151M  9.0G   2% /test
```

17. 자동으로 마운트시 /etc/fstab에 내용 추가

```
192.168.150.18:/data    /mnt                 glusterfs defaults,_netdev      0 0
```

1*. 이 경우, netfs 서비스가 활성화 되어있어야 재부팅 해도 마운트를 한다.

18. Volume 확장

```
용량 부족시 증설가능 (replicate모드는 최초 설정한 Brick만큼 추가 해 주어야 한다.)
\\gluster volume add-brick dist_vol 172.27.0.9:/data
ef) gluster volume add-brick data 192.168.150.16:/data
Brick추가 후 volume 상태
\\추가 전)
gluster> volume status data
Status of volume: data
Gluster process                                 Port    Online  Pid
------------------------------------------------------------------------------
Brick 192.168.150.18:/data                      24011   Y       3245
Brick 192.168.150.19:/data                      24011   Y       2481
NFS Server on localhost                         38467   Y       3251
NFS Server on 192.168.150.16                    38467   Y       4216
NFS Server on 192.168.150.19                    38467   Y       2487
\\추가 후)
gluster> volume status data
Status of volume: data
Gluster process                                 Port    Online  Pid
------------------------------------------------------------------------------
Brick 192.168.150.18:/data                      24011   Y       3245
Brick 192.168.150.19:/data                      24011   Y       2481
Brick 192.168.150.16:/data                      24011   Y       4384
NFS Server on localhost                         38467   Y       3268
NFS Server on 192.168.150.16                    38467   Y       4390
NFS Server on 192.168.150.19                    38467   Y       2501
```

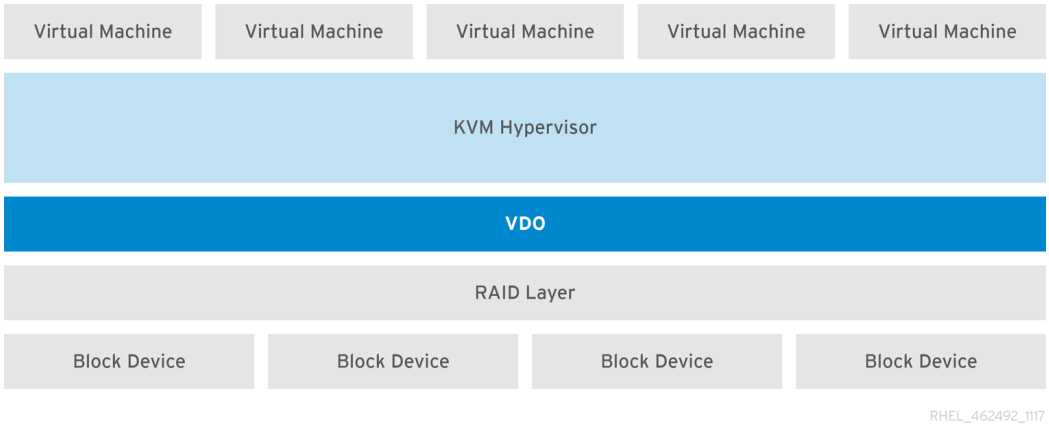
19. 성능튜닝

```
gluster> volume set data performance.write-behind-window-size 1024MB
gluster> volume set data  performance.cache-size 512MB
```

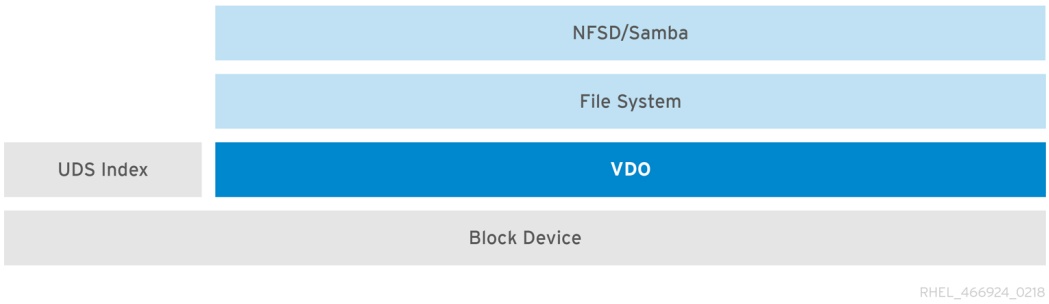
RHEL환경에서 VDO 사용하기

VDO 소개

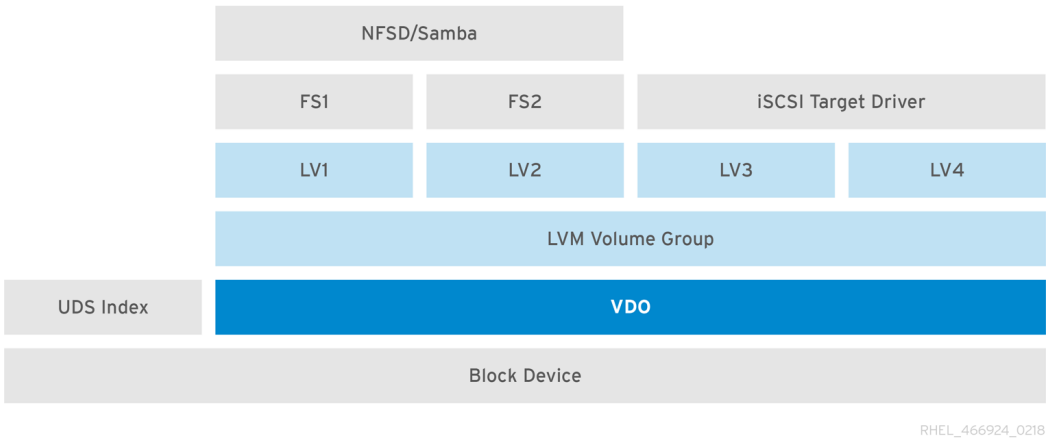
1. VDO; Virtual Data Optimizer 기술은 스토리지의 활용을 증가시키기 위해 데이터 중복제거, 압축 기능을 제공하는 스토리지 기술
2. 활용 목적에 따른 아키텍쳐
    1. VM호스트 기반의 아키텍쳐



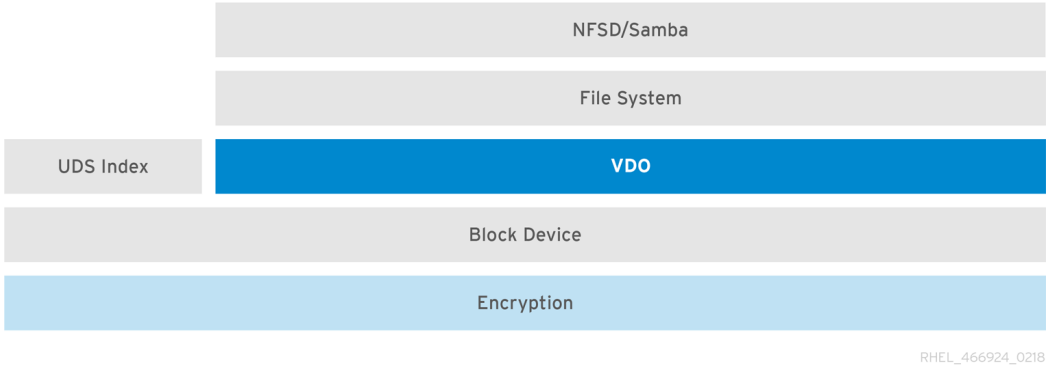
    블록 디바이스 상단에 VDO 디스크 생성 후 하이퍼바이져를 통해 VM image 저장
    2. NFS 활용한 VDO 아키텍쳐



    VDO디스크 생성 후 파일시스템 생성. NFS나 samba로 export 수행
    3. VDO기반의 LVM구성



    VDO디스크 생성 후 LV생성 후 파일시스템 생성. 필요시 NFS / samba로 export 수행
    4. 암호화 적용시 VDO구성



암호화 적용시 VDO하단에 암호화 알고리즘이 적용되어야 하기 때문에 중복제거의 큰 효과는 없음

3. 스토리지 스택별 VDO 구성
    1. VDO 아래 stack에 배치할 component
        1. Software RAID / DM Multi path
    2. VDO 상단 stack
        1. LVM cache / snapshot / thin provisioning
    3. unsupport stack
        1. VDO 상단에 다른 볼륨 구성, LVM 상단에 구성되는 VDO,
4. 장점
    1. 공간활용도 증가
        1. VM호스트 머신으로 운영시 물리/논리의 용량 비율은 약 1:10 비율로 활용 가능
           (1TB디스크로 중복제거와 압축 기능을 화용해 10TB까지 사용 가능)
        2. Ceph기반의 경우 1:3 비율로 용량 활용 가능
5. 단점
    1. 암호화 / 중복제거 적용시 I/O 성능 저하

시스템 요구사항

1. 메모리
    1. VDO모듈 (실제로 점유하는 메모리 용량은 i + ii + iii)
        1. 38MB의 고정 메모리 할당 + 블록 맵 크기 150MB 이상 필요
        2. logical size 1TB당 1.6MB 할당
        3. Physical size 1TB 당 268MB 할당
    2. UDS Size
        1. 기본 250MB 필요
        2. 인덱스 유형별 메모리

인덱스 유형	중복제거용 용량	비 고
Dense	1GB메모리당 당 1TB디스크	일반적으로 4TB 물리디스크에 1GB 인덱스 구성도 충분
Sparse	1GB메모리당 당 10TB디스크	권장 모드, 일반적으로 40TB 물리디스크에 1GB 인덱스 구성도 가능

2. 스토리지
    1. physical size는 최대 256TB 까지 구성 가능
    2. VDO 메타데이터와 UDS 인덱스 구성용 스토리지 구성
    3. 메타 데이터는 스토리지 4GB당 1MB를 저장하고, slabe당 1MB씩 추가로 저장
    4. 일반적으로 UDS 인덱스 구성시 dense는 1GB 메모리가 17GB스토리지를 사용, sparse는 170GB 스토리지를 사용

컴포넌트 리스트

1. kvdo : VDO를 사용하기 위한 커널 모듈
2. uds : VDO기반에서 인덱싱 후 중복데이터를 분석하는 커널 모듈, 새로운 데이터 저장시 이전에 저장된 데이터와 동일한지 검색 후 동일한 데이

터를 두번 저장하지 않도록 참조 구성
3. cli : 스토리지 구성 및 관리 수행

스토리지 용량 관리

1. physical size : VDO를 사용하기 위한 블록장치, 메타데이터 크기를 뺀 값만큼의 용량을 사용
2. logical size : VDO볼륨으로 셍성된 크기, 일반적으로는 physical size보다 크게 설정하면 되나, 기본값은 1:1 비율로 생성, logical 최대 사이즈는 4PB까지 사용가능
3. slab size : VDO볼륨에서 여러개 slab으로 분할 관리. 기본 slab크기는 2GB씩 최대 8192개의 Slab이 생성될 수 있음
   physical size별 권장 slab 크기

Physical Size	Slab Size
10 ~ 99GB	1GB
100GB ~ 1TB	2GB
2 ~ 256TB	32GB

slab사이즈는 vdo 생성시 --vdoSlabSize={{ size }} 옵션으로 생성 가능

VDO 구축

1. VDO 모듈 설치

```
$ yum install vdo kmod-kvdo
$  lsmod  | grep vdo
kvdo               577536  0
uds                253952  1 kvdo
dm_mod             151552  12 kvdo,dm_thin_pool,dm_bufio
```

2. VDO 볼륨 생성

```
$ vdo create --name= {{ VDO 이름 }} --device={{ 디스크 경로 }} --vdoLogicalSize= {{ logical siz }}
```

3. 파일시스템 생성
   #xfs로 생성시

```
$ mkfs.xfs -K /dev/mapper/{{ VDO 이름 }}
```

#ext4로 생성시

```
# mkfs.ext4 -E nodiscard /dev/mapper/{{ VDO 이름 }}
```

4. /etc/fstab에 파일시스템 마운트
   # xfs로 마운트시

```
$ /dev/mapper/{{ VDO 이름 }} {{ 마운트 경로 }} xfs defaults,x-systemd.device-timeout=0,x-systemd.requires=vdo.service 0 0
```

#ext4로 마운트시

```
$ /dev/mapper/{{ VDO 이름 }} {{ 마운트 경로 }} ext4 defaults,x-systemd.device-timeout=0,x-systemd.requires=vdo.service 0 0
```

VDO 운영

1. VDO 볼륨 관리

```
$ vdo start --name= {{ VDO 이름 }} or vdo start --all
$ vdo stop --name= {{ VDO 이름 }} or vdo stop --all
```

2. VDO 볼륨 활성화
   #볼륨 활성화

```
$ vdo active --name {{ VDO 이름 }} or vdo active --all
```

   \\#볼륨 비활성화

```
$ vdo deactivate --name {{ VDO 이름 }} or vdo deactive ~-~-all
```

   기본적으로 OS부팅시 VDO볼륨 활성화 수행 (vdo create 시 ~-~-activate=disabled 옵션을 추가하면 자동 활성화 안함)
3. vdo볼륨 제거

```
$> vdo remove --name {{ VDO 이름 }} or vdo remove --all
```

4. VDO에서 사용하지 않는 블록 삭제

```
$ systemctl enable --now fstrim.timer
```

5. VDO 사용현황 확인

```
$ vdostats ~-~-human-readable
\\Device                 1K-blocks    Used     Available    Use%    Space saving%
/dev/mapper/node1osd1    926.5G       21.0G    905.5G       2%      73%
/dev/mapper/node1osd2    926.5G       28.2G    898.3G       3%      64%
```

6. vdo 볼륨 크기 증가

```
$> vdo growLogical --name={{ VDO 이름 }} --vdoLogicalSize= {{ 변경할 크기 }}
```

reference

1. *https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/storage_administration_guide/vdo-quick-start*
2. *https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/deduplicating_and_compressing_storage/deploying-vdo_deduplicating-and-compressing-storage*