Enterprise ZFS NVMe storage server

21.02.25

Pegasi Knowledge
https://ghost.pegasi.fi/wiki/

Table of Contents

Enterprise ZFS NVMe storage server

Hold on, document is not finished yet. I will finish it as I do the solution so it will be ready as soon as everything is rolling.

ZFS is a powerful, enterprise grade file system containing a collection of innovative technologies that make a powerful service to use for your file backend needs. This document describes how to set up an Almalinux 8 (RHEL/Centos clone) based ZFS storage server cluster using NVME storage media with network transport using high speed NVME-of and NFS over Infiniband RDMA networks.

I am using two storage nodes and multiple front end hypervisor nodes and other clients. Storage network is connected with 56Gbps Infiniband and we use RDMA NVMEof to communicate within the storage cluster and NFS over RDMA to the client network. Storage servers are Supermicro dual AMD Epyc based boxes with 64 cores, 32G RAM and 24 U.2 NVME slots.

It has been a bit challenging to get the performance to a satisfiable level but it seems I am there now. Last time i checked I could write over 4000 mbytes / second over NVME-of share to the ZFS which is a good one with ZFS.

I would like to use zvols to export block devices to my hypervisors but I've read there might be some problems with zvols and snapshots so I will also compare zvols via NVME-of vs qcow2 via NFS RDMA.

Overview

I have 3 NVME drives on each of the two Almalinux 8 (RHEL, Centos) servers for the zpool usage. The servers will see each other's NVME devices through NVME over fabric. In the primary server I do a striped mirrored (raid10) zpool so that I have equivalent devices mirrored on each host so that one host can fail and the pool data is still usable. It looks like this:

```
root@datavault]# zpool status
  pool: datavault
 state: ONLINE
  scan: resilvered 372K in 00:00:00 with 0 errors on Sun Aug  1 17:57:56
2021
config:

        NAME                                                STATE     READ
WRITE CKSUM
        datavault                                           ONLINE       0
0     0
          mirror-0                                          ONLINE       0
0     0
            nvme-INTEL_SSDPE2KE016T8O_PHLN034201L01P6AGN    ONLINE       0
0     0
```

```
            nvme-uuid.097cfca0-5ada-4652-a526-d8e6f29f34e4  ONLINE       0
0     0
          mirror-1                                          ONLINE       0
0     0
            nvme-INTEL_SSDPE2KE016T8O_PHLN034201DS1P6AGN    ONLINE       0
0     0
            nvme-uuid.5f4cf7e9-6057-417c-8f35-58555d86c27d  ONLINE       0
0     0
          mirror-2                                          ONLINE       0
0     0
            nvme-INTEL_SSDPE2KE016T8O_PHLN034301L31P6AGN    ONLINE       0
0     0
            nvme-uuid.1c81d611-7516-4492-a5a0-c91bbb7cf845  ONLINE       0
0     0
```

On tops of this I use another NVME-of subsystem to export the ZFS zvols and RDMA NFS to export the ZFS datasets for QCOW2 and other file needs.

Prequisites

Check out the Infiniband RDMA native setup for Linux document on how to set up your IB and NVME-of environment. You must set up NVME target and client to both storage nodes and NVME clients to each hypervisor nodes if you are going to use zvol block devices.

I am using Almalinux 8 but this should be applicable to any modern Linux distro.

Hardware / OS settings

Target tuning

These settings gave me over 4 x performance boost.

BIOS settings for Intel and AMD

- Intel: Hyperthreading maybe better disabled. Try and compare.
- AMD: SVM (Virtualization) disabled

BIOS settings for AMD EPYC systems

- Global C-state Control disabled
- Hyperthreading disabled
- IOMMU disabled
- SR-IOVand disabled

- Core Performance Boost disabled
- Determinism Slider set to Performance
- Memory Interleaving to Auto

EPYC iSER seems to work a lot better with these set to /etc/default/grub kernel options

```
iommu=ptcpuidle.off=1 processor.max_cstate=0
```

Accoring to the writer in https://openzfs.github.io/openzfs-docs/Getting%20Started/RHEL-based%20distro/index.html the below settings combo might be good. Did not yet but here it is, for future reference.

```
pcie_aspm=off rcu_nocbs=1-63 amd_iommu=on iommu=pt mitigations=off
elevator=noop
```

And update grub.

```
grub2-mkconfig -o /boot/grub2/grub.cfg
sync
reboot
```

Initiator tuning

- Disable hyperthreading
- set /etc/modprobe.d/ib*iser.conf: options ib*iser always_register=N

For hypervisors

```
dnf in tuned
systemctl enable tuned --now
tuned-adm list
tuned-adm profile virtual-host
```

Common settings

Firewalld

```
firewall-cmd --new-zone=nvmeof --permanent
firewall-cmd --reload
firewall-cmd --zone=nvmeof --add-source=1.2.3.4/24 --permanent
firewall-cmd --zone=nvmeof --add-source=1.2.3.5/24 --permanent
firewall-cmd --zone=nvmeof --add-port=4420/tcp --permanent
firewall-cmd --reload
```

Update. Kernel must be updated for the ZFS DKMS module compilation to work.

```
dnf up
```

Install software

```
dnf in nvmet
dnf in rdma-core smartmontools

dnf in https://zfsonlinux.org/epel/zfs-release.el8_4.noarch.rpm
rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-zfsonlinux
dnf in epel-release
dnf in kernel-devel zfs
```

Configure modules to be loaded already so we have all modules up and running after reboot.

```
echo "nvme-rdma" > /etc/modules-load.d/nvme.conf
echo "nvmet-rdma" > /etc/modules-load.d/nvme.conf
echo zfs >/etc/modules-load.d/zfs.conf
```

Set the modules we want to load automatically by setting the file /etc/rdma/rdma.conf like this

```
# Load IPoIB
IPOIB_LOAD=yes
# Load SRP (SCSI Remote Protocol initiator support) module
SRP_LOAD=no
# Load SRPT (SCSI Remote Protocol target support) module
SRPT_LOAD=no
# Load iSER (iSCSI over RDMA initiator support) module
ISER_LOAD=yes
# Load iSERT (iSCSI over RDMA target support) module
ISERT_LOAD=yes
# Load RDS (Reliable Datagram Service) network protocol
RDS_LOAD=no
# Load NFSoRDMA client transport module
XPRTRDMA_LOAD=no
# Load NFSoRDMA server transport module
SVCRDMA_LOAD=yes
# Load Tech Preview device driver modules
TECH_PREVIEW_LOAD=no
```

ZFS installation

Create /etc/modprobe.d/zfs:

```
options zfs zfs_arc_max=4294967296
options zfs zvol_threads=1
```

Do a reboot and continue below.

NVME-of target configuration

Since IB interfaces get their IP addresses a bit slower we must delayt the NVME initializations. Modify /usr/lib/systemd/system/nvmet.service and set “After” line to this:

```
After=sys-kernel-config.mount network.target local-fs.target NetworkManager-wait-online.service
```

Set a delay under [Service]:

```
ExecStartPre=/bin/sleep 40
```

Create an NVMET subsystems for internal storage pool and shared devices.

```
mkdir /sys/kernel/config/nvmet/subsystems/datavault01
```

Next we will do a subsystems for both nodes to export them to each other. I am using config filesystem but you can do this with nvmetcli as well.

```
cd /sys/kernel/config/nvmet/subsystems

mkdir datavault01
echo 1 > datavault01/attr_allow_any_host
cd datavault01/namespaces
mkdir 10 11 12
echo -n /dev/disk/by-id/nvme-INTEL_SSDPE2KE016T8O_xxx > 10/device_path
echo -n /dev/disk/by-id/nvme-INTEL_SSDPE2KE016T8O_xxx > 11/device_path
echo -n /dev/disk/by-id/nvme-INTEL_SSDPE2KE016T8O_xxx > 12/device_path
echo 1 > 10/enable
echo 1 > 11/enable
echo 1 > 12/enable

cd /sys/kernel/config/nvmet/ports
mkdir 1

echo 10.0.1.1 > 1/addr_traddr
echo ipv4 > 1/addr_adrfam
echo rdma > 1/addr_trtype
echo 4420 > 1/addr_trsvcid
```

```
ln -s /sys/kernel/config/nvmet/subsystems/datavault01 1/subsystems/

nvmetcli save
```

NVME RDMA client configuration

Modify /usr/lib/systemd/system/nvmf-autoconnect.service and set a delay under [Service]:

```
ExecStartPre=/bin/sleep 40
```

Then run “systemctl daemon-reload”.

We need to mount the nvmet exported drivers to storage nodes to create the RAID10 zpool. You can also install these on the hypervisor or other client nodes.

```
nvme discover -t rdma -a 10.0.1.2 -s 4420
nvme connect -t rdma -n datavault02 -a 10.0.1.2 -s 4420
systemctl enable nvmf-autoconnect
echo "-t rdma -a 10.0.1.2 -s 4420" >> /etc/nvme/discovery.conf
```

Autoconnect wants to start connecting before IB interface gets IP addresses. Modify /usr/lib/systemd/system/nvmf-autoconnect.service and set a delay under [Service]:

```
ExecStartPre=/bin/sleep 40
```

Then run “systemctl daemon reload”.

ZFS configuration

Our intention is to create RAID10 pool, a striped pool that consists of striped mirrored pairs. Every mirror pair consists of a NVME drive from storage node 01 and 02. This way one of the hosts or drives may fail and we are still operational.

Now try that you we see our devices and mounted devices with these commands:

```
# nvme list-subsys
nvme-subsys0 - NQN=nqn.2014.08.org.nvmexpress:80868086PHLN034201L01P6AGN
INTEL SSDPE2KE016T8O
\
 +- nvme0 pcie 0000:07:00.0 live
nvme-subsys1 - NQN=nqn.2014.08.org.nvmexpress:80868086PHLN034201DS1P6AGN
INTEL SSDPE2KE016T8O
\
```

```
 +- nvme1 pcie 0000:08:00.0 live
nvme-subsys2 - NQN=nqn.2014.08.org.nvmexpress:80868086PHLN034301L31P6AGN
INTEL SSDPE2KE016T8O
\
 +- nvme2 pcie 0000:09:00.0 live
nvme-subsys3 - NQN=datavault02
\
 +- nvme3 rdma traddr=10.0.1.2 trsvcid=4420 live

# nvme list
Node             SN                   Model
Namespace Usage                      Format           FW Rev
---------------- -------------------- ----------------------------------------
-- --------- -------------------------- ---------------- --------
/dev/nvme0n1     PHLN034201L01P6AGN   INTEL SSDPE2KE016T8O
1           1.60  TB /   1.60  TB      4 KiB +  0 B   VDV10152
/dev/nvme1n1     PHLN034201DS1P6AGN   INTEL SSDPE2KE016T8O
1           1.60  TB /   1.60  TB      4 KiB +  0 B   VDV10152
/dev/nvme2n1     PHLN034301L31P6AGN   INTEL SSDPE2KE016T8O
1           1.60  TB /   1.60  TB      4 KiB +  0 B   VDV10152
/dev/nvme3n1     df26ad0a7f1434ae     Linux
20          1.60  TB /   1.60  TB      4 KiB +  0 B   4.18.0-3
/dev/nvme3n2     df26ad0a7f1434ae     Linux
21          1.60  TB /   1.60  TB      4 KiB +  0 B   4.18.0-3
/dev/nvme3n3     df26ad0a7f1434ae     Linux
22          1.60  TB /   1.60  TB      4 KiB +  0 B   4.18.0-3
```

Then it is time to add them to zpool. Lets use the device IDs so we will not get messed up if/when the /dev/nvmeXnY numberings decide to change. Please note that the device id is also visible in the target /sys/kernel/config/nvmet/subsystems/<subsystem>/namespaces/<namespaceid>/device_uuid.

Check allowed block sized from your NVME device and set it:

```
smartctl -a /dev/nvme0n1
```

Locate “supported LBA sizes” and check the largest one supported. I have 4096. Now lets format the nvme devices to 4096 to increase ZFS performance. LBA block size “1” means 4096 so lets use that to get the desired 4096 block size.

```
nvme format /dev/nvme0n1 -l 1
```

Now lets create the pool with the 4096 blocksize. This time we must use “ashift=12” with zpool create command to get the 4096 block size. Sometimes the tech guys could think of us normal mortals when implementing these values.

```
zpool create -o ashift=12 datavault \
mirror /dev/disk/by-id/nvme-INTEL_SSDPE2KE016T8O_PHLN034201L01P6AGN
```

```
/dev/disk/by-id/nvme-uuid.e91943dc-8251-4abf-b8e8-e82db16aa243 \
mirror /dev/disk/by-id/nvme-INTEL_SSDPE2KE016T8O_PHLN034201DS1P6AGN
/dev/disk/by-id/nvme-uuid.53eb36b9-40f9-4d46-91fd-16d4f427f554 \
mirror /dev/disk/by-id/nvme-INTEL_SSDPE2KE016T8O_PHLN034301L31P6AGN
/dev/disk/by-id/nvme-uuid.c14cb835-bc70-4707-9dba-0e19a19f45a9
```

To create NVME friendly zvols try command:

```
zfs create -o primarycache=metadata -s -V 80G datavault/<zvol name>
```

For datasets for QCOW2 files or other try:

```
zfs create -o primarycache=metadata datavault/<dataset name>
```

And check what “zpool status” and “zfs list” is telling you.

iSER target configuration

I will run this thru very fast. Please look more detailed instructions in the Infiniband setup article.

```
firewall-cmd --new-zone=iscsi --permanent
firewall-cmd --reload
firewall-cmd --zone=iscsi --add-port=3260/tcp --permanent
firewall-cmd --zone=iscsi --change-interface=ib0 --permanent
firewall-cmd --reload
firewall-cmd --zone=iscsi --list-all
```

Targetcli.

```
dnf in targetcli
systemctl enable target --now
targetcli
/iscsi> create iqn.2021-08.fi.pegasi:vaultname
/> iscsi/
/iscsi> create iqn.2021-08.fi.pegasi:vaultname
/iscsi> /backstores/block
/backstores/block> create name=host.pegasi.fi
dev=/dev/storagegroup01/host.pegasi.fi
/backstores/block> /iscsi/iqn.2021-08.fi.pegasi:vaultname/tpg1/portals
delete 0.0.0.0 3260
/backstores/block> /iscsi/iqn.2021-08.fi.pegasi:vaultname/tpg1/portals
create 10.0.0.1 3260
/backstores/block>
/iscsi/iqn.2021-08.fi.pegasi:vaultname/tpg1/portals/10.0.0.1:3260
enable_iser boolean=true
```

```
/> /iscsi/iqn.2021-08.fi.pegasi:vaultname/tpg1/luns
/> create /backstores/block/host.pegasi.fi
/> /iscsi/iqn.2021-08.fi.pegasi:vaultname/tpg1/acls create
iqn.2021-08.fi.pegasi:hypervisor01
/> /iscsi/iqn.2021-08.fi.pegasi:vaultname/tpg1/acls create
iqn.2021-08.fi.pegasi:hypervisor02
/> /iscsi/iqn.2021-08.fi.pegasi:vaultname/tpg1/acls create
iqn.2021-08.fi.pegasi:hypervisor03
/> saveconfig
```

NVME target configuration

I will not use this with the client / hypervisor. This will only serve as backend storage cross-mounts to give me RAID1 arrays. But if you wish to create exports with this then pleace proceed.

We will do a subsystem called “datavault” per node (nvmet port 1) for zvol and NFS RDMA sharing to clients. We will use the floating service IP for the datavault subsystem. This will serve clients.

```
cd /sys/kernel/config/nvmet/subsystems

mkdir datainternal01
cd datainternal01/namespaces
mkdir 10 11 12
echo -n /dev/disk/by-id/nvme-INTEL_SSDPE2KE016T8O_xxx > 10/device_path
echo -n /dev/disk/by-id/nvme-INTEL_SSDPE2KE016T8O_xxx > 11/device_path
echo -n /dev/disk/by-id/nvme-INTEL_SSDPE2KE016T8O_xxx > 12/device_path
echo 1 > 10/enable
echo 1 > 11/enable
echo 1 > 11/enable

cd /sys/kernel/config/nvmet/ports
echo 10.0.0.10 > 1/addr_traddr
echo ipv4 > 1/addr_adrfam
echo rdma > 1/addr_trtype
echo 4420 > 1/addr_trsvcid
ln -s /sys/kernel/config/nvmet/subsystems/datavault 1/subsystems/

nvmetcli save
```

ZVOL exports to hypervisors and other usage

After ZFS pool creation we can add the devices to nvmet namespaces.

```
mkdir datavault
```

```
echo 1 > datainternal01/attr_allow_any_host
mkdir 50
echo -n /dev/zvol/datavault/virtual.guest.1 > 50/device_path
echo 1 > 50/enable
```

NFS over RDMA

NFS server

NFS over RDMA is one way to bring files and directories to clients. I am considering it over NVME-of block device export. But it is not that well performing.

Set in /etc/nfs.conf:

```
rdma=y
rdma-port=20049
```

```
dnf in nfs-utils
systemctl enable --now nfs-server rpcbind
```

Reboot and test that NFS RDMA is active by verifying that file /proc/fs/nfsd/portlist has the RDMA port included. You can also say:

```
echo rdma 20049 > /proc/fs/nfsd/portlist
```

To activate RDMA port immediately. Next set exports to /etc/exports:

```
/datavault/mydataset 10.0.0.0/24(rw,sync)
```

And export them with command:

```
exportfs -arv
```

NFS client

```
dnf in nfs-utils
```

Try mounting with command:

```
mount -t nfs -o rdma,port=20049 10.0.0.1:/datavault/images /mnt/images
```

Add to /etc/fstab the mountpoint you wish to use:

```
10.0.0.10:/datavault/live-images       /mnt/live-images        nfs
rdma,port=20049 0 0
```

Manual Cluster configuration

First lets do manual clustering piece by piece to proof what things need to happen. This makes automatic cluster configuration more familiar.

I am not sure if I need to disable NVMET port with service IP since it is not appearing in TCP stack and it does not produce any error messages when I remove the IP with nmvli but for sureness sake lets keep it in line. I am not sure what happens in RDMA world where there is something trying to listen to the same IP address in the same subnet.

Node tasks for starting and stopping a cluster node

These are tasks that need to be done to switch over the service from node to another.

Active node tasks

These are tasks that must be done automatically by the cluster software or manually to make a node active. Add service IP, import pool, add exports to iSER and activate iSER service IP port.

```
nmcli con mod ib0 +ipv4.addresses "10.0.0.100/24"
nmcli device reapply ib0
systemctl reload firewalld
zpool import -d /dev/disk/by-id/ datavault
echo 1 > /sys/kernel/config/nvmet/subsystems/datavault/namespaces/50/enable
echo 1 > /sys/kernel/config/nvmet/subsystems/datavault/namespaces/NN/enable
ln -s /sys/kernel/config/nvmet/subsystems/datavault
/sys/kernel/config/nvmet/ports/1/subsystems/
```

Passive node tasks

Tasks that must be done by the node about to become passive so that another node can take over.

```
rm -f /sys/kernel/config/nvmet/ports/1/subsystems/datavault
echo 0 > /sys/kernel/config/nvmet/subsystems/datavault/namespaces/50/enable
echo 0 > /sys/kernel/config/nvmet/subsystems/datavault/namespaces/NN/enable
zpool export datavault
nmcli con mod ib0 -ipv4.addresses "10.0.0.100/24"
nmcli device reapply ib0
```

```
systemctl reload firewalld
```

Node tasks for NVMEoF-only

Again please discard if you are using iSER for exporting devices to frontend hypervisors and clients.

Active node tasks

These are tasks that must be done automatically by the cluster software or manually to make a node active. Add service IP, import pool, add exports to NVMET and activate NVMET service IP port.

```
nmcli con mod ib0 +ipv4.addresses "10.0.0.100/24"
nmcli device reapply ib0
systemctl reload firewalld
zpool import -d /dev/disk/by-id/ datavault
echo 1 > /sys/kernel/config/nvmet/subsystems/datavault/namespaces/50/enable
echo 1 > /sys/kernel/config/nvmet/subsystems/datavault/namespaces/NN/enable
ln -s /sys/kernel/config/nvmet/subsystems/datavault
/sys/kernel/config/nvmet/ports/1/subsystems/
```

Passive node tasks

Tasks that must be done by the node about to become passive so that another node can take over.

```
rm -f /sys/kernel/config/nvmet/ports/1/subsystems/datavault
echo 0 > /sys/kernel/config/nvmet/subsystems/datavault/namespaces/50/enable
echo 0 > /sys/kernel/config/nvmet/subsystems/datavault/namespaces/NN/enable
zpool export datavault
nmcli con mod ib0 -ipv4.addresses "10.0.0.100/24"
nmcli device reapply ib0
systemctl reload firewalld
```

Useful commands for the future

Bring offline devices online

If you lose on storage host or a disk goes offline you may and up with degraded status of the drive. If the host / drive is back online you can simply give a command to bring it back:

```
zpool online datavault nvme-uuid.1c81d611-7516-4492-a5a0-c91bbb7cf845
```

Replace drives in RAID10 vdevs

This is not so evident in the documentation and some say this cannot be done so for clarity's sake I will document this here. If you happen to create a new NVMET configuration and disk UUIDs change you will have a degraded but functional zpool. Also your “new” drives will be identical to the existing ones (because they are the same disks) which makes the zpool confused and normal replace commands will not work. To replace the drives we need to remove and add them drive back using new UUIDs.

You could edit the identical UUIDs to the other node /sys/kernel/config/nvmet/subsystems/datainternalXX/namespaces/NN/device_uuid and it would fix the situation but here goes anyway. This is good to know.

First check pool status if it is aware of the changed/dissappeared NVMEof disks:

```
zpool status
```

If disks are all online it means your zpool possibly still sees the same /dev/nvmeXXXX devices but the UUIDs have changed and it is a problem waiting to escalate. Verify this by checking if the /dev/disk/by-id/nvme-uuid* values match or not. If not it means the device UUIDs have changed you must nudge the spool for it to notice the situation:

```
zpool resilver datavault
```

Now check status again and it will show UNAVAIL drives. Now we will detach the drives from mirror vdevs and add new (same) ones to replace them. We use the functional, local drives at VDEVs to identify the mirror VDEVs to attach the replacement drives to. There are 3 mirror VDEVs, of which each only has a single functional disk, the local one. By attaching a disk using the online disk id as the first identifier we create a new mirror drive for that disk.

```
zdb #look for GUID of the the failed drives
zpool detach datavault 505931440553767481 #detach the drive
zpool status #observe that the mirror has been replaced by the single drive
left, which we use to create a new mirror next
zpool attach datavault nvme-INTEL_SSDPE2KE016T8O_PHLN034201L01P6AGN
/dev/disk/by-id/nvme-uuid.f40660f3-7013-4fe3-aa99-5f77cbda0f5f
zpool status #now you should have mirror again with online drives - repeat
for other drives
```

Move zpool to another cluster node manually

In case you need to relocate the pool to another cluster first disable all zvol / zfs exports and export zpool from the old node:

```
echo 0 > /sys/kernel/config/nvmet/subsystems/datavault/namespaces/50/enable
```

```
zpool export datavault
```

Then import + enable in the new node. It is important to import with disk IDs because the /dev/nvme* names are not static.

```
zpool import -d /dev/disk/by-id datavault
echo 1 > /sys/kernel/config/nvmet/subsystems/datavault/namespaces/50/enable
```

Also remember to relocate the service IP address.