High Availability: How to Install Lustre with ZFS and HA for OSS and MDS Nodes

Prerequisites:

- 2 OSS nodes connected via SAS over multiple paths to their own JBOD shared storage (minimum eight drives)
- 2 MDS nodes connected via SAS over multiple paths to their own JBOD shared storage (minimum four drives)
- Git is installed on the head node

Requirements:

- The Bright version is 8.0.
- The MDS nodes must be setup for active-passive high availability.
- The backend storage for the MDS nodes will be software RAID 10.
- The OSS nodes must be setup for active-active high availability.
- The backend storage for the OSS nodes will be ZFS raidz2 (RAID 6) pools (at least one pool for each node).

Create MDS software image

1. Clone the default-image to initialize the MDS software image.

```
cmsh
% softwareimage
% clone default-image lustre-mds-image
% commit
```

2. Clone the lustre-release git repository onto the head node:

```
cd /root
git clone git://git.hpdd.intel.com/fs/lustre-release.git
```

3. Use the cloned repository to create the RPMs required for the MDS nodes:

```
cd /root/lustre-release
sh ./autogen.sh
./configure
make rpms
```

4. Install the new RPMs onto the MDS software image:

```
yum --installroot=/cm/images/lustre-mds-image localinstall *.rpm
```

5. Set the software image's kernelversion to the newly installed lustre kernel:

```
cmsh -c
"softwareimage use lustre-mds-image; set kernelversion <new-lustre-ker
nel>; commit"
```

6. Install device mapper multipath onto image:

```
yum --installroot=/cm/images/lustre-mds-image install
 device-mapper-multipath
```

Create category for MDS nodes

```
cmsh
% category
% clone default lustre-mds
% set softwareimage lustre-mds-image
% set installbootrecord yes
% clear roles
% commit
% device -n mds01..mds02 (set category lustre-mds; commit)
```

Now provision and boot the two MDS nodes.

Setup MDS backend storage

1. Log into mds01.

2. Setup multipath:

```
mpathconf --enable --with_multipathd y
```

3. Copy /etc/multipath.conf and /etc/multipath/bindings to mds02 so that the mappings are consistent:

```
scp /etc/multipath.conf mds02:/etc/multipath.conf
scp /etc/multipath/bindings mds02:/etc/multipath/bindings
```

4. Create software RAID 10 volume using multipath mappings and save its configuration to file:

```
mdadm --create /dev/md0 --level=10 --raid-devices=4 /dev/mapper/mpath[
a-d]
mdadm --detail --scan --verbose >> /etc/mdadm.conf
scp /etc/mdadm.conf mds02:/etc/mdadm.conf
```

5. Create the mgs and mdt using /dev/md0:

```
mkfs.lustre --fsname lustre00 --mdt --mgs --failnode=<NID-of-mds02> /
dev/md0
```

If Lustre is to communicate solely over Infiniband and if there are multiple types of network interfaces on the Lustre server and client nodes, then add the following modprobe configuration to each of those nodes; otherwise, skip to step 6.

```
# cat /etc/modprobe.d/lustre.conf
options lnet networks=o2ib
```

Because Lustre may actually choose to use Ethernet for its NIDs, the above configuration is necessary to force Lustre to only use Infiniband. After adding the configuration, the Lustre nodes will need to be rebooted.

6. Mount the mdt:

```
mkdir /mnt/mdt; mount -t lustre /dev/md0 /mnt/mdt
```

Now go back to the head node.

Create failovergroup for MDS nodes

1. Update the software image for the MDS nodes:

```
cmsh -c "device use mds01; grabimage -w"
```

2. Create the Bright failover group:

```
cmsh
% partition use base
% failovergroups
% add mdsgroup
% set nodes mds01..mds02
% set automaticfailoveraftergracefulshutdown yes
% set mountscript <path-to-script-with-contents-shown-below>
% set unmountscript <path-to-script-with-contents-shown-below>
% commit
```

Here are the contents of the mount script, which will be used when a passive MDS becomes active:

```
#!/bin/sh

/bin/systemctl start multipathd.service || { echo
"Multipath not started." ; exit 1; }
/sbin/mdadm --assemble --scan || { echo "Software RAID not started."
 ; exit 1; }
/bin/mount /mnt/mdt || { echo "Cannot mount mdt." ; exit 1; }
```

Here are the contents of the unmount script, which will be used when an active MDS becomes passive:

```
#!/bin/sh

/bin/umount -f /mnt/mdt || { echo "Unmount failed." ; exit 1; }
/sbin/mdadm --stop --scan || { echo "Software RAID not stopped." ;
exit 1; }
/bin/systemctl stop multipathd.service || { echo
"Multipath not stopped." ; exit 1; }
/sbin/multipath -F || { echo "Paths not flushed." ; exit 1; }
```

3. Add the multipathd service to the MDS category. It will only be run on the active MDS:

```
% category use lustre-mds
% services
% add multipathd
% set runif active
% set autostart yes
% set monitoring yes
% commit
```

Create software images for OSS nodes

1. Clone the default-image to initialize the software image for oss01:

```
cmsh
% softwareimage
% clone default-image lustre-oss01-image
% commit
```

2. Use the lustre-release git repository to create the RPMs for the OSS nodes:

```
cd /root/lustre-release
make clean
rm -f *.rpm
sh ./autogen.sh
./configure --disable-ldiskfs
make rpms
```

3. Install the new RPMs onto the oss01 software image:

```
yum --installroot=/cm/images/lustre-oss01-image localinstall *.rpm
```

4. Set the software image's kernelversion to the newly installed lustre kernel:

```
cmsh -c
"softwareimage use lustre-oss01-image; set kernelversion <new-lustre-k
ernel>; commit"
```

5. Install device mapper multipath onto image:

```
yum --installroot=/cm/images/lustre-oss01-image install
 device-mapper-multipath
```

6. Install zfs onto image and enable the appropriate services for boot:

```
yum --installroot=/cm/images/lustre-oss01-image install http://
download.zfsonlinux.org/epel/zfs-release.el7_3.noarch.rpm
```

Enable the [zfs-kmod] repository in /cm/images/lustre-oss01-image/etc/yum.repos.d/zfs.repo.

```
yum --installroot=/cm/images/lustre-oss01-image install
 zfs libzfs2-devel kmod-spl-devel kmod-zfs-devel
chroot /cm/images/lustre-oss01-image
systemctl enable
 zfs-import-scan zfs-mount zfs-share zfs-zed zfs.target
systemctl disable zfs-import-cache
```

Modify /lib/systemd/system/zfs-import-scan.service to have the following contents:

```
[Unit]
Description=Import ZFS pools by device scanning
DefaultDependencies=no
Requires=systemd-udev-settle.service
After=systemd-udev-settle.service
After=cryptsetup.target
ConditionPathExists=!/etc/zfs/zpool.cache

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStartPre=/sbin/modprobe zfs
ExecStart=/usr/local/bin/zpool-import.sh
ExecStop=/usr/local/bin/zpool-export.sh

[Install]
WantedBy=zfs-mount.service
WantedBy=zfs.target
```

The /usr/local/bin/zpool-import.sh script will import a specified set of zpools on each OSS node,

and that set of zpools will be different on each. This is the normal HA state, which we will call State 0. Create this script to have the following contents in this example:

```
#!/bin/bash

/sbin/zpool import -o cachefile=none ostpool01
```

Create the complementing /usr/local/bin/zpool-export.sh as follows:

```
#!/bin/bash

/sbin/zpool export -f ostpool01
```

7. Clone the oss01 image to create the oss02 image:

```
cmsh -c
"softwareimage clone lustre-oss01-image lustre-oss02-image; commit"
```

Now change /usr/local/bin/zpool-import.sh on lustre-oss02-image as follows:

```
#!/bin/bash

/sbin/zpool import -o cachefile=none ostpool02
```

Create the complementing /usr/local/bin/zpool-export.sh as follows:

```
#!/bin/bash

/sbin/zpool export -f ostpool02
```

Create categories for OSS nodes

```
cmsh
% category
% clone lustre-mds lustre-oss01
% set softwareimage lustre-oss01-image
% services
% add multipathd
% set autostart yes
% set monitoring yes
% ..
% clone lustre-oss01 lustre-oss02
% set softwareimage lustre-oss02-image
% commit
% device use oss01; set category lustre-oss01; commit
% device use oss02; set category lustre-oss02; commit
```

Now provision and boot the two OSS nodes.

Setup OSS backend storage

1. Log into oss01.

2. Setup multipath:

```
mpathconf --enable --with_multipathd y
```

3. Restart multipathd:

```
systemctl restart multipathd
```

4. Copy /etc/multipath/bindings to oss02 for consistency:

```
scp /etc/multipath/bindings oss02:/etc/multipath/bindings
```

5. Back on oss01, create your raidz2 zpools. For this example, the shared storage only has 8 drives so a raidz2 pool of 4 drives will be created:

```
zpool create -fo cachefile=none ostpool01 raidz2 /dev/mapper/mpath[a-d
]
```

6. Create the osts using the created zpools:

```
mkfs.lustre --ost --backfstype=zfs --fsname=lustre00 --index=0
--mgsnode=<NID-of-mds01> --mgsnode=<NID-of-mds02> --servicenode=<
NID-of-oss01> --servicenode=<NID-of-oss02> ostpool01/ost01
zfs set recordsize=1M ostpool01/ost01
```

7. Mount the osts:

```
mkdir /mnt/ost01; mount -t lustre ostpool01/ost01 /mnt/ost01
```

8. On oss02, create the remaining raidz2 pools. For this example, there are only 4 unallocated drives left so there is only one pool left to create:

```
zpool create -fo cachefile=none ostpool02 raidz2 /dev/mapper/mpath[e-h
]
```

9. Create the osts using the created zpools:

```
mkfs.lustre --ost --backfstype=zfs --fsname=lustre00 --index=0
```

```
--mgsnode=<NID-of-mds01> --mgsnode=<NID-of-mds02> --servicenode=<
NID-of-oss01> --servicenode=<NID-of-oss02> ostpool02/ost02
zfs set recordsize=1M ostpool02/ost02
```

10. Mount the osts:

```
mkdir /mnt/ost02; mount -t lustre ostpool02/ost02 /mnt/ost02
```

11. On the head node, create mounts for the osts:

```
cmsh
% category use lustre-oss01
% fsmounts
% add /mnt/ost01
% set device ostpool01/ost01
% set filesystem lustre
% set mountoptions "rw,_netdev"
% category use lustre-oss02
% add /mnt/ost02
% set device ostpool02/ost02
% set filesystem lustre
% set mountoptions "rw,_netdev"
% commit
```

12. Make sure both images are up-to-date:

```
% device use oss01; grabimage -w
% device use oss02; grabimage -w
```

Create failovergroup for OSS nodes

Create the Bright failover group:

```
cmsh
% partition use base
% failovergroups
% add ossgroup
% set nodes oss01..oss02
% set automaticfailoveraftergracefulshutdown yes
% set prefailoverscript <path-to-script-with-contents-shown-below>
% set postfailoverscript <path-to-script-with-contents-shown-below>
% commit
```

The contents of the prefailover script should be:

```
#!/bin/sh

# Do not import zpools if missing multipath mpaths
[ $(/usr/sbin/multipath -ll | grep -c mpath) -ne 8 ] && { echo
"FATAL: Not seeing all multipath mpaths." ; exit 1; }

# Import appropriate zpools on each OSS node
/usr/local/bin/zpool-import.sh || { echo
"FATAL: Cannot import usual zpools." ; exit 1; }
```

NOTE: The 8 multipath paths should be altered in this script to be the actual number of multipaths on your system.

The contents of the postfailover script should be:

```
#!/usr/bin/python

import os
import subprocess

# Read current state
f = open('/var/spool/cmd/state')
state = f.readline()
f.close()

# Get the hostname
```

```
command = subprocess.Popen('hostname', stdout=subprocess.PIPE);
(hostname, err) = command.communicate()
hostname = hostname.rstrip("\n")

# function to check if the counter part is online
def isonline (nodename):


# here we do the ping check to determine whether the other node is up
or down
    response = os.system("ping -c 1 " + nodename);
    if response != 0:
       print "passive node is down";
       # passive node down
       # check if the multipath is correct
       command = subprocess.Popen(
'/usr/sbin/multipath -ll | grep -c mpath' , stdout=subprocess.PIPE,
 stderr=subprocess.PIPE, shell=True);
       (multipathcount, err) = command.communicate()
       multipathcount = multipathcount.rstrip("\n")
       print multipathcount
       if int(multipathcount) != 8:
          print "FATAL: Not seeing all multipath mpaths."
          exit(1)

#passive node down and mutilpaths are present, run the zpool-import-gl
obal.sh
       response = os.system("/usr/local/bin/zpool-import-global.sh");


if state == 'SLAVEACTIVE':
   print "SLAVEACTIVE";
   if hostname=='oss01':
      print "oss01"
      isonline("oss02");

   else: # I'm oss02
      print "oss02"
      isonline("oss01")

if state == 'SLAVEPASSIVE':
   print "SLAVEPASSIVE";
```

NOTE: The 8 multipath paths should be altered in this script to be the actual number of

multipaths on your system.

Note that the postfailover script calls /usr/local/bin/zpool-import-global.sh, which will import and mount the failed OSS node's zpools and OSTs, respectively. For this example, the contents of this script on oss01 should be:

```
#!/bin/bash

/sbin/zpool import -f -o cachefile=none ostpool02
/usr/bin/mount -t lustre ostpool02/ost02 /mnt/ost02
```

And the contents of this script on oss02 should be:

```
#!/bin/bash

/sbin/zpool import -f -o cachefile=none ostpool01
/usr/bin/mount -t lustre ostpool01/ost01 /mnt/ost01
```

If the check has any timeout issues, then add the following advanced config setting to cmd.conf:

AdvancedConfig = { "FailoverPowerRetries=120" }

The actual number of retries can be lowered, as desired.

Create healthcheck for "passive" OSS node

The above failovergroup defines an active-passive HA scenario for the OSS nodes. To make the scenario active-active, a healthcheck can be added to handle when the passive OSS node of that failovergroup crashes.

1. Create the script, /cm/local/apps/cmd/scripts/healthchecks/check_ha_passive.sh, with these contents:

```
#!/bin/bash

. /etc/profile.d/modules.sh
module load cmsh

# Failover has already occurred; OSTs are already imported and mounted
```

```
; Exit
FAILEDBEFORE="/var/spool/cmd/hafailed"
if [ -e "$FAILEDBEFORE" ]; then
   echo FAIL
   exit 0
fi

# Determine passive node
activeOSS=
passiveOSS=
state=
for oss in oss01 oss02; do
   state=$(ssh -o StrictHostKeyChecking=no -q $oss cat /var/spool/cmd/
state)
   if [ "$state" == "SLAVEACTIVE" ]; then
      activeOSS=$oss
   else
      passiveOSS=$oss
   fi
done

# If both nodes are down, FAIL
if [ -z "$activeOSS" ] && [ -z "$passiveOSS" ]; then
   echo FAIL
   exit 0
fi

# If active node is down, this scenario is handled by Bright failoverg
roup so we PASS here
if [ -z "$activeOSS" ]; then
   echo PASS
   exit 0
fi

# here we do the ping check to determine whether the other node is up
or down
ping -q -c1 $passiveOSS > /dev/null
if [ $? -eq 0 ]; then
   echo PASS
else
   # Ensure that passive node is powered off before migrating zpools
   cmsh -c "device use $passiveOSS; power off" 2>&1 > /dev/null
   sleep 15

   # check if the multipath is correct on active node
   if [ $(ssh -o StrictHostKeyChecking=no -q $activeOSS /usr/sbin/
multipath -ll | grep -c mpath) -ne 8 ]; then
      echo FAIL
```

```
      exit 0
   fi

   # Now import zpools
   ssh -o StrictHostKeyChecking=no -q $activeOSS /usr/local/bin/
zpool-import-global.sh 2>&1 > /dev/null
   touch $FAILEDBEFORE
   echo FAIL
# FAIL because admin show see that the passive node is down
fi
```

NOTE: The 8 multipath paths should be altered in this script to be the actual number of multipaths on your system.

2. Create the healthcheck and assign it to the headnode(s):

```
cmsh
% monitoring setup
% show checkhapassive
Parameter                      Value
-------------------------------
----------------------------------------------------------
Arguments
Automatic reinitialize         yes
Class                          Failover
Consolidator
Description
Disabled                       no
Execution multiplexer          <0 in submode>
Fuzzy offset                   0
Gap                            0
Interval                       5m
Maximal age                    0s
Maximal samples                4096
Measurables                    1 / 3194
Name                           checkhapassive
Node execution filters         <1 in submode>
Notes                          <0 bytes>
Offset                         0s
Only when idle                 no
Revision
Run in bash                    no
Script                         /cm/local/apps/cmd/scripts/
healthchecks/check_ha_passive.sh
```

```
Timeout                          60
Type                             HealthCheckScript
When                             Timed
% use checkhapassive
% nodeexecutionfilters
% show active\ head\ node
Parameter                        Value
--------------------------------
------------------------------------------------
Filter                           Active
Name                             Active head node
Resources                        Active
Revision
Type                             MonitoringResourceExecutionFilter
```

Create the Lustre client RPMs

```
cd /root/lustre-release
make clean
rm -f *.rpm
sh ./autogen.sh
./configure --disable-server --enable-client
make rpms
```

The RPMs can now be installed onto whatever image you choose for nodes that will be mounting the Lustre filesystem. For instructions on creating that image and a Lustre client category and setting up nodes to be Lustre clients, please refer to that section of the Bright installation manual. The one exception will be that the default-image, not any Lustre server image created earlier in these procedures, should be used in cloning to create the Lustre client image; otherwise, unnecessary files and packages associated with multipath, software RAID, and zfs will end up on the client nodes.

Unique solution ID: #1389
Author: Sean Eubanks
Last update: 2017-10-02 20:29