Walk-thru- Build Lustre 1.8 on CentOS 5.5 or 5.6 from Whamcloud git

Purpose

Describe the steps you need to build and test a 1.8 Lustre system (MGS, MDT, MDS, OSS, OST, client) on a CentOS 5.5 or 5.6 machine. The only difference between building on 5.5 and 5.6 is that a different kernel SRPM is required. Minor differences may appear in the output of some commands below.

This walk-thou has reportedly been used successfully on **CentOS 6.0**. One significant difference is that CentOS 6.0 replaces `/etc/modprobe.conf` with a directory: `/etc/modprobe.d/`. For CentOS 6.0 clients, the advice is to create a file `/etc/modprobe.d/lustre.conf` to contain Lustre specific module configurations.

Prerequisite

- A newly installed CentOS 5.5 x86_64 machine with the hostname: client-10.
- EPEL Repository: this is a convenient source for git.

```
# wget http://download.fedoraproject.org/pub/epel/5/i386/epel-release-5-4.noarch.rpm
# rpm -ivh ./epel-release-5-4.noarch.rpm
```

Overview

Lustre 1.8 servers require a patched and compiled kernel. Patches are readily available in the Whamcloud git source repository. A test suite is included with the Lustre 1.8 source. This document walks through the steps of patching the kernel, building Lustre and running a basic test of the complete system.

Procedure

The procedure requires that a OS is setup for development - this includes Lustre sources, kernel source and build tools. Once setup, a new kernel can be patched, compiled, run and tested. Further reading on building a CentOS RPM based kernel is available on the CentOS site.

Provision Machine

Once CentOS 5.5 is newly installed on client-10 login as root.

1. Install required kernel development tools.

```
# yum -y groupinstall "Development Tools"
# yum -y install rpm-build redhat-rpm-config unifdef gnupg quilt git
```

2. Create a user `build` with the home directory `/build`

```
# useradd -d /build build
```

3. Switch to the user `build` and change to the build `$HOME` directory.

```
# su build
# cd $HOME
```

4. Get the 1.8 branch from the Whamcloud git account.

```
# git clone git://git.whamcloud.com/fs/lustre-release.git
# cd lustre-release
# git checkout --track -b b1_8 origin/b1_8
```

5. Run `sh ./autogen.sh`

6. Resolve any outstanding dependencies until `autogen.sh` completes successfully. Success will look like:

```
# sh ./autogen.sh
Checking for a complete tree...
checking for automake-1.9 >= 1.9... found 1.9.6
...
Running automake-1.9...
configure.ac: installing `./install-sh'
configure.ac: installing `./missing'
configure.ac:9: installing `./config.guess'
configure.ac:9: installing `./config.sub'
Running autoconf...
```

Prepare the kernel source

In this walk-thru, the kernel is built using rpmbuild - a tool specific to RPM based distributions.

1. Get the kernel source. First create the directory structure, then get the source from the RPM. Create a `.rpmmacros` file to install the kernel source in our user dir.

```
# cd $HOME
# mkdir -p kernel/rpmbuild/{BUILD,RPMS,SOURCES,SPECS,SRPMS}
# cd kernel
# echo '%_topdir %(echo $HOME)/kernel/rpmbuild' > ~/.rpmmacros
```

2. Install the kernel source:

```
# rpm -i http://mirror.centos.org/centos/5.5/updates/SRPMS/kernel-2.6.18-194.32.1.el5.src.rpm 2>&1 |
grep -v mockb
```

> **Kernel versions**
>
> RHEL periodically releases updates to the kernel. The Lustre Master branch tracks the most recent kernel from Red Hat. In the event that the link above is not completely up-to-date, you should visit the Red Hat source RPM download site or CentOS source RPM download site and manually ensure you are downloading the most recent kernel.

NOTE If you are performing this walk-thru on CentOS 5.6, the kernel source can be found here:
`http://vault.centos.org/5.6/os/SRPMS/kernel-2.6.18-238.el5.src.rpm`

1. Expand the source. Using rpmbuild will also apply CentOS patches.

```
# rpmbuild -bp --target=`uname -m` ~/kernel/rpmbuild/SPECS/kernel.spec
```

This will end with:

```
...
+ patch -p1 --fuzz=2 -s
+ echo 'Patch #20240 (xen-hvm-add-hvmop_get_time-hypercall.patch):'
Patch #20240 (xen-hvm-add-hvmop_get_time-hypercall.patch):
+ patch -p1 --fuzz=2 -s
+ echo 'Patch #20241 (xen-fix-64-bit-pv-guest-user-mode-segv-crashing-host.patch):'
Patch #20241 (xen-fix-64-bit-pv-guest-user-mode-segv-crashing-host.patch):
+ patch -p1 --fuzz=2 -s
+ exit 0
```

At this point, we now have kernel souce, with all the CentOS patches applied, residing in the directory `/build/kernel/rpmbuild/BUILD/kernel-2.6.18/linux-2.6.18.x86_64`

Patch the kernel source with the Lustre code.

1. Add a unique build id so we can be certain our kernel is booted. Edit `~build/kernel/rpmbuild/BUILD/kernel-2.6.18/linux-2.6.18.x86_64/Makefile` and modify line 4, the `EXTRAVERSION` to read:

```
EXTRAVERSION = .lustre18
```

2. enter the directory `/build/kernel/rpmbuild/BUILD/kernel-2.6.18/linux-2.6.18.x86_64`

```
# cd /build/kernel/rpmbuild/BUILD/kernel-2.6.18/linux-2.6.18.x86_64
```

3. overwrite the `.config` file with `/build/lustre-release/lustre/kernel_patches/kernel_configs/kernel-2.6.18-2.6-rhel5-x86_64.config`

```
# cp /build/lustre-release/lustre/kernel_patches/kernel_configs/kernel-2.6.18-2.6-rhel5-x86_64.config ./.config
```

4. link the Lustre series and patches

```
# ln -s ~/lustre-release/lustre/kernel_patches/series/2.6-rhel5.series series
# ln -s ~/lustre-release/lustre/kernel_patches/patches patches
```

5. Apply the patches to the kernel source using quilt

```
# quilt push -av
...
...
Applying patch patches/md-avoid-bug_on-when-bmc-overflow.patch
patching file drivers/md/bitmap.c
Hunk #1 succeeded at 1161 (offset 1 line).
Hunk #3 succeeded at 1224 (offset 1 line).
patching file include/linux/raid/bitmap.h

Applying patch patches/jbd2_stats_proc_init-wrong-place.patch
patching file fs/jbd2/journal.c
Hunk #1 succeeded at 1051 (offset 152 lines).

Now at patch patches/jbd2_stats_proc_init-wrong-place.patch
```

Build the new kernel as an RPM.

1. Go into the kernel source directory and issue the following commands to build a kernel rpm.

```
# cd /build/kernel/rpmbuild/BUILD/kernel-2.6.18/linux-2.6.18.x86_64
# make oldconfig || make menuconfig
# make include/asm
# make include/linux/version.h
# make SUBDIRS=scripts
# make include/linux/utsrelease.h
# make
# make rpm
```

2. A successful build will return:

```
...
...
Requires(rpmlib): rpmlib(CompressedFileNames) <= 3.0.4-1 rpmlib(PayloadFilesHavePrefix) <= 4.0-1
Checking for unpackaged file(s): /usr/lib/rpm/check-files /var/tmp/kernel-2.6.18.lustre18-root
Wrote: /build/kernel/rpmbuild/SRPMS/kernel-2.6.18.lustre18-1.src.rpm
Wrote: /build/kernel/rpmbuild/RPMS/x86_64/kernel-2.6.18.lustre18-1.x86_64.rpm
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.35163
+ umask 022
+ cd /build/kernel/rpmbuild/BUILD
+ cd kernel-2.6.18.lustre18
+ exit 0
rm ../kernel-2.6.18.lustre18.tar.gz
```

NOTE If you receive a request to generate more entropy, you need to trigger some disk I/O or keyboard I/O. I would recommend (in another terminal):

```
# grep -Ri 'whamcloud' /usr
```

At this point, you should have a fresh kernel RPM `/build/kernel/rpmbuild/RPMS/x86_64/kernel-2.6.18.lustre18-1.x86_64.rpm`

Configure and build Lustre

1. Configure Lustre source

```
# cd ~/lustre-release/
# ./configure --with-linux=/build/kernel/rpmbuild/BUILD/kernel-2.6.18.lustre18/
...
...
LLCPPFLAGS:    -D__arch_lib__ -D_LARGEFILE64_SOURCE=1
CFLAGS:        -g -O2 -Werror
EXTRA_KCFLAGS: -include /build/lustre-release/config.h  -g -I/build/lustre-release/lnet/include -I/build
/lustre-release/lnet/include -I/build/lustre-release/lustre/include
LLCFLAGS:      -g -Wall -fPIC -D_GNU_SOURCE

Type 'make' to build Lustre.
```

2. make rpms:

```
# make rpms
...
...
Wrote: /build/kernel/rpmbuild/RPMS/x86_64/lustre-debuginfo-1.8.5.54-2.6.18_194.32.1.el5.
lustre18_201103071000.x86_64.rpm
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.15638
+ umask 022
+ cd /build/kernel/rpmbuild/BUILD
+ cd lustre-1.8.5.54
+ rm -rf /var/tmp/lustre-1.8.5.54-root
+ exit 0
make[1]: Leaving directory `/build/lustre-release'
```

3. You should now have build the following, similarly named, rpms:

```
# ls ~build/kernel/rpmbuild/RPMS/x86_64/
lustre-debuginfo-1.8.5.54-2.6.18.lustre18-1.x86_64.rpm
lustre-tests-1.8.5.54-2.6.18.lustre18-1.x86_64.rpm
lustre-source-1.8.5.54-2.6.18.lustre18-1.x86_64.rpm
lustre-modules-1.8.5.54-2.6.18.lustre18-1.x86_64.rpm
lustre-1.8.5.54-2.6.18.lustre18-1.x86_64.rpm
lustre-ldiskfs-3.1.5-2.6.18.lustre18-1.x86_64.rpm
lustre-ldiskfs-debuginfo-3.1.5-2.6.18.lustre18-1.x86_64.rpm
kernel-2.6.18.lustre18-1.x86_64.rpm
```

Installing the Lustre kernel and rebooting.

1. As root, Install the kernel

```
# rpm -ivh ~build/kernel/rpmbuild/RPMS/x86_64/kernel-2.6.18.lustre18-1.x86_64.rpm
```

2. Create a initrd

```
# mkinitrd /boot/initrd-2.6.18.lustre18.img 2.6.18.lustre18
```

3. Check that `/boot/grub/menu.lst` is configured to boot the new kernel. Add the following lines to `/boot/grub/menu.lst`

```
title CentOS Lustre18 (2.6.18.lustre18)
        root (hd0,0)
        kernel /vmlinuz-2.6.18.lustre18 ro root=/dev/VolGroup00/LogVol00
        initrd /initrd-2.6.18.lustre18.img
```

4. Ensure `/boot/grub/menu.lst` has the Lustre kernel is selected: This is 1:

```
Default=1
```

5. `reboot`
6. view the login prompt with satisfaction:

```
CentOS release 5.5 (Final)
Kernel 2.6.18.lustre18 on an x86_64

localhost login:
```

Installing Lustre.

1. Change to `root` and Change directory into `/build/kernel/rpmbuild/RPMS/x86_64/`
2. Install modules `lustre-modules` and user space tools `lustre-`

```
# rpm -ivh /build/kernel/rpmbuild/RPMS/x86_64/lustre-modules-* /build/kernel/rpmbuild/RPMS/x86_64/lustre-
1.8.* /build/kernel/rpmbuild/RPMS/x86_64/lustre-ldiskfs-.* /build/kernel/rpmbuild/RPMS/x86_64/lustre-
tests-*
```

Installing e2fsprogs

e2fsprogs is needed to run the test suite.

1. Download e2fsprogs from http://downloads.whamcloud.com/
2. Install with `rpm -ivh e2fsprogs`

A quick test

1. run `/usr/lib64/lustre/tests/llmount.sh`

```
[root@client-10 ~]# /usr/lib64/lustre/tests/llmount.sh
Stopping clients: rhel5_build /mnt/lustre (opts:)
Stopping clients: rhel5_build /mnt/lustre2 (opts:)
Loading modules from /usr/lib64/lustre/tests/..
lnet.debug=0x33f1504
lnet.subsystem_debug=0xffb7e3ff
lnet options: ' accept=all'
Formatting mgs, mds, osts
Checking servers environments
Checking clients rhel5_build environments
Setup mgs, mdt, osts
Starting mds: -o loop  /tmp/lustre-mdt /mnt/mds
lnet.debug=0x33f1504
lnet.subsystem_debug=0xffb7e3ff
lnet.debug_mb=10
Started lustre-MDT0000
Starting ost1: -o loop  /tmp/lustre-ost1 /mnt/ost1
lnet.debug=0x33f1504
lnet.subsystem_debug=0xffb7e3ff
lnet.debug_mb=10
Started lustre-OST0000
Starting ost2: -o loop  /tmp/lustre-ost2 /mnt/ost2
lnet.debug=0x33f1504
lnet.subsystem_debug=0xffb7e3ff
lnet.debug_mb=10
Started lustre-OST0001
Starting client: rhel5_build: -o user_xattr,acl,flock rhel5_build@tcp:/lustre /mnt/lustre
lnet.debug=0x33f1504
lnet.subsystem_debug=0xffb7e3ff
lnet.debug_mb=10
Using TIMEOUT=20
[root@client-10 ~]#
```

2. you will now have a Lustre filesystem available at `/mnt/lustre`
3. NOTE: if you receive an error: `mkfs.lustre: Can't parse NID 'client-10@tcp'` you'll need to associate the ip address of a non-loopback interface with name of your machine into the /etc/hosts file.

ENDS~