Walk-thru- Build Lustre MASTER on RHEL 6.4/CentOS 6.4 from Git

This walk-thru is targeting developers who want to explore the bleeding edge of Lustre. If you are evaluating Lustre for production, you should choose a Lustre Release.

Purpose

Describe the steps you need to build and test a Lustre system (MGS, MDT, MDS, OSS, OST, client) from the **`master`** branch on a x86_64, RHEL/CentOS 6.4 machine.

Prerequisite

- A newly installed RHEL/CentOS 6.4 x86_64 machine connected to the internet.
- EPEL Repository: this is a convenient source for git.
  NOTE the EPEL 5 repository is used because it includes `quilt`.
- **NOTE** It is suggested that you have at least 1GB of memory on the machine you are using for the build.
- **NOTE** Verify that SElinux is disabled.

Overview

pre-built RPMs are available

Lustre servers currently require a patched and compiled kernel. A patched and compiled Lustre server kernel is available from Whamcloud. A separate page is available to walk thru setting up Lustre with these pre-built RPMs. This document is for those who wish to build their Lustre system from source. Note that if you are not modifying the kernel patches on the server, it is possible to use the pre-built Lustre server kernel RPMs, and only build the Lustre code. Note that a patched kernel is **NOT** needed for the Lustre client.

Patches are available in the Git source repository. A test suite is included with the Lustre 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 RHEL RPM based kernel is available from, among other sources, the CentOS site.

Provision machine and installing dependencies.

Once RHEL 6.3 is newly installed on `rhel6-master` login as root.

1. Install the kernel development tools.

```
# yum -y groupinstall "Development Tools"
```

Problem with installing 'Development Tools'

If the Development Tools group is not be available for some reason, you may find the following list if individual packages necessary to install.

```
# yum -y install automake xmlto asciidoc elfutils-libelf-devel zlib-devel binutils-devel newt-
devel python-devel hmaccalc perl-ExtUtils-Embed rpm-build make gcc redhat-rpm-config patchutils
git
```

2. Install additional dependencies

```
# yum -y install xmlto asciidoc elfutils-libelf-devel zlib-devel binutils-devel newt-devel python-devel
hmaccalc perl-ExtUtils-Embed bison elfutils-devel audit-libs-devel
```

3. Install EPEL 5. **NOTE** EPEL5 is used because it contains `quilt` and `libselinux-devel`

```
# rpm -ivh http://download.fedoraproject.org/pub/epel/5/x86_64/epel-release-5-4.noarch.rpm
```

4. Install `quilt` and `libselinux-devel`

```
# yum -y install quilt libselinux-devel
```

newt-devel

`newt-devel` may not be available if you are using RHEL6. One option is to download newt-devel, slang-devel, and asciidoc RPMs from CentOS and install with:

```
yum --nogpgcheck localinstall ./newt-devel-0.52.11-3.el6.x86_64.rpm ./slang-devel-2.2.1-1.el6.
x86_64.rpm ./asciidoc-8.4.5-4.1.el6.noarch.rpm
```

Preparing the Lustre source.

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

```
# useradd -m build
```

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

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

3. Get the MASTER branch from HPDD git.

```
# git clone git://git.whamcloud.com/fs/lustre-release.git
# cd lustre-release
```

4. Run `sh ./autogen.sh`
5. 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
...
...
configure.ac:10: installing `./config.sub'
configure.ac:12: installing `./install-sh'
configure.ac:12: installing `./missing'

Running autoconf
```

Prepare a patched kernel for Lustre

You can have different ways to prepare a patched kernel for Lustre. The easier method is to download built RPM packages from the Releases page. You're going to need the packages starting with '`kernel-`'. After new kernel packages are downloaded, you can skip the following few steps and go to the section 'Installing the Lustre kernel and rebooting'.

If you want a more challenge life, you can patch the kernel by yourself, in that case, please follow the steps below.

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 -ivh http://ftp.redhat.com/pub/redhat/linux/enterprise/6Server/en/os/SRPMS/kernel-2.6.32-431.5.1.
el6.src.rpm 2>&1 | grep -v mockb

NOTE: For RHEL 7, you can use
# rpm -ivh http://vault.centos.org/7.2.1511/updates/Source/SPackages/kernel-3.10.0-327.22.2.el7.src.
rpm 2>&1 | grep -v exist
```

RHEL kernel versions

Red Hat periodically release updates to their distributed kernel. The Lustre Master attempts to stay up-to-date with 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 and ensure you are downloading the most recent kernel. The most recent supported kernel is recorded in `lustre/kernel_patches/which_patch`.

3. Prepare the source using rpmbuild.

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

This will end with:

```
...
gpg: Total number processed: 1
gpg:               imported: 1
+ gpg --homedir . --export --keyring ./kernel.pub Red
gpg: WARNING: unsafe permissions on homedir `.'
+ gcc -o scripts/bin2c scripts/bin2c.c
+ scripts/bin2c ksign_def_public_key __initdata
+ cd ..
+ exit 0
```

At this point, we now have kernel source, with all the RHEL/CentOS patches applied, residing in the directory `/home/build/kernel/rpmbuild/BUILD/kernel-2.6.431.5.1.el6/linux-2.6.32-431.5.1.el6.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 `~/kernel/rpmbuild/BUILD/kernel-2.6.32-431.5.1.el6/linux-2.6.32-431.5.1.el6.x86_64/Makefile` and modify line 4, the `EXTRAVERSION` to read:

```
EXTRAVERSION = .431.5.1.el6_lustre
```

2. Enter the directory `~/kernel/rpmbuild/BUILD/kernel-2.6.32-431.5.1.el6/linux-2.6.32-431.5.1.el6.x86_64/`

```
# cd ~/kernel/rpmbuild/BUILD/kernel-2.6.32-431.5.1.el6/linux-2.6.32-431.5.1.el6.x86_64/
```

3. Overwrite the `.config` file with `~/lustre-release/lustre/kernel_patches/kernel_configs/kernel-2.6.32-2.6-rhel6-x86_64.config`

```
# cp ~/lustre-release/lustre/kernel_patches/kernel_configs/kernel-2.6.32-2.6-rhel6-x86_64.config ./.
config
```

4. Link the Lustre series and patches

```
# ln -s ~/lustre-release/lustre/kernel_patches/series/2.6-rhel6.series series
# ln -s ~/lustre-release/lustre/kernel_patches/patches patches
```

5. Apply the patches to the kernel source using quilt

```
# quilt push -av
...
...
patching file fs/jbd2/transaction.c
Hunk #3 succeeded at 1222 (offset 3 lines).
Hunk #4 succeeded at 1357 (offset 3 lines).

Now at patch patches/jbd2-jcberr-2.6-rhel6.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 ~/kernel/rpmbuild/BUILD/kernel-2.6.32-431.5.1.el6/linux-2.6.32-431.5.1.el6.x86_64/
# make oldconfig || make menuconfig
# make include/asm
# make include/linux/version.h
# make SUBDIRS=scripts
# make include/linux/utsrelease.h
# make rpm

NOTE: with RHEL 7,
# make oldconfig
# make -j4 rpm
```

2. A successful build will return:

```
...
...
Wrote: /home/build/kernel/rpmbuild/SRPMS/kernel-2.6.32lustremaster-1.src.rpm
Wrote: /home/build/kernel/rpmbuild/RPMS/x86_64/kernel-2.6.32.lustremaster-1.x86_64.rpm
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.f73m1V
+ umask 022
+ cd /home/build/kernel/rpmbuild/BUILD
+ cd kernel-2.6.32lustremaster
+ rm -rf /home/build/kernel/rpmbuild/BUILDROOT/kernel-2.6.32.lustremaster-1.x86_64
+ exit 0
rm ../kernel-2.6.32lustremaster.tar.gz
```

If you receive a request to generate more entropy, you need to trigger some disk I/O or keyboard I/O. In another terminal, you can either type randomly or execute the following command to generate entropy:

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

At this point, you should have a fresh kernel RPM `~/kernel/rpmbuild/RPMS/x86_64/kernel-2.6.32.lustremaster-1.x86_64.rpm`

Installing the Lustre kernel and rebooting.

1. As **root**, Install the kernel

```
# rpm -ivh $PKG_PATH/kernel-*.rpm
```

Depending on how you got your kernel packages, the **PKG_PATH** should be `~build/kernel/rpmbuild/RPMS/x86_64` if you built the packages by yourself, or any other directory where you downloaded the packages from https://build.whamcloud.com/.

2. Create initrd using `dracut` (*This may not be required because `initrd` should have been created by installing new kernel*)

```
# /sbin/new-kernel-pkg --package kernel --mkinitrd --dracut --depmod --install 2.6.32.431.5.1.el6_lustre
```

3. optional turn on `lustre` services, and specify network type for `lnet`

```
* chkconfig lustre on
* vi /etc/modprobe.d/lustre.conf
```

   If you don't know what should be written to this file, just leave it empty for now.
4. Reboot the `system with the reboot` command.
5. view the login prompt with satisfaction, and make sure that new kernel is running:

```
Red Hat Enterprise Linux Server release 6.0 (Santiago)
Kernel 2.6.32l-'I'm new kernel' on an x86_64

client-10 login:
```

Configure and build Lustre

1. Configure Lustre source

```
# cd ~/lustre-release/
# ./configure --with-linux=/lib/modules/kernel-2.6.32_lustremaster/build
...
...
LLCPPFLAGS:    -D__arch_lib__ -D_LARGEFILE64_SOURCE=1
CFLAGS:        -g -O2 -Werror
EXTRA_KCFLAGS: -include /home/build/lustre-release/config.h  -g -I/home/build/lustre-release/libcfs
/include -I/home/build/lustre-release/lnet/include -I/home/build/lustre-release/lustre/include
LLCFLAGS:      -g -Wall -fPIC -D_GNU_SOURCE

Type 'make' to build Lustre.
```

2. Make rpms:

```
# make rpms
...
...
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.TsLWpD
+ umask 022
+ cd /home/build/kernel/rpmbuild/BUILD
+ cd lustre-2.0.61
+ rm -rf /home/build/kernel/rpmbuild/BUILDROOT/lustre-2.0.61-2.6.32_lustremaster_g0533e7b.x86_64
+ exit 0
make[1]: Leaving directory `/home/build/lustre-release'
```

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

```
# ls *.rpm
kernel-2.6.32lustremaster-1.x86_64.rpm
lustre-2.0.61-2.6.32.lustremaster_g0533e7b.x86_64.rpm
lustre-debuginfo-2.0.61-2.6.32.lustremaster_g0533e7b.x86_64.rpm
lustre-ldiskfs-3.3.0-2.6.32.lustremaster_g0533e7b.x86_64.rpm
lustre-ldiskfs-debuginfo-3.3.0-2.6.32.lustremaster_g0533e7b.x86_64.rpm
lustre-modules-2.0.61-2.6.32.lustremaster_g0533e7b.x86_64.rpm
lustre-source-2.0.61-2.6.32.lustremaster_g0533e7b.x86_64.rpm
lustre-tests-2.0.61-2.6.32.lustremaster_g0533e7b.x86_64.rpm
```

Installing e2fsprogs

`e2fsprogs` is needed to run the test suite.

1. Download `e2fsprogs` from http://downloads.whamcloud.com/public/e2fsprogs/latest/
2. Install with

```
# rpm -Uvh ./e2fsprogs-1.42.6.wc2-7.el6.x86_64.rpm  ./e2fsprogs-libs-1.42.6.wc2-7.el6.x86_64.rpm
```

Installing Lustre.

1. Change to `root` and Change directory into `~build/lustre-release/`
2. Install modules `lustre-modules` and user space tools `lustre-`

```
# rpm -ivh lustre-ldiskfs-3.3.0-2.6.32.lustremaster*
# rpm -ivh lustre-modules-2.0.61-2.6.32.lustremaster*
# rpm -ivh lustre-2.0.61-2.6.32.lustremaster_*
# rpm -ivh lustre-tests-*
```

Disable SELinux (Lustre Servers)

SELinux, which is on by default in RHEL/CentOS, will prevent the format commands for the various Lustre targets from completing. Therefore you must either disable it or adjust the settings. These instructions explain how to disable it.

1. Run `getenforce` to see if SELinux is enabled. It should return 'Enforcing' or 'Disabled'.
2. To disable it, edit `/etc/selinux/config` and change the line '`selinux=enforcing`' to '`selinux=disabled`'.
3. Finally, reboot your system.

```
# vi /etc/selinux/config

----
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
# targeted - Only targeted network daemons are protected.
# strict - Full SELinux protection.
SELINUXTYPE=targeted
---
# shutdown -r now
```

Testing

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

```
# /usr/lib64/lustre/tests/llmount.sh
Loading modules from /usr/lib64/lustre/tests/..
debug=0x33f0404
subsystem_debug=0xffb7e3ff
gss/krb5 is not supported
Formatting mgs, mds, osts
Format mds1: /tmp/lustre-mdt1
Format ost1: /tmp/lustre-ost1
Format ost2: /tmp/lustre-ost2
Checking servers environments
Checking clients rhel6-master environments
Loading modules from /usr/lib64/lustre/tests/..
debug=0x33f0404
subsystem_debug=0xffb7e3ff
gss/krb5 is not supported
Setup mgs, mdt, osts
Starting mds1: -o loop,user_xattr,acl  /tmp/lustre-mdt1 /mnt/mds1
debug=0x33f0404
subsystem_debug=0xffb7e3ff
debug_mb=10
Started lustre-MDT0000
Starting ost1: -o loop  /tmp/lustre-ost1 /mnt/ost1
debug=0x33f0404
subsystem_debug=0xffb7e3ff
debug_mb=10
Started lustre-OST0000
Starting ost2: -o loop  /tmp/lustre-ost2 /mnt/ost2
debug=0x33f0404
subsystem_debug=0xffb7e3ff
debug_mb=10
Started lustre-OST0001
Starting client: rhel5-build: -o user_xattr,acl,flock rhel6-master@tcp:/lustre /mnt/lustre
debug=0x33f0404
subsystem_debug=0xffb7e3ff
debug_mb=10
Using TIMEOUT=20
disable quota as required
```

2. You will now have a Lustre filesystem available at `/mnt/lustre`
3. NOTE: if you receive an error similar to: `mkfs.lustre: Can't parse NID 'rhel6-master@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~