Walk-thru- Build Lustre MASTER on RHEL 8.0/CentOS 8.0 from Git

Purpose

Describe the steps you need to build and test a Lustre system (MGS, MDT, MDS, OSS, OST, client) from the **`master`** branch on an **arm64**, RHEL/CentOS 8.0 machine.

Prerequisite

- A newly installed RHEL/CentOS 8.0 arm64 machine connected to the internet.
- **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

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/CentOS 8.0 is newly installed on an `arm64 machine` login as user `root`.

1. Install the kernel development tools:

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

2. Install additional dependencies:

```
# yum -y install audit-libs-devel binutils-devel elfutils-devel kabi-dw ncurses-devel newt-devel numactl-devel openssl-devel pciutils-devel perl perl-devel python2 python3-docutils xmlto xz-devel elfutils-libelf-devel libcap-devel libcap-ng-devel llvm-toolset libyaml libyaml-devel kernel-rpm-macros kernel-abi-whitelists
```

3. Install e2fsprogs packages:

   Download the following packages from https://build.whamcloud.com/job/e2fsprogs-master/arch=aarch64,distro=el8/:

```
e2fsprogs e2fsprogs-devel e2fsprogs-libs libcom_err libcom_err-devel libss libss-devel
```

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 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
configure.ac:15: installing 'config/compile'
configure.ac:10: installing 'config/config.guess'
configure.ac:10: installing 'config/config.sub'
configure.ac:12: installing 'config/install-sh'
configure.ac:12: installing 'config/missing'
libcfs/libcfs/autoMakefile.am: installing 'config/depcomp'
$
```

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 directory:

```
$ 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 https://vault.centos.org/8.0.1905/BaseOS/Source/SPackages/kernel-4.18.0-80.11.2.el8_0.src.rpm
```

3. Prepare the source using `rpmbuild`:

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

This will end with:

```
...
Processed config files are in /home/build/kernel/rpmbuild/BUILD/kernel-4.18.0-80.11.2.el8_0/linux-4.18.0-
80.11.2.el8.aarch64/configs
+ cd ..
+ find . '(' -name '*.orig' -o -name '*~' ')' -exec rm -f '{}' ';'
+ find . -name .gitignore -exec rm -f '{}' ';'
+ cd ..
+ exit 0
```

4. Copy the kernel config file into lustre tree:

```
cp ~/kernel/rpmbuild/BUILD/kernel-4.18.0-80.11.2.el8_0/linux-4.18.0-80.11.2.el8.aarch64/configs/kernel-4.18.0-4.18.0-aarch64.config ~/lustre-release/lustre/kernel_patches/kernel_configs/kernel-4.18.0-4.18-rhel8-aarch64.config
```

5. Edit the kernel config file ~/lustre-release/lustre/kernel_patches/kernel_configs/kernel-4.18.0-4.18-rhel8-aarch64.config:

   Find the line with '# IO Schedulers' and insert following two lines below it:

   CONFIG_IOSCHED_DEADLINE=y
   CONFIG_DEFAULT_IOSCHED="deadline"

At this point, we now have kernel source, with all the RHEL/CentOS patches applied, residing in the directory `~/kernel/rpmbuild/BUILD/kernel-4.18.0-80.11.2.el8_0/linux-4.18.0-80.11.2.el8.aarch64/.`

Patch the kernel source with the Lustre code.

1. Gather all the patches from `lustre` tree into a single file:

```
$ cd ~
$ rm -f ~/lustre-kernel-aarch64-lustre.patch
$ cd ~/lustre-release/lustre/kernel_patches/series
$ for patch in $(<"4.18-rhel8.series"); do \
      patch_file="$HOME/lustre-release/lustre/kernel_patches/patches/${patch}"; \
      cat "${patch_file}" >> "$HOME/lustre-kernel-aarch64-lustre.patch"; \
  done
$
```

2. Copy the kernel patch into RPM build tree:

```
cp ~/lustre-kernel-aarch64-lustre.patch ~/kernel/rpmbuild/SOURCES/patch-4.18.0-lustre.patch
```

3. Edit the kernel spec file `~/kernel/rpmbuild/SPECS/kernel.spec`:

   Find the line with '`find $RPM_BUILD_ROOT/lib/modules/$KernelVer`' and insert following line below it:

   ```
   cp -a fs/ext4/* $RPM_BUILD_ROOT/lib/modules/$KernelVer/build/fs/ext4
   ```

   ```
   Note: please add the following extra line for RHEL 8.4:
   ```

   ```
   rm -f $RPM_BUILD_ROOT/lib/modules/$KernelVer/build/fs/ext4/ext4-inode-test*
   ```

   Find the line with '`# empty final patch to facilitate testing of kernel patches`' and insert following two lines below it:

   ```
   # adds Lustre patches
   ```

   ```
   Patch99995: patch-%{version}-lustre.patch
   ```

   Find the line with '`ApplyOptionalPatch linux-kernel-test.patch`' and insert following two lines below it:

   ```
   # lustre patch
   ```

   ```
   ApplyOptionalPatch patch-%{version}-lustre.patch
   ```

   Save and close the spec file.

   or

   Run the following command to edit the kernel spec file:

```
sed -i.inst -e '/^    find $RPM_BUILD_ROOT\/lib\/modules\/$KernelVer/a\
    cp -a fs/ext4/* $RPM_BUILD_ROOT/lib/modules/$KernelVer/build/fs/ext4\
    rm -f $RPM_BUILD_ROOT/lib/modules/$KernelVer/build/fs/ext4/ext4-inode-test*' \
-e '/^# empty final patch to facilitate testing of kernel patches/i\
Patch99995: patch-%{version}-lustre.patch' \
-e '/^ApplyOptionalPatch linux-kernel-test.patch/i\
ApplyOptionalPatch patch-%{version}-lustre.patch' \
~/kernel/rpmbuild/SPECS/kernel.spec
```

4. Overwrite the kernel `config` file with `~/lustre-release/lustre/kernel_patches/kernel_configs/kernel-4.18.0-4.18-rhel8-aarch64.config`:

```
echo '# arm64' > ~/kernel/rpmbuild/SOURCES/kernel-4.18.0-aarch64.config
cat ~/lustre-release/lustre/kernel_patches/kernel_configs/kernel-4.18.0-4.18-rhel8-aarch64.config >> ~
/kernel/rpmbuild/SOURCES/kernel-4.18.0-aarch64.config
```

Build the new kernel as an RPM.

1. Start building the kernel with `rpmbuild`:

```
$ cd ~/kernel/rpmbuild
$ buildid="_lustre" # Note: change to any string that identify your work
$ rpmbuild -ba --with firmware --target aarch64 --with baseonly \
           --without kabichk --define "buildid ${buildid}" \
           ~/kernel/rpmbuild/SPECS/kernel.spec
```

2. A successful build will return:

```
...
Checking for unpackaged file(s): /usr/lib/rpm/check-files /home/build/kernel/rpmbuild/BUILDROOT/kernel-
4.18.0-80.11.2.el8_lustre.aarch64
Wrote: /home/build/kernel/rpmbuild/SRPMS/kernel-4.18.0-80.11.2.el8_lustre.src.rpm
Wrote: /home/build/kernel/rpmbuild/RPMS/aarch64/kernel-4.18.0-80.11.2.el8_lustre.aarch64.rpm
Wrote: /home/build/kernel/rpmbuild/RPMS/aarch64/kernel-headers-4.18.0-80.11.2.el8_lustre.aarch64.rpm
Wrote: /home/build/kernel/rpmbuild/RPMS/aarch64/kernel-cross-headers-4.18.0-80.11.2.el8_lustre.aarch64.
rpm
Wrote: /home/build/kernel/rpmbuild/RPMS/aarch64/kernel-debuginfo-common-aarch64-4.18.0-80.11.2.
el8_lustre.aarch64.rpm
Wrote: /home/build/kernel/rpmbuild/RPMS/aarch64/perf-4.18.0-80.11.2.el8_lustre.aarch64.rpm
Wrote: /home/build/kernel/rpmbuild/RPMS/aarch64/perf-debuginfo-4.18.0-80.11.2.el8_lustre.aarch64.rpm
Wrote: /home/build/kernel/rpmbuild/RPMS/aarch64/python3-perf-4.18.0-80.11.2.el8_lustre.aarch64.rpm
Wrote: /home/build/kernel/rpmbuild/RPMS/aarch64/python3-perf-debuginfo-4.18.0-80.11.2.el8_lustre.aarch64.
rpm
Wrote: /home/build/kernel/rpmbuild/RPMS/aarch64/kernel-tools-4.18.0-80.11.2.el8_lustre.aarch64.rpm
Wrote: /home/build/kernel/rpmbuild/RPMS/aarch64/kernel-tools-libs-4.18.0-80.11.2.el8_lustre.aarch64.rpm
Wrote: /home/build/kernel/rpmbuild/RPMS/aarch64/kernel-tools-libs-devel-4.18.0-80.11.2.el8_lustre.
aarch64.rpm
Wrote: /home/build/kernel/rpmbuild/RPMS/aarch64/kernel-tools-debuginfo-4.18.0-80.11.2.el8_lustre.aarch64.
rpm
Wrote: /home/build/kernel/rpmbuild/RPMS/aarch64/bpftool-4.18.0-80.11.2.el8_lustre.aarch64.rpm
Wrote: /home/build/kernel/rpmbuild/RPMS/aarch64/bpftool-debuginfo-4.18.0-80.11.2.el8_lustre.aarch64.rpm
Wrote: /home/build/kernel/rpmbuild/RPMS/aarch64/kernel-core-4.18.0-80.11.2.el8_lustre.aarch64.rpm
Wrote: /home/build/kernel/rpmbuild/RPMS/aarch64/kernel-devel-4.18.0-80.11.2.el8_lustre.aarch64.rpm
Wrote: /home/build/kernel/rpmbuild/RPMS/aarch64/kernel-modules-4.18.0-80.11.2.el8_lustre.aarch64.rpm
Wrote: /home/build/kernel/rpmbuild/RPMS/aarch64/kernel-modules-extra-4.18.0-80.11.2.el8_lustre.aarch64.
rpm
Wrote: /home/build/kernel/rpmbuild/RPMS/aarch64/kernel-debuginfo-4.18.0-80.11.2.el8_lustre.aarch64.rpm
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.SqV3vr
+ umask 022
+ cd /home/build/kernel/rpmbuild/BUILD
+ cd kernel-4.18.0-80.11.2.el8_0
+ rm -rf /home/build/kernel/rpmbuild/BUILDROOT/kernel-4.18.0-80.11.2.el8_lustre.aarch64
+ exit 0
```

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
```

Installing the Lustre kernel and rebooting.

1. As **root**, Install the `kernel` packages:

```
# cd ~/kernel/rpmbuild/RPMS/aarch64/
# rpm -Uvh kernel-core-4.18.0-80.11.2.el8_lustre.aarch64.rpm kernel-headers-4.18.0-80.11.2.el8_lustre.
aarch64.rpm kernel-modules-4.18.0-80.11.2.el8_lustre.aarch64.rpm kernel-modules-extra-4.18.0-80.11.2.
el8_lustre.aarch64.rpm kernel-4.18.0-80.11.2.el8_lustre.aarch64.rpm kernel-devel-4.18.0-80.11.2.
el8_lustre.aarch64.rpm kernel-tools-4.18.0-80.11.2.el8_lustre.aarch64.rpm kernel-tools-libs-4.18.0-
80.11.2.el8_lustre.aarch64.rpm
```

2. Reboot the system.
3. Login system after reboot:

```
# uname -r
4.18.0-80.11.2.el8_lustre.aarch64
```

Now you are running a Lustre patched kernel!

Configure and build Lustre

1. Configure Lustre source:

```
$ cd ~/lustre-release/
$ ./configure --with-linux=/home/build/kernel/rpmbuild/BUILD/kernel-4.18.0-80.11.2.el8_0/linux-4.18.0-
80.11.2.el8_lustre.aarch64/
...
...
config.status: executing libtool commands

CC:            gcc
LD:            /usr/bin/ld
CPPFLAGS:      -include /home/build/lustre-release/undef.h -include /home/build/lustre-release/config.h -
I/home/build/lustre-release/lnet/include/uapi -I/home/build/lustre-release/lustre/include/uapi -I/home
/build/lustre-release/libcfs/include -I/home/build/lustre-release/lnet/utils -I/home/build/lustre-release
/lustre/include
CFLAGS:        -g -O2
EXTRA_KCFLAGS: -include /home/build/lustre-release/undef.h -include /home/build/lustre-release/config.h
-g -I/home/build/lustre-release/libcfs/include -I/home/build/lustre-release/libcfs/include/libcfs -I/home
/build/lustre-release/lnet/include/uapi -I/home/build/lustre-release/lnet/include -I/home/build/lustre-
release/lustre/include/uapi -I/home/build/lustre-release/lustre/include -Wno-format-truncation -Wno-
stringop-truncation -Wno-stringop-overflow

Type 'make' to build Lustre.
```

2. Make RPMs:

```
$ make rpms
...
...
Wrote: /tmp/rpmbuild-lustre-build-UPMee2HJ/RPMS/aarch64/lustre-2.13.57_47_g475173d-1.el8.aarch64.rpm
Wrote: /tmp/rpmbuild-lustre-build-UPMee2HJ/RPMS/aarch64/kmod-lustre-2.13.57_47_g475173d-1.el8.aarch64.rpm
Wrote: /tmp/rpmbuild-lustre-build-UPMee2HJ/RPMS/aarch64/kmod-lustre-osd-ldiskfs-2.13.57_47_g475173d-1.
el8.aarch64.rpm
Wrote: /tmp/rpmbuild-lustre-build-UPMee2HJ/RPMS/aarch64/lustre-osd-ldiskfs-mount-2.13.57_47_g475173d-1.
el8.aarch64.rpm
Wrote: /tmp/rpmbuild-lustre-build-UPMee2HJ/RPMS/aarch64/lustre-resource-agents-2.13.57_47_g475173d-1.el8.
aarch64.rpm
Wrote: /tmp/rpmbuild-lustre-build-UPMee2HJ/RPMS/aarch64/lustre-devel-2.13.57_47_g475173d-1.el8.aarch64.
rpm
Wrote: /tmp/rpmbuild-lustre-build-UPMee2HJ/RPMS/aarch64/lustre-tests-2.13.57_47_g475173d-1.el8.aarch64.
rpm
Wrote: /tmp/rpmbuild-lustre-build-UPMee2HJ/RPMS/aarch64/kmod-lustre-tests-2.13.57_47_g475173d-1.el8.
aarch64.rpm
Wrote: /tmp/rpmbuild-lustre-build-UPMee2HJ/RPMS/aarch64/lustre-iokit-2.13.57_47_g475173d-1.el8.aarch64.
rpm
Wrote: /tmp/rpmbuild-lustre-build-UPMee2HJ/RPMS/aarch64/lustre-debugsource-2.13.57_47_g475173d-1.el8.
aarch64.rpm
Wrote: /tmp/rpmbuild-lustre-build-UPMee2HJ/RPMS/aarch64/lustre-debuginfo-2.13.57_47_g475173d-1.el8.
aarch64.rpm
Wrote: /tmp/rpmbuild-lustre-build-UPMee2HJ/RPMS/aarch64/kmod-lustre-debuginfo-2.13.57_47_g475173d-1.el8.
aarch64.rpm
Wrote: /tmp/rpmbuild-lustre-build-UPMee2HJ/RPMS/aarch64/kmod-lustre-osd-ldiskfs-debuginfo-2.13.57
_47_g475173d-1.el8.aarch64.rpm
Wrote: /tmp/rpmbuild-lustre-build-UPMee2HJ/RPMS/aarch64/lustre-osd-ldiskfs-mount-debuginfo-2.13.57
_47_g475173d-1.el8.aarch64.rpm
Wrote: /tmp/rpmbuild-lustre-build-UPMee2HJ/RPMS/aarch64/lustre-tests-debuginfo-2.13.57_47_g475173d-1.el8.
aarch64.rpm
Wrote: /tmp/rpmbuild-lustre-build-UPMee2HJ/RPMS/aarch64/kmod-lustre-tests-debuginfo-2.13.57_47_g475173d-
1.el8.aarch64.rpm
Executing(%clean): /bin/sh -e /tmp/rpmbuild-lustre-build-UPMee2HJ/TMP/rpm-tmp.CovhjA
+ umask 022
+ cd /tmp/rpmbuild-lustre-build-UPMee2HJ/BUILD
+ cd lustre-2.13.57_47_g475173d
+ rm -rf /tmp/rpmbuild-lustre-build-UPMee2HJ/BUILDROOT/lustre-2.13.57_47_g475173d-1.el8.aarch64
+ rm -rf /tmp/rpmbuild-lustre-build-UPMee2HJ/TMP/kmp
+ exit 0
Executing(--clean): /bin/sh -e /tmp/rpmbuild-lustre-build-UPMee2HJ/TMP/rpm-tmp.P4Pr7b
+ umask 022
+ cd /tmp/rpmbuild-lustre-build-UPMee2HJ/BUILD
+ rm -rf lustre-2.13.57_47_g475173d
+ exit 0
```

3. You should now have the Lustre RPMs.

Installing Lustre.

Change to `root` and change directory into /home/`build/lustre-release/`:

```
# yum localinstall *.aarch64.rpm
```

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
Stopping clients: sr050 /mnt/lustre (opts:-f)
Stopping clients: sr050 /mnt/lustre2 (opts:-f)
sr050: executing set_hostid
Loading modules from /usr/lib64/lustre/tests/..
detected 160 online CPUs by sysfs
libcfs will create CPU partition based on online CPUs
Formatting mgs, mds, osts
Format mds1: /tmp/lustre-mdt1
Format ost1: /tmp/lustre-ost1
Format ost2: /tmp/lustre-ost2
Checking servers environments
Checking clients sr050 environments
Loading modules from /usr/lib64/lustre/tests/..
detected 160 online CPUs by sysfs
libcfs will create CPU partition based on online CPUs
Setup mgs, mdt, osts
Starting mds1: -o localrecov  /dev/mapper/mds1_flakey /mnt/lustre-mds1
Commit the device label on /tmp/lustre-mdt1
Started lustre-MDT0000
Starting ost1: -o localrecov  /dev/mapper/ost1_flakey /mnt/lustre-ost1
Commit the device label on /tmp/lustre-ost1
Started lustre-OST0000
Starting ost2: -o localrecov  /dev/mapper/ost2_flakey /mnt/lustre-ost2
Commit the device label on /tmp/lustre-ost2
Started lustre-OST0001
Starting client: sr050:  -o user_xattr,flock sr050@tcp:/lustre /mnt/lustre
UUID                   1K-blocks        Used   Available Use% Mounted on
lustre-MDT0000_UUID       125056        1644      112176   2% /mnt/lustre[MDT:0]
lustre-OST0000_UUID       313104        1244      284700   1% /mnt/lustre[OST:0]
lustre-OST0001_UUID       313104        1244      284700   1% /mnt/lustre[OST:1]

filesystem_summary:       626208        2488      569400   1% /mnt/lustre

Using TIMEOUT=20
osc.lustre-OST0000-osc-ffffce9052651800.idle_timeout=debug
osc.lustre-OST0001-osc-ffffce9052651800.idle_timeout=debug
setting jobstats to procname_uid
Setting lustre.sys.jobid_var from disable to procname_uid
Waiting 90s for 'procname_uid'
Updated after 4s: want 'procname_uid' got 'procname_uid'
disable quota as required
lod.lustre-MDT0000-mdtlov.mdt_hash=crush
```

2. You will now have a Lustre filesystem available at `/mnt/lustre`

ENDS~