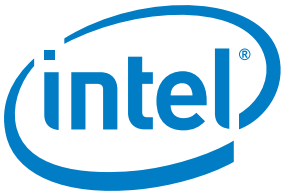


Building Lustre* Servers with Intel® Omni-Path Architecture

Application Note

October 2015

Document Number: J10040-1.0

Legal Disclaimer

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting: http://www.intel.com/design/literature.htm

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at http://www.intel.com/ or from the OEM or retailer.

No computer system can be absolutely secure.

Intel Omni-Path and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Contents

Revision History

Date	Revision	Description
October 2015	1.0	Initial release.

§

1 Introduction

This guide is intended for developers who want to explore the "bleeding edge" of Lustre Intel® Omni-Path Architecture. If you are evaluating Lustre for production, contact Intel HPDD (http://www.intel.com/content/www/us/en/software/intel-enterprise-edition-for-lustre-software.html) or your preferred vendor.

The guide describes the steps you need to build and test a Lustre system (MGS, MDT, MDS, OSS, OST, client) from the HPDD master branch on a x86_64, RHEL/CentOS 7.1 machine. For these instructions, we assume CentOS version 7.1.1503 is used.

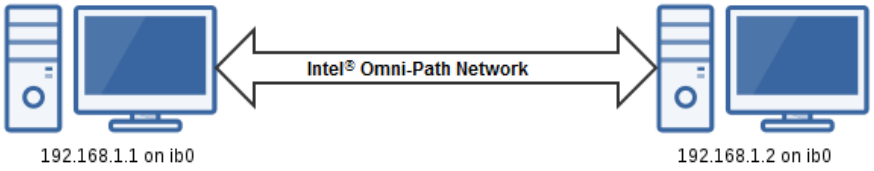
Note: This document describes using CentOS but instructions for RHEL are identical.

2 Prerequisites

Lustre servers require a patched and compiled kernel. A patched and compiled Lustre server kernel is available from Intel that works with in-kernel network drivers. A separate page is available for setting up Lustre with these pre-built RPMs. This guide is for those who want to build their Lustre system from source and are using a distribution that has not included the relevant headers to support Intel® Omni-Path Architecture. Note that a patched kernel is NOT needed for the Lustre client.

Intel® Omni-Path Architecture is a high-performance network.

- Two newly installed CentOS 7.1 x86_64 machines connected to the Internet to install packages. Each machine must have an Intel® Omni-Path Architecture card inserted.



- EPEL Repository: This is a convenient source for quilt.
- IFS tool suite is installed on the nodes and the nodes are operational.
- Familiarity with the contents of the Lustre Operations Manual.

Note: Intel suggests that you have more than 1GB of memory and more than 20GB hard disk space available on the machine you are using for the build.

Note: Verify that SELinux is disabled.

3 Overview

At the time of writing, the availability of compat-rdma shim for Intel® Omni-Path Architecure has been limited. Currently, compat-rdma requires a rebuild when used with a Lustre patched kernel. Completing this task is described in detail in the steps below.

Lustre is an open source file system with an active community rapidly fixing bugs and providing enhancements. It is expected that with the formal release of Intel® Omni-Path Architecure, the latest Lustre source code will receive the enhancements it needs to considerably simplify Lustre servers on Intel® Omni-Path Architecure. If you prefer not to compile your own Lustre file system, contact Intel directly.

To get Intel® Omni-Path Architecture working with Lustre servers, the high-level steps, which are detailed in sections later in the document, are the following:

1. Build the distro kernel with Lustre patches to give k'. See Building the Distro Kernel on page 8.
2. Build compat-rdma from Intel against k' to give I'. See Configuring, Building, and Installing compat-rdma for I' on page 12.
3. Build Lustre against k' and I'. See Configuring and Building Lustre on page 13.
4. Install Lustre packages in order. See Install Lustre on page 14.
5. Set up and test Lustre. See Testing Lustre on page 15.

Lustre source code is available in the HPDD Git source repository. A test suite is included with the Lustre source. This guide provides the steps for patching the kernel, building Lustre, and running a basic test of the complete system. Detailed steps follow.

4 Building the Distro Kernel

The following procedure requires that an OS is set up for development, including Lustre sources, kernel source, and build tools. After being set up, a new kernel can be patched, compiled, run, and tested. Further reading on building a CentOS RPM-based kernel is available from the CentOS site, among other sources.

4.1 Provisioning the Machine and Installing Dependencies

1. After CentOS 7.1 is newly installed, log in as root.
2. Install the kernel development tools:
```
yum -y groupinstall "Development Tools"
```
3. Install additional dependencies:
```
yum -y install net-tools bc xmlto asciidoc hmaccalc python-devel
newt-devel perl-ExtUtils-Embed pesign elfutils-devel zlib-devel
binutils-devel audit-libs-devel numactl-devel pciutils-devel ncurses-
devel python-docutils libselinux-devel
```
4. Install EPEL 7:
```
rpm -ivh https://dl.fedoraproject.org/pub/epel/epel-release-latest-
7.noarch.rpm
```

Note: EPEL7 is used because it contains quilt.

5. Install quilt:
```
yum -y install quilt
```

4.2 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 Lustre MASTER branch from HPDD git:
```
git clone git://git.hpdd.intel.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 the following:
```
$ sh ./autogen.sh
configure.ac:12: installing 'config/config.guess'
configure.ac:12: installing 'config/config.sub'
configure.ac:14: installing 'config/install-sh'
configure.ac:14: installing 'config/missing'
libcfs/libcfs/autoMakefile.am: installing 'config/depcomp'
$
```

4.3 Preparing the Kernel Source

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

1. Create the directory structure, get the source from the RPM, and 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 http://vault.centos.org/7.1.1503/os/Source/SPackages/kernel-3.10.0-229.el7.src.rpm
```
   CentOS periodically releases updates to their distributed kernel. The Lustre Master attempts to stay up-to-date with the most recent kernel from CentOS. In the event that the link above is not completely up-to-date, you should visit the CentOS 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:
```
...
#
# configuration written to .config
#
+ echo '# x86_64'
+ cat .config
+ find . '(' -name '*.orig' -o -name '*~' ')' -exec rm -f '{}' ';'
+ find . -name .gitignore -exec rm -f '{}' ';'
+ cd ..
+ exit 0
```
   At this point, we now have kernel source, with all the CentOS patches applied, residing in the directory
```
/home/build/kernel/rpmbuild/BUILD/kernel-<CentOS 7.1_kernel_version>.el7/linux-3.10.0-229.el7.centos.x86_64/
```
   For convenience we will set up a variable:
```
kpath=/home/build/kernel/rpmbuild/BUILD/kernel-3.10.0-229.el7/linux-<CentOS7.1_kernel_version>.el7.centos.x86_64/
```

4.4 Patching the Kernel Source with Kernel Patches from Lustre

1. Add a unique build ID so we can be certain our kernel is booted. Edit `$kpath/Makefile` and modify line 4, the `EXTRAVERSION` to read:
   ```
   EXTRAVERSION = .lustreopa
   ```
2. Enter the directory `$kpath`:
   ```
   cd $kpath
   ```
3. Overwrite the .config file:
   ```
   cp /home/build/lustre-release/lustre/kernel_patches/kernel_configs/
   kernel-3.10.0-3.10-rhel7-x86_64.config ./.config
   ```
4. Link the Lustre series and patches:
   ```
   ln -s ~/lustre-release/lustre/kernel_patches/series/3.10-rhel7.series
   series
   ln -s ~/lustre-release/lustre/kernel_patches/patches patches
   ```
5. Apply the patches to the kernel source using quilt:
   ```
   quilt push -av
   ...
   ...
   Hunk #1 succeeded at 25 (offset 5 lines).
   patching file drivers/scsi/isci/init.c
   patching file drivers/message/fusion/Kconfig
   patching file drivers/message/fusion/mptbase.h
   Hunk #1 succeeded at 166 (offset 1 line).
   Now at patch patches/blkdev_tunables-3.7.patch
   ```

4.5 Building the New Kernel as an RPM

1. Go into the kernel source directory and build a kernel rpm:
   ```
   cd $kpath
   make oldconfig || make menuconfig
   make rpm
   ```

   A successful build will return:
   ```
   ...
   ...
   Wrote: /home/build/kernel/rpmbuild/RPMS/x86_64/kernel-
   3.10.0.lustreopa-2.x86_64.rpm
   Wrote: /home/build/kernel/rpmbuild/RPMS/x86_64/kernel-headers-
   3.10.0.lustreopa-2.x86_64.rpm
   Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.dAcare
   + umask 022
   + cd /home/build/kernel/rpmbuild/BUILD
   + cd kernel-3.10.0.lustreopa
   + rm -rf /home/build/kernel/rpmbuild/BUILDROOT/kernel-
   3.10.0.lustreopa-2.x86_64
   + exit 0
   rm kernel-3.10.0.lustreopa.tar.gz kernel.spec
   ```

At this point, you will have a fresh kernel RPM
`~/kernel/rpmbuild/RPMS/x86_64/kernel-3.10.0.lustreopa-1.x86_64.rpm`

4.6 Installing the Lustre Kernel and Rebooting

1. As root, install the kernel:
   ```
   rpm -ivh --force ~build/kernel/rpmbuild/RPMS/x86_64/kernel-
   3.10.0.lustreopa-1.x86_64.rpm
   ```
   It is necessary to force the install of the kernel while https://jira.hpdd.intel.com/browse/LU-7176 is unresolved.
2. Create initrd using new-kernel-pkg:
   ```
   /sbin/new-kernel-pkg --package kernel --mkinitrd --dracut --depmod --
   install 3.10.0.lustreopa
   ```
3. Reboot.
4. From the initial login prompt, you can confirm the lustreopa patched kernel has booted using these commands:
   ```
   CentOS Linux 7 (Core)
   Kernel 3.10.0.lustreopa on an x86_64

   localhost login:
   ```

This concludes the first step giving us a booted k' in our three overview steps.

5 Configuring, Building, and Installing compat-rdma for I'

1. Create the directory structure, get the source from the RPM and create a .rpmmacros file to install the kernel source in our user dir.
```
cd $HOME
mkdir -p compat-rdma/rpmbuild/{BUILD,RPMS,SOURCES,SPECS,SRPMS}
cd compat-rdma
echo '%_topdir %(echo $HOME)/compat-rdma/rpmbuild' > ~/.rpmmacros
```
2. Install the compat-rdma source rpm. The compat-rdma source rpm is contained in the IntelOPA-IFS.RHEL71 archive available from http://download.intel.com.
3. Use rpmbuild to create a compat-rdma that is build for the Lustre patched kernel we have just build and booted:
```
rpmbuild --ba --define "KVERSION $(uname -r)" \
     --define "_version<RHEL7.1_latest_kernel>.el7.x86_64"
     --define "_release 5" \
--define "build_kernel_ib 1" \
--define "build_kernel_ib_devel 1" \
--define "configure_options --with-core-mod
--with-user_mad-mod \
--with-user_access-mod \
--with-addr_trans-mod \
--with-ipoib-mod \
--with-hfi1-mod" \
./SPECS/compat-rdma.spec
```
4. This will build compat-rdma and you will see a successful build ending with:
```
...
...
Wrote: /home/build/compat-rdma/rpmbuild/RPMS/x86_64/compat-rdma-
<RHEL7.1_latest_kernel>.el7.x86_64-5.x86_64.rpm
Wrote: /home/build/compat-rdma/rpmbuild/RPMS/x86_64/compat-rdma-
devel-<RHEL7.1_latest_kernel>.el7.x86_64-5.x86_64.rpm
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.quPXH8
+ umask 022
+ cd /home/build/compat-rdma/rpmbuild/BUILD
+ cd compat-rdma-3.10.0_229.el7.x86_64
+ rm -rf /home/build/compat-rdma/rpmbuild/BUILDROOT/compat-rdma-
<RHEL7.1_latest_kernel>.el7.x86_64-5.x86_64
+ rm -rf /home/build/compat-rdma/rpmbuild/BUILD/compat-rdma-
<RHEL7.1_latest_kernel>.el7.x86_64
+ exit 0
```
5. Install your newly build compat-rdma and compat-rdma-devel:
```
rpm -ivh /home/build/compat-rdma/rpmbuild/RPMS/x86_64/compat-rdma-
<RHEL7.1_latest_kernel>.el7.x86_64-5.x86_64.rpm
rpm -ivh /home/build/compat-rdma/rpmbuild/RPMS/x86_64/compat-rdma-
devel-<RHEL7.1_latest_kernel>.el7.x86_64-5.x86_64.rpm
```

This concludes building I'.

6 Configuring and Building Lustre

1. As root, create a symbolic link of the Module.symvers file for compat-rdma to where Lustre configure is expecting it.
   ```
   ln -s /usr/src/compat-rdma/include/Module.symvers  /usr/src/compat-rdma/
   ```
2. Configure Lustre source
   ```
   cd ~/lustre-release/
   ./configure --with-linux=/home/build/kernel/rpmbuild/BUILD/kernel-3.10.0.lustreopa --with-o2ib=/usr/src/compat-rdma/include
   ```
   The configure script will return successfully with:
   ```
   ...
   ...
   config.status: executing depfiles commands
   config.status: executing libtool commands
   CC:            gcc
   LD:            /usr/bin/ld -m elf_x86_64
   CPPFLAGS:      -include /home/build/lustre-release/config.h -I/home/build/lustre-release/libcfs/include -I/home/build/lustre-release/lnet/include -I/home/build/lustre-release/lustre/include
   CFLAGS:        -g -O2 -Wall -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
   Type 'make' to build Lustre.
   ```
3. make rpms:
   ```
   make rpms
   ```
   make rpms will conclude successfully with:
   ```
   ...
   ...
   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'
   ```

You should now have built the following, similarly named, rpms:

```
lustre-2.7.59-1_g703195a.src.rpm
lustre-2.7.59-3.10.0.lustreopa_g703195a.x86_64.rpm
lustre-debuginfo-2.7.59-3.10.0.lustreopa_g703195a.x86_64.rpm
lustre-iokit-2.7.59-3.10.0.lustreopa_g703195a.x86_64.rpm
lustre-modules-2.7.59-3.10.0.lustreopa_g703195a.x86_64.rpm
lustre-osd-ldiskfs-2.7.59-3.10.0.lustreopa_g703195a.x86_64.rpm
lustre-osd-ldiskfs-mount-2.7.59-3.10.0.lustreopa_g703195a.x86_64.rpm
lustre-source-2.7.59-3.10.0.lustreopa_g703195a.x86_64.rpm
lustre-tests-2.7.59-3.10.0.lustreopa_g703195a.x86_64.rpm
```

7 Installing e2fsprogs

A modified version of e2fsprogs is necessary to understand the Lustre storage on-disk format. This version is able to understand the ldiskfs format (based on ext4) that Lustre typically uses.

1. Download e2fsprogs from
   `http://downloads.hpdd.intel.com/public/e2fsprogs/latest/`
2. As root, install with
   ```
   rpm -Uvh ./e2fsprogs-1.42.13.wc3-4.el7.x86_64.rpm ./e2fsprogs-libs-
   1.42.13.wc3-4.el7.x86_64.rpm ./libcom_err-1.42.13.wc3-
   4.el7.x86_64.rpm ./libss-1.42.13.wc3-4.el7.x86_64.rpm
   ```

7.1 Install Lustre

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

Note: You may receive warnings during the installation as the package tries to establish symlinks for the new lustre modules into the old kernel modules tree. They do not affect the correct operation of Lustre and can be ignored.

7.2 Disable SELinux

SELinux, which is on by default in 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, vi /etc/selinux/config and change the line 'selinux=enforcing' to 'selinux=disabled'.
3. 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
   ```

7.3 Configuring Luster and LNet to Use the Intel® Omni-Path Architecture Network

1. On both machines, verify that OPA cards appear with their correct IP addresses on interface ib0

```
ib0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 2044
        inet 192.168.1.1  netmask 255.255.0.0  broadcast
192.168.255.255
        inet6 fe80::211:7501:16a:327b  prefixlen 64  scopeid
0x20<link>
Infiniband hardware address can be incorrect! Please read BUGS
section in ifconfig(8).
        infiniband
80:00:00:02:FE:80:00:00:00:00:00:00:00:00:00:00:00:00:00:00
txqueuelen 256  (InfiniBand)
        RX packets 12  bytes 672 (672.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

2. On both machines, map the ib0 interface to the o2ib LNet subnet by editing the `/etc/modprobe.d/lustre.conf` file.

```
# /etc/modprobe.d/lustre.conf
options lnet networks="o2ib0(ib0)"
```

7.4 Testing LNet

Lustre networking (LNet) can be tested in isolation. This can be useful to verify correct network operation. For more information, consult the Lustre Operations manual:

https://build.hpdd.intel.com/job/lustre-manual/lastSuccessfulBuild/artifact/lustre_manual.xhtml#lnetselftest

7.4.1 Testing Lustre

1. On the first machine (192.168.1.1), 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. On the second machine, mount Lustre from the first machine:
   ```
   mount  -t lustre 192.168.1.1@o2ib:/lustre /mnt
   ```

   You will now have a memory backed test Lustre filesystem available on the first machine at `/mnt/lustre`
3. For more details setting up and configuring Lustre, review the Lustre Operations Manual at https://build.hpdd.intel.com/job/lustre-manual/lastSuccessfulBuild/artifact/lustre_manual.xhtml

§