DOD Specific
Open|SpeedShop
Build and Installation Guide
Version 2.2.3
July 25, 2016

Introduction

This document supports the Open|SpeedShop 2.2 release and subsequent updates. The CBTF version of Open|SpeedShop is considered the primary release with the offline version of Open|SpeedShop once the default release is available as a backup version, as of this release. The CBTF version contains new experiments (memory, lightweight I/O, and lightweight MPI) that are not available in the offline version.

Open|SpeedShop release 2.2 comes with an install script (install-tool) that will build all the supporting components (also referred to as the krellroot) and Open|SpeedShop itself, in one step. Or the builder can invoke the script once to build the supporting component set (krellroot) and another time to build Open|SpeedShop.

The install script relies on arguments to the script instead of environment variables like the previous install script (install.sh). With install-tool, the environment variables are set internally to the script so that the builder is not required to keep track of the numerous build environment variables that Open|SpeedShop uses. In addition, more build environment variables are set by default, requiring less builder-required action.

The new install script also supports building the Component Based Tool Framework (CBTF) for the new Open|SpeedShop CBTF instrumentor version that is described below.

What is CBTF and how does it relate to Open|SpeedShop?

CBTF stands for Component Based Tool Framework and is a key scalability feature for the new version of Open|SpeedShop that uses CBTF as its instrumentation methodology. The Open|SpeedShop team is in the process of changing the mechanisms that gather the performance data and transfer the performance data from the application, where it was gathered, to the Open|SpeedShop client where it is stored into an SQLite database file. In order for Open|SpeedShop to operate efficiently at high rank and thread counts it could not continue to write raw data files to disk and then read them up at the end of the application execution. This is a bottleneck, which is too time-consuming at high processor counts. So, CBTF version of Open|SpeedShop using the new CBTF gathering and transfer mechanisms which transmit raw performance data to the client tool without writing files. This will be significantly faster than the offline version. CBTF can also be used independently from Open|SpeedShop to rapidly prototype tools. Please see the CBTF wiki at: https://github.com/OpenSpeedShop/cbtf/wiki for more information.

Some Initial Notes

- In addition to some of the open source components using the "cmake" build tools to build their components, the Open|SpeedShop team is now using cmake as the build tool for Open|SpeedShop, and the CBTF components that the team supplies. So, cmake is now a build pre-requisite package. Versions of cmake with a version number of 3.0 and above are favored. cmake version 2.12.2 is a minimum.
- Note: For version 2.2, please replace "**--build-cbtf**" with "**--build-cbtf-all**" in install-tool scripts from previous versions, as we are now using cmake as our underlying build tool instead of GNU autotools. With cmake we are building cbtf, cbtf-krell, cbtf-argonavis, and cbtf-lanl separately. With GNU autotools we built these in one build. So, now they are all built when "--build-cbtf-all" is used.
- **Caution:** When installing a new version, please install into an empty directory, as we do not clean the install directories out. You have end up with multiple copies of the same component, which can lead to execution issues.
- We refer to root or krell-root packages or building the krell-root. What does this mean? These are the external packages that Open|SpeedShop and/or CBTF uses as part of the tool. Sometimes one or several of these external packages are not available on the platform that Open|SpeedShop and/or CBTF is being installed on, so we have provided the ability to build and install them. The root/external packages are provided as a convenience. You could install these packages without the use of the krell-root on the install system, if desired.
- The OSS/CBTF version is the default and official version of Open|SpeedShop, as of this release. The offline version is still available, but is the OSS/CBTF version is default now.
- You can install Open|SpeedShop in non-standard locations (root permission is not needed)
- Versions of Open|SpeedShop 2.2 and beyond install using the information in this document.
- **NOTE: Open|SpeedShop is normally built with the GNU compilers, but now supports builds using the Intel compiler.** It does not build with PGI, Cray, or other compilers. Even though Open|SpeedShop only builds with GNU or Intel compilers, Open|SpeedShop supports performance analysis of applications built with a wide variety of compilers, including GNU, Intel, PGI, Pathscale, and Cray.
- There are a number of development environment packages that are required if you are starting from a clean install on your desktop or laptop. See the Prerequisite Packages section for the list of packages needed on non-development systems. Most laboratory systems normally have all the required packages installed, because most tools need these packages to build and execute successfully.
- There is now an option to build the krell root components that might be missing on the system or needed separately from Open|SpeedShop. The

builder can build the krell root then use that installation to build CBTF and/or Open|SpeedShop. Or, you can choose to build all components into the same install directory.

- The install-tool script supports building both Open|SpeedShop modes of execution:
    - offline – Write raw performance information to shared file system disk files while the application is executing. Then convert the raw data files to an Open|SpeedShop database file when the application completes execution.
    - cbtf – Performance information is transported from the application across a TCP/IP network as it is collected and stored in the Open|SpeedShop database on the fly.
- install-tool script options for building Open|SpeedShop are listed and described below:
    - To build the existing offline version use two invocations of the install-tool script individually, in the order listed.
        - "--build-krell-root"
        - "--build-offline"
    - To build the new CBTF version use three invocations of the install-tool script individually, in the order listed.
        - "--build-krell-root"
        - "--build-cbtf-all"
        - "--build-onlyosscbtf"

- The examples below are for illustration and will need adjustment based on the software installations on your particular system.
- On systems with heterogeneous front-end and compute node processor sets, multiple install-tool invocations are required. Examples of these types of systems are the BG/Q, Cray, Intel MIC systems, or on any other systems where targeted builds are needed. We suggest a separate set of builds, one set for the front-end viewer tool build and another set of builds for the compute node runtimes and performance data collectors build.
    - For building the client tool that runs on the front-end node:
        - "--build-krell-root"
        - "--build-offline"
    - For building the collectors and runtimes that execute on the compute node:
        - "--build-krell-root" using --target-arch=<platform>
        - "--build-offline" using --target-arch=<platform>

    - Note: It appears that some systems with heterogeneous front-end versus compute node processor types default to the compute node compilers. Most of our instructions assume the default compilers are front-end compilers. Putting "CXX=g++" and "CC=gcc" in front of the install-tool command when building the front-end version of Open|SpeedShop may get around this issue on platforms where the defaults are not what we expected (defaults to compute node compiler version).

Prerequisite Packages

There are some prerequisite packages that are necessary for building and running the Open|SpeedShop performance tool. Most will be present on a system that is used for debugging and performance analysis. This is true of most national laboratory clusters, Cray, and Blue Gene platforms. However, if you are installing on machines without the necessary tool supporting packages installed then this section might be helpful.

Ubuntu/Debian:

It is necessary to apply these packages to the base system installation, if they are not already installed, in order to successfully build Open|SpeedShop and the external tool specific packages required by Open|SpeedShop. Example required packages based on Ubuntu 16.04 install.

Packages to aid in development:

```
Offline: sudo apt-get install cvs git
CBTF:   sudo apt-get install git cmake cvs
```

Packages to aid in downloading and building:

```
sudo apt-get install flex bison libxml2-dev python-dev g++ make patch
environment-modules libz-dev binutils-dev libdwarf-dev libelf-dev
```

Packages that install-tool will build if you don't have the installed: (but makes build shorter if these are installed)

```
sudo apt-get install automake autoconf libtool m4 libltdl-dev(--build-
autotools)
sudo apt-get install binutils binutils-dev (--build-binutils)
sudo apt-get install libelf1 elfutils libelf-dev (--build-libelf)
sudo apt-get install sqlite3 (--build-sqlite)
```

If the package: qt3-apps-dev is available use it, if not, you may need these (libx11-dev libxext-dev) instead.

RedHat/Fedora:

It is necessary to apply a list of supporting packages to the base system installation, if they are not already installed, in order to successfully build Open|SpeedShop and the external tool specific packages required by Open|SpeedShop. This list is somewhat variable depending on what was installed during the initial installation and prior to attempting to install Open|SpeedShop. Previous experiences have resulted in this list of candidate packages:

```
Offline and CBTF:
    yum install -y rpm-build \
        gcc gcc-c++ \
        openmpi \
        patch \
        autoconf automake \
        elfutils-libelf elfutils-libelf-devel \
        libxml2 libxml2-devel \
        binutils binutils-devel \
        python python-devel \
        flex bison bison-devel bison-runtime \
        libtool libtool-ltdl libtool-ltdl-devel cmake
CBTF:
    yum install -y cmake git
```

SLES/SUSE:

It is necessary to apply these packages to the base system installation, if they are not already installed, in order to successfully build Open|SpeedShop and the external tool specific packages required by Open|SpeedShop.

Packages to aid in development:
    CBTF: Modules git
Packages to aid in downloading and building:
    Offline and CBTF: flex bison rpm libxml2 libxml2-dev python-dev g++ make patch cmake
    CBTF: cmake

Packages that install-tool will build if you don't have the installed: (but makes the build shorter if these are installed)
    qt3 qt3-devel

If your system was installed with any development package group, then the need for extra package installs may be reduced significantly.

Using the install-tool that comes with the Open|SpeedShop release will install many of the key open source tool related components, but the above mentioned system components are required to be installed by the system administrator.

Building from the Release Tarballs (located on sourceforge)

The release top-level directory contains the install script (install-tool) that is intended to make the build and installation of the external tool support components (known as the krell root) that are needed to support Open|SpeedShop and Open|SpeedShop itself easier.

Because Open|SpeedShop depends on components that are highly dependent on operating system interfaces and libraries, it is difficult to produce executable rpms for every installation platform. This is the reason why we offer the source build installation instead of rpms.

Installation Information

Please gunzip and untar the openspeedshop-release-2.2.3 tar.gz file and change directory into the openspeedshop-release-2.2 directory.

For example:

```
tar -xzvf openspeedshop-release-2.2.3.tar.gz
cd openspeedshop-release-2.2
```

Inside the top-level release directory is the key script, install-tool that can be used in building the Open|SpeedShop performance tool.

The script, install-tool has a help option:

"install-tool –-help"

That can provide information about the possible options a builder of Open|SpeedShop could use.

The typical build for the offline version of Open|SpeedShop is done with an install line something like this:

```
./install-tool --build-krell-root
               --krell-root-prefix /opt/krellroot_v2.2.3
               --with-openmpi /opt/openmpi-1.8.2

./install-tool   --build-offline
                 --openss-prefix /opt/ossoffline_v2.2.3
                 --krell-root-prefix /opt/krellroot_v2.2.3
                 --with-openmpi /opt/openmpi-1.8.2
```

The first install line above builds all the external (krell root) packages that Open|SpeedShop needs and installs them in /opt/krellroot_v2.2.3. The second install line above uses those external packages to build Open|SpeedShop. It also uses the openmpi MPI implementation to build the Open|SpeedShop MPI experiment collectors.

Hint:
To get the paths for the "--with-mpt" or other mpi or other "--with" option clauses, one can do:
1) module load mpi-sgi/mpt.2.12r26
2) echo $LD_LIBRARY_PATH
3) Look for the MPT path (up to the "/lib" and use that in the "--with-mpt" clause in the install-tool command.

Example: echo $LD_LIBRARY_PATH
/p/home/apps/sgi/mpt-2.12-sgi712r26/lib:/p/home/galarowi/python_root/lib

Use this --with-mpt clause: "--with-mpt /p/home/apps/sgi/mpt-2.12-sgi712r26"

However, on some platforms the typical build install line is not adequate. Platforms like the Cray and Blue Gene requires more options to the install-tool script. If you are familiar with the previous install.sh script method of building Open|SpeedShop, in that build scenario extra environment variables needed to be set. With install-tool, we need additional arguments to the script to specify things like the target platform, so there is a --target-arch <target> option available in the install-tool script. For example, to build on a Cray system, this is an example build line:

```
./install-tool   --build-krell-root –target-arch cray
                 --krell-root-prefix /tmp/work/<userid>/krellroot_v2.2.3
                 --with-boost <boost install directory>
                 --with-mpich2 /opt/cray/mpt/default/gni/mpich2-gnu/47
```

```
./install-tool   --build-offline –target-arch cray
                 --openss-prefix /tmp/work/<userid>/ossoffline_v2.2.3
                 --krell-root-prefix /tmp/work/<userid>/krellroot_v2.2.3
                 --with-boost <boost install directory>
                 --with-mpich2 /opt/cray/mpt/default/gni/mpich2-gnu/47
```

One can specify their own versions of the components needed by Open|SpeedShop to build properly or you can rely on the defaults that are installed on the system and used automatically by the build script. If the default system installed component is determined to be not adequate for the Open|SpeedShop build (missing development headers) then the install script will build a version of that component into the externals/root directory. If the builder has a specific version of a component, such as, PAPI, they may use the "--with-papi" option to specify the path to that installation and Open|SpeedShop will be built using that installation of PAPI.

One can also build individual components and specify the location to place the component. Here is the "install-tool --help" output:

```
$
$ ./install-tool --help
usage: ./install-tool [option]

--help, -h
  This help text.

Introduction:
  This install script can be used to build the krell externals package (--build-krell-root),
  the CBTF components and libraries (--build-cbtf-all), and/or the two modes/versions of
Open|SpeedShop
  (offline mode: --build-offline or cbtf mode: --build-onlyosscbtf).

  Typical usages examples are followed by the option descriptions. More examples and
explanations
  can be found in the Build and Install Guides for CBTF and Open|SpeedShop.

  The build can use rpm files (default) or build directly from tarballs (--no-rpm option).
  Building with the rpm default requires rpm and rpmbuild to be installed.
```

Typical usage examples:

```
 # Build only the krell-root
 ./install-tool --build-krell-root
             --krell-root-prefix /opt/krellroot_v2.2.3
             --with-openmpi /opt/openmpi-1.8.2

 # Build cbtf components using the krell-root
 ./install-tool --build-cbtf-all
             --cbtf-prefix /opt/cbtf_only_v2.2.3
             --krell-root-prefix /opt/krellroot_v2.2.3
             --with-openmpi /opt/openmpi-1.8.2
             --with-cupti /usr/local/cuda-6.5/extras/CUPTI
             --with-cuda /usr/local/cuda-6.5

 # Build only OSS using the cbtf components and the krell-root
 ./install-tool --build-onlyosscbtf
             --cbtf-prefix /opt/cbtf_only_v2.2.3
             --krell-root-prefix /opt/krellroot_v2.2.3
             --openss-prefix /opt/osscbtf_v2.2.3
             --with-openmpi /opt/openmpi-1.8.2
             --with-cupti /usr/local/cuda-6.5/extras/CUPTI
             --with-cuda /usr/local/cuda-6.5

 # Build the krell-root, the cbtf components, and OSS using cbtf instrumentor
 ./install-tool --build-all
             --cbtf-prefix /opt/cbtf_only_v2.2.3
             --krell-root-prefix /opt/krellroot_v2.2.3
             --openss-prefix /opt/osscbtf_v2.2.3
             --with-openmpi /opt/openmpi-1.8.2


 # Build OSS using offline instrumentor with and without krell-root
 # With krell-root
 ./install-tool --build-offline
             --krell-root-prefix /opt/krellroot_v2.2.3
             --openss-prefix /opt/ossoffline_v2.2.3
             --with-openmpi /opt/openmpi-1.8.2

 # Without krell-root
 ./install-tool --build-offline
             --openss-prefix /opt/ossoffline_v2.2.3
             --with-openmpi /opt/openmpi-1.8.2
```

Option Descriptions:

--krell-root-prefix <directory>
 Where <directory> is the location to install the components
 needed for building CBTF and Open|SpeedShop
 and it's supporting tools and libraries. The default

```
  is /opt/KRELLROOT. It is not recommended to use /usr.
  NOTE: This option will override any setting for
  the environment variable KRELL_ROOT_PREFIX.

--cbtf-prefix <directory>
  Where <directory> is the location to install CBTF
  and it's supporting tools and libraries. The default
  is /opt/CBTF. It is not recommended to use /usr.
  NOTE: This option will override any setting for
  the environment variable CBTF_PREFIX.

--openss-prefix <directory>
  Where <directory> is the location to install Open|SpeedShop
  and it's supporting tools and libraries. The default
  is /opt/OSS. It is not recommended to use /usr.
  NOTE: This option will override any setting for
  the environment variable OPENSS_PREFIX.

--build-all
  Build the krell-root, cbtf components, and openspeedshop
  The cbtf-prefix, krell-root-prefix, and the openss-prefix must be specified

--build-osscbtf
  Build the krell-root, cbtf components, and openspeedshop
  The cbtf-prefix, krell-root-prefix, and the openss-prefix must be specified

--build-cbtf-all
  Build only the cbtf components using the existing krell-root
  The krell-root-prefix and cbtf-install-prefix must be specified

--build-krell-root
  Build only the krell-root. The krell-root-prefix must be specified

--build-onlyosscbtf )
  Build only the Open|SpeedShop component for the cbtf instrumentor
  using the cbtf and krell-root components.

--build-onlyossoffline)
  Build only the Open|SpeedShop component for the offline instrumentor
  using the openss-prefix root component or the krell-root components (if krell-root-prefix
specified.)

  Specify a target architecture
--target-arch <target>
  Where acceptable target values are: cray, mic, bgp, bgq, bgqfe, arm

--no-rpm
  Build directly from source tarballs

  Use these MPI installations when building
```

```
--with-mpich <directory>
--with-mpich2 <directory>
--with-mpich2-driver <driver name>
--with-mpich <directory>
--with-mvapich2 <directory>
--with-mvapich2-driver <driver name>
--with-openmpi <directory>
--with-mpt <directory>
  Where <directory> is the installed path to the mpi implementation
  to use for MPI support.

  Build only the component specified by the build clause.
  The krell-root-prefix must be specified and components will
  be installed into that krell-root-prefix specified directory path.
--build-autotools
--build-bison
--build-boost
--build-binutils
--build-cmake
--build-dyninst
--build-libelf
--build-libdwarf
--build-libmonitor
--build-libunwind
--build-mrnet
--build-ompt
--build-papi
--build-python
--build-sqlite
--build-symtabapi
--build-ptgf
--build-ptgfossgui
--build-qcustomplot
--build-qgraphviz
--build-qt3
--build-serene
--build-vampirtrace
--build-xercesc

  Force these components to be built and installed into the krellroot
  or OSS offline/online install directories.
--force-binutils-build
--force-boost-build
--force-libelf-build
--force-libdwarf-build
--force-libunwind-build
--force-papi-build
--force-qt3-build
--force-sqlite-build
--force-xercesc-build
```

Build all of the above by force
--force-all

--skip-binutils-build
--skip-boost-build
--skip-dyninst-build
--skip-libdwarf-build
--skip-libelf-build
--skip-libunwind-build
--skip-mrnet-build
--skip-papi-build
--skip-symtabapi-build
--skip-vampirtrace-build
--skip-qt3-build

Use these non-root or alternative components when building
--with-binutils <directory>
--with-boost <directory>
--with-dyninst <directory>
--with-libelf <directory>
--with-libdwarf <directory>
--with-libmonitor <directory>
--with-libunwind <directory>
--with-ltdl <directory>
--with-mrnet <directory>
--with-papi <directory>
--with-python <directory>
--with-qt3 <directory>
--with-sqlite <directory>k
--with-symtabapi <directory>
--with-xercesc <directory>
Where <directory> is the installed path to the alternative component.

Use these non-root or alternative compute node components when building a cbtf-krell fe that points to targeted runtimes (cray, mic platforms)
--with-cn-cbtf <directory>
--with-cn-cbtf-krell <directory>
--with-cn-binutils <directory>
--with-cn-dyninst <directory>
--with-cn-libelf <directory>
--with-cn-libdwarf <directory>
--with-cn-libmonitor <directory>
--with-cn-libunwind <directory>
--with-cn-mrnet <directory>
--with-cn-symtabapi <directory>
--with-cn-xercesc <directory>
--with-cn-papi <directory>
--with-cn-boost <directory>
Where <directory> is the installed path to the alternative component.

--with-tls < explicit | implicit >
  where the default is implicit

  Only build ptgf gui related components, not qt3 related
--use-only-ptgf

Optional install-tool arguments are needed for configuring Open|SpeedShop for MPI experiments. If the installation of Open|SpeedShop is intended to support running MPI specific Open|SpeedShop experiments, then one or more MPI implementation arguments must be specified. A MPI argument is necessary if the Open|SpeedShop build is intended to support running VampirTrace for the mpiotf experiment, because VampirTrace needs a MPI implementation specified in order to be built. The Open|SpeedShop install-tool script will build MPI experiment collectors for one or more of these MPI implementations by specifying one or more of the "--with-<mpi implementation> arguments:

```
For openMPI:
 --with-openmpi <openmpi install path>
For mpich:
--with-mpich <mpich install path>
For mpich2:
--with-mpich2 <mpich2 install path>
For SGI mpt[1]:
--with-mpt <mpt install path>
For mvapich
--with-mvapich <mvapich install path>
For mvapich2:
--with-mvapich2 <mvapich2 install path>
```

If none of the above arguments are not specified, every non-MPI specific Open|SpeedShop experiments will be built and execute properly, but the mpi, mpit, mpiotf experiments will not be built. The MPI implementation arguments are not necessary to run pcsamp, usertime, hwc, io, or fpe and variants of those experiments, even on MPI applications. They are only needed for mpi, mpit, and mpiotf experiment creation because the tool needs to know the MPI data structure definitions to process MPI performance data.

The install-tool script and the Open|SpeedShop configuration code will operate on those MPI arguments and will configure Open|SpeedShop to recognize these MPI implementations. When the MPI specific argument collectors are built and installed, users will have the ability to create MPI experiments (mpi, mpit, mpiotf) and gather

[1] MPT is transitioning to MPI-3 support but it is not completed. Please use MPT-2.08 and above when building the MPI collectors. We have added special code to support their implementation of mpi.h MPI Function specifications with our wrappers.

performance data for MPI applications built with those specific MPI implementations.

Install tool example commands from DOD systems

These examples show optional ways of building both the CBTF version and the offline version of Open|SpeedShop.

When building the offline version of Open|SpeedShop you can build the supporting components (krell root) and Open|SpeedShop tool separately. Or you can build the offline version of Open|SpeedShop into the same install directory and not build the krell root separately. Using the option "--build-offline" without the "--krell-root-prefix" option will build the offline version of Open|SpeedShop and the root components into the same directory specified by --openss-prefix"

Building the CBTF based Open|SpeedShop version cannot be done in a similar fashion (one command to build all parts), because the krell root components, the cbtf components, and the Open|SpeedShop client components are installed into separate install directories by design.

Armstrong armstrong.navydsrc.hpc.mil NAVY

```
#!/bin/bash

# ----------------------------
# First Build the compute node version (root and oss versions if you use the
recommended krell externals root build):
# For COMPUTE NODE builds:
# ----------------------------------

. /opt/modules/default/init/bash

module unload PrgEnv-cray PrgEnv-pgi PrgEnv-gnu PrgEnv-intel
module load PrgEnv-gnu gcc/4.8.2
export CC=gcc
export CXX=g++
export BASE_IDIR=/app/PET/pkgs/openss
export TOOL_VERS="_v2.2.3"
export CMAKE_IDIR=${CMAKE_IDIR}/cmake-3.2.2
export PATH=${CMAKE_IDIR}/bin:$PATH
export MPICH_IDIR=/opt/cray/mpt/6.3.1/gni/mpich2-gnu/48
export PAPI_IDIR=/opt/cray/papi/5.3.2
export BINUTILS_IDIR=/opt/cray/xc-sysroot/default/usr
export KROOT_IDIR=${BASE_IDIR}/krellroot${TOOL_VERS}
export CBTF_IDIR=${BASE_IDIR}/cbtf${TOOL_VERS}
export OSSCBTF_IDIR=${BASE_IDIR}/osscbtf${TOOL_VERS}
export OSSOFF_IDIR=${BASE_IDIR}/ossoff${TOOL_VERS}
export PYTHON_IDIR=${BASE_IDIR}/python-2.7.3
export ALPS_IDIR=/opt/cray/alps/5.2.1-2.0502.9041.11.6.ari

# Build compute node tool portions

./install-tool --runtime-only --target-arch cray --target-shared --build-krell-root --krell-
root-prefix ${KROOT_IDIR}/compute --with-mpich ${MPICH_IDIR} --with-papi
${PAPI_IDIR} --with-binutils ${BINUTILS_IDIR} --with-alps ${ALPS_IDIR}

./install-tool --runtime-only --target-arch cray --target-shared --build-offline --openss-
prefix ${OSSOFF_IDIR}/compute --krell-root-prefix ${KROOT_IDIR}/compute --with-
mpich ${MPICH_IDIR} --with-papi ${PAPI_IDIR} --with-binutils ${BINUTILS_IDIR}

./install-tool --build-cbtf-all --runtime-only --target-arch cray --target-shared --cbtf-prefix
${CBTF_IDIR}/compute --krell-root-prefix  ${KROOT_IDIR}/compute --with-mpich
${MPICH_IDIR} --with-papi ${PAPI_IDIR} --with-alps ${ALPS_IDIR}


# -----------------------------
# Next build the login node version (root and oss versions if you use the recommended
krell externals root build):
# For LOGIN NODE builds:
# ---------------------------------

module unload PrgEnv-cray PrgEnv-gnu craype-ivybridge
module load craype-target-native
module load gcc/4.8.2
export LD_LIBRARY_PATH=${PYTHON_IDIR}/lib:$LD_LIBRARY_PATH
export PATH={CMAKE_IDIR}/bin:$PATH
export CC=gcc
export CXX=g++


# Build python because I didn't see one that was available

./install-tool --build-cmake --krell-root-prefix ${CMAKE_IDIR}
./install-tool --build-python --krell-root-prefix ${PYTHON_IDIR}

export PATH={CMAKE_IDIR}/bin:$PATH
export LD_LIBRARY_PATH=${PYTHON_IDIR}/lib:$LD_LIBRARY_PATH
```

```
./install-tool --build-krell-root --krell-root-prefix ${KROOT_IDIR} --with-mpich
${MPICH_IDIR} --with-python ${PYTHON_IDIR} --with-papi ${PAPI_IDIR} --force-binutils-
build --with-alps ${ALPS_IDIR}

./install-tool --runtime-target-arch cray --build-cbtf-all --cbtf-prefix ${CBTF_IDIR} --krell-
root-prefix ${KROOT_IDIR} --with-mpich ${MPICH_IDIR} --with-cn-boost
${KROOT_IDIR}/compute --with-cn-mrnet ${KROOT_IDIR}/compute --with-cn-xercesc
${KROOT_IDIR}/compute --with-cn-libmonitor ${KROOT_IDIR}/compute --with-cn-
libunwind ${KROOT_IDIR}/compute --with-cn-dyninst ${KROOT_IDIR}/compute --with-
cn-papi ${PAPI_IDIR} --with-cn-cbtf-krell ${KROOT_IDIR}/compute --with-cn-cbtf
${CBTF_IDIR}/compute --with-binutils ${KROOT_IDIR} --with-boost ${KROOT_IDIR} --
with-mrnet ${KROOT_IDIR} --with-xercesc ${KROOT_IDIR} --with-libmonitor
${KROOT_IDIR} --with-libunwind ${KROOT_IDIR} --with-dyninst ${KROOT_IDIR} --with-
papi ${PAPI_IDIR} --with-alps ${ALPS_IDIR}

./install-tool --build-offline --target-arch cray --openss-prefix ${OSSOFF_IDIR} --with-
runtime-dir ${OSSOFF_IDIR}/compute --krell-root-prefix ${KROOT_IDIR} --with-mpich
${MPICH_IDIR} --with-python ${PYTHON_IDIR} --with-papi ${PAPI_IDIR}

./install-tool --target-arch cray --build-onlyosscbtf --openss-prefix ${OSSCBTF_IDIR} --
with-cn-cbtf-krell ${CBTF_IDIR}/compute --krell-root-prefix ${KROOT_IDIR} --with-mpich
${MPICH_IDIR} --with-boost ${KROOT_IDIR} --with-mrnet ${KROOT_IDIR} --with-xercesc
${KROOT_IDIR} --with-libmonitor ${KROOT_IDIR} --with-libunwind ${KROOT_IDIR} --
with-dyninst ${KROOT_IDIR} --with-libelf ${KROOT_IDIR} --with-libdwarf ${KROOT_IDIR}
--with-binutils ${KROOT_IDIR} --cbtf-prefix ${CBTF_IDIR} --with-papi ${PAPI_IDIR}

# Compiling with a module loaded non-default installed compiler requires that the
libstc++ library be made available on the compute nodes
cp /opt/gcc/4.8.2/snos/lib64/libstdc++.so.6 ${KROOT_IDIR}/lib64/libstdc++.so.6
chmod 755 ${KROOT_IDIR}/lib64/libstdc++.so.6

cp /opt/gcc/4.8.2/snos/lib64/libstdc++.so.6
${KROOT_IDIR}/compute/lib64/libstdc++.so.6
chmod 755 ${KROOT_IDIR}/compute/lib64/libstdc++.so.6
```

Conrad   **conrad.navydsrc.hpc.mil**   **NAVY**

```
#!/bin/bash

# -----------------------------
# First Build the compute node version (root and oss versions if you use the
recommended krell externals root build):
```

```
# For COMPUTE NODE builds:
# ---------------------------------

. /opt/modules/default/init/bash

module unload PrgEnv-cray PrgEnv-gnu craype-haswell
module load craype-target-native
module load gcc/4.9.2

export BASE_IDIR=/p/home/app/PET/pkgs/openss
export CMAKE_IDIR=${BASE_IDIR}/cmake-3.2.2

./install-tool --build-cmake --krell-root-prefix ${CMAKE_IDIR}
export PATH={CMAKE_IDIR}/bin:$PATH

module unload PrgEnv-cray PrgEnv-pgi PrgEnv-gnu PrgEnv-intel
module load PrgEnv-gnu gcc/4.9.2
module unload craype-target-native
module load craype-haswell

export CC=gcc
export CXX=g++
export TOOL_VERS="_v2.2.3"
export CMAKE_IDIR=${BASE_IDIR}/cmake-3.2.2
export PATH=${CMAKE_IDIR}/bin:$PATH
export MPICH_IDIR=/opt/cray/mpt/7.3.2/gni/mpich-gnu/5.1
export PAPI_IDIR=/opt/cray/papi/5.4.3.1
export BINUTILS_IDIR=/opt/cray/xc-sysroot/default/usr
export KROOT_IDIR=${BASE_IDIR}/krellroot${TOOL_VERS}
export CBTF_IDIR=${BASE_IDIR}/cbtf${TOOL_VERS}
export OSSCBTF_IDIR=${BASE_IDIR}/osscbtf${TOOL_VERS}
export OSSOFF_IDIR=${BASE_IDIR}/ossoff${TOOL_VERS}
export ALPS_IDIR=/opt/cray/alps/5.2.1-2.0502.9072.13.1.ari

./install-tool --runtime-only --target-arch cray --target-shared --build-krell-root --krell-
root-prefix ${KROOT_IDIR}/compute --with-mpich ${MPICH_IDIR} --with-papi
${PAPI_IDIR} --with-binutils ${BINUTILS_IDIR} --with-alps ${ALPS_IDIR}

./install-tool --build-cbtf-all --runtime-only --target-arch cray --target-shared --cbtf-prefix
${CBTF_IDIR}/compute --krell-root-prefix  ${KROOT_IDIR}/compute --with-mpich
${MPICH_IDIR} --with-papi ${PAPI_IDIR} --with-alps ${ALPS_IDIR}

# ----------------------------
# Next build the login node version (root and oss versions if you use the recommended
```

krell externals root build):
# For LOGIN NODE builds:

```
module unload PrgEnv-cray PrgEnv-gnu craype-haswell
module load craype-target-native
module load gcc/4.9.2
export PATH={CMAKE_IDIR}/bin:$PATH
export CC=gcc
export CXX=g++

./install-tool --build-krell-root --krell-root-prefix ${KROOT_IDIR} --with-mpich
${MPICH_IDIR} --with-papi ${PAPI_IDIR} --force-binutils-build --with-alps ${ALPS_IDIR}

./install-tool --runtime-target-arch cray --build-cbtf-all --cbtf-prefix ${CBTF_IDIR} --krell-
root-prefix ${KROOT_IDIR} --with-mpich ${MPICH_IDIR} --with-cn-boost
${KROOT_IDIR}/compute --with-cn-mrnet ${KROOT_IDIR}/compute --with-cn-xercesc
${KROOT_IDIR}/compute --with-cn-libmonitor ${KROOT_IDIR}/compute --with-cn-
libunwind ${KROOT_IDIR}/compute --with-cn-dyninst ${KROOT_IDIR}/compute --with-
cn-papi ${PAPI_IDIR} --with-cn-cbtf-krell ${KROOT_IDIR}/compute --with-cn-cbtf
${CBTF_IDIR}/compute --with-binutils ${KROOT_IDIR} --with-boost ${KROOT_IDIR} --
with-mrnet ${KROOT_IDIR} --with-xercesc ${KROOT_IDIR} --with-libmonitor
${KROOT_IDIR} --with-libunwind ${KROOT_IDIR} --with-dyninst ${KROOT_IDIR} --with-
papi ${PAPI_IDIR} --with-alps ${ALPS_IDIR}

./install-tool --target-arch cray --build-onlyosscbtf --openss-prefix ${OSSCBTF_IDIR} --
with-cn-cbtf-krell ${CBTF_IDIR}/compute --krell-root-prefix ${KROOT_IDIR} --with-mpich
${MPICH_IDIR} --with-boost ${KROOT_IDIR} --with-mrnet ${KROOT_IDIR} --with-xercesc
${KROOT_IDIR} --with-libmonitor ${KROOT_IDIR} --with-libunwind ${KROOT_IDIR} --
with-dyninst ${KROOT_IDIR} --with-libelf ${KROOT_IDIR} --with-libdwarf ${KROOT_IDIR}
--with-binutils ${KROOT_IDIR} --cbtf-prefix ${CBTF_IDIR} --with-papi ${PAPI_IDIR}

./install-tool --build-offline --openss-prefix ${OSSOFF_IDIR} --with-runtime-dir
${OSSOFF_IDIR}/compute --krell-root-prefix ${KROOT_IDIR} --with-mpich ${MPICH_IDIR}
--with-papi ${PAPI_IDIR}

# Compiling with a module loaded non-default installed compiler requires that the
libstc++ library be made available on the compute nodes
cp /opt/gcc/4.9.2/snos/lib64/libstdc++.so.6 ${KROOT_IDIR}/lib64/libstdc++.so.6
chmod 755 ${KROOT_IDIR}/lib64/libstdc++.so.6
cp /opt/gcc/4.9.2/snos/lib64/libstdc++.so.6
${KROOT_IDIR}/compute/lib64/libstdc++.so.6
chmod 755 ${KROOT_IDIR}/compute/lib64/libstdc++.so.6
```

Copper copper0[1-2].ors.hpc.mil ORS

```
#!/bin/bash

# ----------------------------
# First Build the compute node version (root and oss versions if you use the
recommended krell externals root build):
# For COMPUTE NODE builds:
# ---------------------------------

. /opt/modules/default/init/bash

module unload PrgEnv-cray PrgEnv-gnu craype-interlagos
module load craype-target-native
module load gcc/4.9.2

export BASE_IDIR=/usr/local/usp/PETtools/CE/pkgs/openss
export CMAKE_IDIR=${BASE_IDIR}/cmake-3.2.2

./install-tool --build-cmake --krell-root-prefix ${CMAKE_IDIR}
export PATH={CMAKE_IDIR}/bin:$PATH

module unload PrgEnv-cray PrgEnv-pgi PrgEnv-gnu PrgEnv-intel
module load PrgEnv-gnu gcc/4.9.2
module unload craype-target-native
module load craype-interlagos

export cc=gcc
export CC=gcc
export CXX=g++
export TOOL_VERS="_v2.2.3"
export CMAKE_IDIR=${BASE_IDIR}/cmake-3.2.2
export PATH=${CMAKE_IDIR}/bin:$PATH
export MPICH_IDIR=/opt/cray/mpt/7.3.0/gni/mpich-gnu/5.1
export PAPI_IDIR=/opt/cray/papi/5.3.2.1
export BINUTILS_IDIR=/opt/cray/xe-sysroot/default/usr
export KROOT_IDIR=${BASE_IDIR}/krellroot${TOOL_VERS}
export CBTF_IDIR=${BASE_IDIR}/cbtf${TOOL_VERS}
export OSSCBTF_IDIR=${BASE_IDIR}/osscbtf${TOOL_VERS}
export OSSOFF_IDIR=${BASE_IDIR}/ossoff${TOOL_VERS}
export ALPS_IDIR=/opt/cray/alps/5.2.4-2.0502.9774.31.12.gem

# Build compute node tool libraries and executables
```

```
./install-tool --runtime-only --target-arch cray --target-shared --build-krell-root --krell-
root-prefix ${KROOT_IDIR}/compute --with-mpich ${MPICH_IDIR} --with-papi
${PAPI_IDIR} --with-binutils ${BINUTILS_IDIR} --with-alps ${ALPS_IDIR}

./install-tool --runtime-only --target-arch cray --target-shared --build-offline --openss-
prefix ${OSSOFF_IDIR}/compute --krell-root-prefix ${KROOT_IDIR}/compute --with-
mpich ${MPICH_IDIR} --with-papi ${PAPI_IDIR} --with-binutils ${BINUTILS_IDIR}

./install-tool --build-cbtf-all --runtime-only --target-arch cray --target-shared --cbtf-prefix
${CBTF_IDIR}/compute --krell-root-prefix  ${KROOT_IDIR}/compute --with-mpich
${MPICH_IDIR} --with-papi ${PAPI_IDIR}

# -----------------------------
# Next build the login node version (root and oss versions if you use the recommended
krell externals root build):
# For LOGIN NODE builds:
# ---------------------------------

module unload PrgEnv-cray PrgEnv-gnu craype-interlagos
module load craype-target-native
module load gcc/4.9.2
export PATH={CMAKE_IDIR}/bin:$PATH
export CC=gcc
export CXX=g++

./install-tool --build-krell-root --krell-root-prefix ${KROOT_IDIR} --with-mpich
${MPICH_IDIR} --with-papi ${PAPI_IDIR} --force-binutils-build --with-alps ${ALPS_IDIR}

./install-tool --runtime-target-arch cray --build-cbtf-all --cbtf-prefix ${CBTF_IDIR} --krell-
root-prefix ${KROOT_IDIR} --with-mpich ${MPICH_IDIR} --with-cn-boost
${KROOT_IDIR}/compute --with-cn-mrnet ${KROOT_IDIR}/compute --with-cn-xercesc
${KROOT_IDIR}/compute --with-cn-libmonitor ${KROOT_IDIR}/compute --with-cn-
libunwind ${KROOT_IDIR}/compute --with-cn-dyninst ${KROOT_IDIR}/compute --with-
cn-papi ${PAPI_IDIR} --with-cn-cbtf-krell ${KROOT_IDIR}/compute --with-cn-cbtf
${CBTF_IDIR}/compute --with-binutils ${KROOT_IDIR} --with-boost ${KROOT_IDIR} --
with-mrnet ${KROOT_IDIR} --with-xercesc ${KROOT_IDIR} --with-libmonitor
${KROOT_IDIR} --with-libunwind ${KROOT_IDIR} --with-dyninst ${KROOT_IDIR} --with-
papi ${PAPI_IDIR} --with-alps ${ALPS_IDIR}

./install-tool --target-arch cray --build-onlyosscbtf --openss-prefix ${OSSCBTF_IDIR} --
with-cn-cbtf-krell ${CBTF_IDIR}/compute --krell-root-prefix ${KROOT_IDIR} --with-mpich
${MPICH_IDIR} --with-boost ${KROOT_IDIR} --with-mrnet ${KROOT_IDIR} --with-xercesc
${KROOT_IDIR} --with-libmonitor ${KROOT_IDIR} --with-libunwind ${KROOT_IDIR} --
```

```
with-dyninst ${KROOT_IDIR} --with-libelf ${KROOT_IDIR} --with-libdwarf ${KROOT_IDIR}
--with-binutils ${KROOT_IDIR} --cbtf-prefix ${CBTF_IDIR} --with-papi ${PAPI_IDIR}

./install-tool --build-offline --openss-prefix ${OSSOFF_IDIR} --with-runtime-dir
${OSSOFF_IDIR}/compute --krell-root-prefix ${KROOT_IDIR} --with-mpich ${MPICH_IDIR}
--with-papi ${PAPI_IDIR}

# Compiling with a module loaded non-default installed compiler requires that the
libstc++ library be made available on the compute nodes
cp /opt/gcc/4.9.2/snos/lib64/libstdc++.so.6 ${KROOT_IDIR}/lib64/libstdc++.so.6
chmod 755 ${KROOT_IDIR}/lib64/libstdc++.so.6
cp /opt/gcc/4.9.2/snos/lib64/libstdc++.so.6
${KROOT_IDIR}/compute/lib64/libstdc++.so.6
chmod 755 ${KROOT_IDIR}/compute/lib64/libstdc++.so.6
```

Excalibur excalibur.arl.hpc.mil ARL

```
#!/bin/bash

. /opt/modules/3.2.10.3/init/bash

module unload PrgEnv-pgi PrgEnv-cray PrgEnv-intel
module load PrgEnv-gnu
module unload gcc; module load gcc/4.9.3
export SYSROOT_DIR=/opt/cray/xc-sysroot/default

export CC=gcc
export CXX=g++

BASE_DIR=/p/app/pet/pkgs/openss
TOOL_VERS="_v2.2.3"
ALPS_ROOT=/opt/cray/alps/5.2.4-2.0502.9774.31.11.ari
MPICH_ROOT=/opt/cray/mpt/7.3.2/gni/mpich-gnu/5.1
PAPI_ROOT=/opt/cray/papi/default
PYTHON_IROOT=${BASE_DIR}/python-2.7.3
PYTHON_VERS=2.7

# Build root components runtime only
echo  "Build root components runtime only"

echo "Step 1 ./install-tool --runtime-only --target-arch cray --target-shared --build-krell-
root --krell-root-install-prefix ${BASE_DIR}/krellroot${TOOL_VERS}/compute --with-
mpich ${MPICH_ROOT} --with-papi ${PAPI_ROOT} --with-alps ${ALPS_ROOT}"
```

```
./install-tool --runtime-only --target-arch cray --target-shared --build-krell-root --krell-root-install-prefix ${BASE_DIR}/krellroot${TOOL_VERS}/compute --with-mpich ${MPICH_ROOT} --with-papi ${PAPI_ROOT} --with-alps ${ALPS_ROOT}

echo "Step 2 ./install-tool --build-offline --target-shared --target-arch cray --runtime-only --openss-prefix ${BASE_DIR}/ossoff2.2/compute --krell-root-install-prefix ${BASE_DIR}/krellroot${TOOL_VERS}/compute  --with-mpich ${MPICH_ROOT} --with-papi ${PAPI_ROOT}"

./install-tool --build-offline --target-shared --target-arch cray --runtime-only --openss-prefix ${BASE_DIR}/ossoff${TOOL_VERS}/compute --krell-root-install-prefix ${BASE_DIR}/krellroot${TOOL_VERS}/compute  --with-mpich ${MPICH_ROOT} --with-papi ${PAPI_ROOT}

echo "Step 3 ./install-tool --build-cbtf-all --runtime-only --target-arch cray --target-shared --cbtf-prefix ${BASE_DIR}/cbtf_all${TOOL_VERS}/compute --krell-root-prefix ${BASE_DIR}/krellroot${TOOL_VERS}/compute --with-mpich ${MPICH_ROOT} --with-papi ${PAPI_ROOT}"

./install-tool --build-cbtf-all --runtime-only --target-arch cray --target-shared --cbtf-prefix ${BASE_DIR}/cbtf_all${TOOL_VERS}/compute --krell-root-prefix ${BASE_DIR}/krellroot${TOOL_VERS}/compute --with-mpich ${MPICH_ROOT} --with-papi ${PAPI_ROOT}


# ---------------------------------
# For LOGIN NODE builds
# ---------------------------------

module unload PrgEnv-pgi PrgEnv-cray PrgEnv-intel PrgEnv-gnu craype-haswell
module load craype-target-native
module unload gcc
module load gcc/4.9.3
export SYSROOT_DIR=/opt/cray/xc-sysroot/default

echo "Special Build expat Step ./install-tool --build-expat --krell-root-install-prefix ${BASE_DIR}/expat-2.1.0"
./install-tool --build-expat --krell-root-install-prefix ${BASE_DIR}/expat-2.1.0
echo "Special Build python Step ./install-tool --build-expat --krell-root-install-prefix ${BASE_DIR}/expat-2.1.0"
./install-tool --build-python --krell-root-install-prefix ${PYTHON_IROOT}

# Build root components for login nodes
```

```
echo "Step 4 ./install-tool --build-krell-root --krell-root-install-prefix
${BASE_DIR}/krellroot${TOOL_VERS} --with-mpich ${MPICH_ROOT} --with-papi
${PAPI_ROOT} --with-alps ${ALPS_ROOT} --with-expat ${BASE_DIR}/expat-2.1.0"
./install-tool --build-krell-root --krell-root-install-prefix
${BASE_DIR}/krellroot${TOOL_VERS} --with-mpich ${MPICH_ROOT} --with-papi
${PAPI_ROOT} --with-alps ${ALPS_ROOT} --with-expat ${BASE_DIR}/expat-2.1.0

# Build offline for login nodes
echo "Step 5 ./install-tool --build-offline --openss-prefix
${BASE_DIR}/ossoff${TOOL_VERS} --with-runtime-dir
${BASE_DIR}/ossoff${TOOL_VERS}/compute --krell-root-install-prefix
${BASE_DIR}/krellroot${TOOL_VERS} --with-mpich ${MPICH_ROOT} --with-papi
${PAPI_ROOT} --with-python ${PYTHON_IROOT} --with-python-vers ${PYTHON_VERS}"
./install-tool --build-offline --openss-prefix ${BASE_DIR}/ossoff${TOOL_VERS} --with-
runtime-dir ${BASE_DIR}/ossoff${TOOL_VERS}/compute --krell-root-install-prefix
${BASE_DIR}/krellroot${TOOL_VERS} --with-mpich ${MPICH_ROOT} --with-papi
${PAPI_ROOT} --with-python ${PYTHON_IROOT} --with-python-vers ${PYTHON_VERS}

echo "Step 6 ./install-tool --runtime-target-arch cray --build-cbtf-all --cbtf-prefix
${BASE_DIR}/cbtf_all${TOOL_VERS} --krell-root-prefix
${BASE_DIR}/krellroot${TOOL_VERS} --with-mpich ${MPICH_ROOT} --with-cn-boost
${BASE_DIR}/krellroot${TOOL_VERS}/compute --with-cn-mrnet
${BASE_DIR}/krellroot${TOOL_VERS}/compute --with-cn-xercesc
${BASE_DIR}/krellroot${TOOL_VERS}/compute --with-cn-libmonitor
${BASE_DIR}/krellroot${TOOL_VERS}/compute --with-cn-libunwind
${BASE_DIR}/krellroot${TOOL_VERS}/compute --with-cn-dyninst
${BASE_DIR}/krellroot${TOOL_VERS}/compute --with-cn-papi ${PAPI_ROOT} --with-cn-
cbtf-krell ${BASE_DIR}/krellroot${TOOL_VERS}/compute --with-cn-cbtf
${BASE_DIR}/cbtf_all${TOOL_VERS}/compute --with-binutils
${BASE_DIR}/krellroot${TOOL_VERS} --with-boost ${BASE_DIR}/krellroot${TOOL_VERS} -
-with-mrnet ${BASE_DIR}/krellroot${TOOL_VERS} --with-xercesc
${BASE_DIR}/krellroot${TOOL_VERS} --with-libmonitor
${BASE_DIR}/krellroot${TOOL_VERS} --with-libunwind
${BASE_DIR}/krellroot${TOOL_VERS}  --with-dyninst  ${BASE_DIR}/krellroot${TOOL_VERS}
--with-papi ${PAPI_ROOT} --with-alps ${ALPS_ROOT}"

./install-tool --runtime-target-arch cray --build-cbtf-all --cbtf-prefix
${BASE_DIR}/cbtf_all${TOOL_VERS} --krell-root-prefix
${BASE_DIR}/krellroot${TOOL_VERS} --with-mpich ${MPICH_ROOT} --with-cn-boost
${BASE_DIR}/krellroot${TOOL_VERS}/compute --with-cn-mrnet
${BASE_DIR}/krellroot${TOOL_VERS}/compute --with-cn-xercesc
${BASE_DIR}/krellroot${TOOL_VERS}/compute --with-cn-libmonitor
${BASE_DIR}/krellroot${TOOL_VERS}/compute --with-cn-libunwind
${BASE_DIR}/krellroot${TOOL_VERS}/compute --with-cn-dyninst
```

```
${BASE_DIR}/krellroot${TOOL_VERS}/compute --with-cn-papi ${PAPI_ROOT} --with-cn-
cbtf-krell ${BASE_DIR}/krellroot${TOOL_VERS}/compute --with-cn-cbtf
${BASE_DIR}/cbtf_all${TOOL_VERS}/compute --with-binutils
${BASE_DIR}/krellroot${TOOL_VERS} --with-boost ${BASE_DIR}/krellroot${TOOL_VERS} -
-with-mrnet ${BASE_DIR}/krellroot${TOOL_VERS} --with-xercesc
${BASE_DIR}/krellroot${TOOL_VERS} --with-libmonitor
${BASE_DIR}/krellroot${TOOL_VERS} --with-libunwind
${BASE_DIR}/krellroot${TOOL_VERS}  --with-dyninst  ${BASE_DIR}/krellroot${TOOL_VERS}
--with-papi ${PAPI_ROOT} --with-alps ${ALPS_ROOT}


echo "Step 7 ./install-tool --target-arch cray --build-onlyosscbtf --openss-prefix
${BASE_DIR}/osscbtf${TOOL_VERS} --with-cn-cbtf-krell
${BASE_DIR}/cbtf_all${TOOL_VERS}/compute --krell-root-prefix
${BASE_DIR}/krellroot${TOOL_VERS} --with-mpich ${MPICH_ROOT} --with-boost
${BASE_DIR}/krellroot${TOOL_VERS} --with-mrnet ${BASE_DIR}/krellroot${TOOL_VERS}
--with-xercesc ${BASE_DIR}/krellroot${TOOL_VERS} --with-libmonitor
${BASE_DIR}/krellroot${TOOL_VERS} --with-libunwind
${BASE_DIR}/krellroot${TOOL_VERS}  --with-dyninst  ${BASE_DIR}/krellroot${TOOL_VERS}
--with-libelf ${BASE_DIR}/krellroot${TOOL_VERS} --with-libdwarf
${BASE_DIR}/krellroot${TOOL_VERS}  --with-binutils  ${BASE_DIR}/krellroot${TOOL_VERS}
--cbtf-prefix ${BASE_DIR}/cbtf_all${TOOL_VERS} --with-papi ${PAPI_ROOT} --with-
python ${PYTHON_IROOT} --with-python-vers ${PYTHON_VERS}"

./install-tool --target-arch cray --build-onlyosscbtf --openss-prefix
${BASE_DIR}/osscbtf${TOOL_VERS} --with-cn-cbtf-krell
${BASE_DIR}/cbtf_all${TOOL_VERS}/compute --krell-root-prefix
${BASE_DIR}/krellroot${TOOL_VERS} --with-mpich ${MPICH_ROOT} --with-boost
${BASE_DIR}/krellroot${TOOL_VERS} --with-mrnet ${BASE_DIR}/krellroot${TOOL_VERS}
--with-xercesc ${BASE_DIR}/krellroot${TOOL_VERS} --with-libmonitor
${BASE_DIR}/krellroot${TOOL_VERS} --with-libunwind
${BASE_DIR}/krellroot${TOOL_VERS}  --with-dyninst  ${BASE_DIR}/krellroot${TOOL_VERS}
--with-libelf ${BASE_DIR}/krellroot${TOOL_VERS} --with-libdwarf
${BASE_DIR}/krellroot${TOOL_VERS}  --with-binutils  ${BASE_DIR}/krellroot${TOOL_VERS}
--cbtf-prefix ${BASE_DIR}/cbtf_all${TOOL_VERS} --with-papi ${PAPI_ROOT} --with-
python ${PYTHON_IROOT} --with-python-vers ${PYTHON_VERS}

# Compiling with a module loaded non-default installed compiler requires that the
libstc++ library be made available on the compute nodes
echo "Special cp libstdc++  Step cp /opt/gcc/4.9.3/snos/lib64/libstdc++.so.6
${BASE_DIR}/krellroot${TOOL_VERS}/compute/lib64/libstdc++.so.6"
cp /opt/gcc/4.9.3/snos/lib64/libstdc++.so.6
${BASE_DIR}/krellroot${TOOL_VERS}/compute/lib64/libstdc++.so.6
echo "Special cp libstdc++  Step cp /opt/gcc/4.9.3/snos/lib64/libstdc++.so.6
```

```
${BASE_DIR}/krellroot${TOOL_VERS}/lib64/libstdc++.so.6"
cp /opt/gcc/4.9.3/snos/lib64/libstdc++.so.6
${BASE_DIR}/krellroot${TOOL_VERS}/lib64/libstdc++.so.6
```

Garnet   garnet.erdc.hpc.mil   ERDC

```
#!/bin/bash

# ----------------------------
# First Build the compute node version (root and oss versions if you use the
recommended krell externals root build):
# For COMPUTE NODE builds:
# ---------------------------------

. /opt/modules/default/init/bash

module swap PrgEnv-pgi PrgEnv-gnu
export cc=gcc
export CC=gcc
export CXX=g++

# Template values for the install-tool references below
BASE_DIR=/usr/local/usp/PETtools/CE/pkgs/openss
#BASE_DIR=/u/galarowi/OSS
TOOL_VERS="_v2.2.3"
KROOT_DIR=${BASE_DIR}/krellroot${TOOL_VERS}
OSSOFF_DIR=${BASE_DIR}/ossoff${TOOL_VERS}
OSSCBTF_DIR=${BASE_DIR}/osscbtf${TOOL_VERS}
CBTFALL_DIR=${BASE_DIR}/cbtf${TOOL_VERS}
MPICH2_ROOT=/opt/cray/mpt/7.3.0/gni/mpich-gnu/5.1
PAPI_ROOT=/opt/cray/papi/default
ALPS_ROOT=/opt/cray/alps/default


# Build root components runtime only
echo "./install-tool --runtime-only --target-arch cray --target-shared --build-krell-root --
krell-root--prefix ${KROOT_DIR}/compute --with-mpich2 ${MPICH2_ROOT} --with-papi
${PAPI_ROOT} --with-alps ${ALPS_ROOT}"

./install-tool --runtime-only --target-arch cray --target-shared --build-krell-root --krell-
root-install-prefix ${KROOT_DIR}/compute --with-mpich2 ${MPICH2_ROOT} --with-papi
${PAPI_ROOT} --with-alps ${ALPS_ROOT} --skip-binutils-build --skip-libdwarf-build --skip-
```

```
libunwind-build --skip-papi-build --skip-vampirtrace-build --skip-sqlite-build --skip-
libmonitor-build --skip-boost-build

# Build offline runtime only
echo "./install-tool --build-offline --target-shared --target-arch cray --runtime-only --
openss-prefix ${OSSOFF_DIR}/compute --krell-root-install-prefix
${KROOT_DIR}/compute --with-mpich2 ${MPICH2_ROOT} --with-papi ${PAPI_ROOT}"

./install-tool --build-offline --target-shared --target-arch cray --runtime-only --openss-
prefix ${OSSOFF_DIR}/compute --krell-root-install-prefix ${KROOT_DIR}/compute --with-
mpich2 ${MPICH2_ROOT} --with-papi ${PAPI_ROOT}

# Build cbtf runtime only
echo "./install-tool --build-cbtf-all --target-shared --target-arch cray --runtime-only --
cbtf-prefix ${CBTFALL_DIR}/compute --krell-root-install-prefix ${KROOT_DIR}/compute --
with-mpich2 ${MPICH2_ROOT} --with-papi ${PAPI_ROOT}"

./install-tool --build-cbtf-all --target-shared --target-arch cray --runtime-only --cbtf-prefix
${CBTFALL_DIR}/compute --krell-root-install-prefix ${KROOT_DIR}/compute --with-
mpich2 ${MPICH2_ROOT} --with-papi ${PAPI_ROOT}

# ----------------------------
# Next build the login node version (root and oss versions if you use the recommended
krell externals root build):
# ----------------------------
# ---------------------------------
# For LOGIN NODE builds
# ---------------------------------

. /opt/modules/default/init/bash
module unload PrgEnv-pgi craype-interlagos PrgEnv-gnu craype-interlagos
module load craype-target-native
module load gcc
export SYSROOT_DIR=/opt/cray/xe-sysroot/default

# Build root components for login nodes
echo "./install-tool --build-krell-root --krell-root-install-prefix ${KROOT_DIR} --with-
mpich2 ${MPICH2_ROOT} --with-papi ${PAPI_ROOT} --with-alps ${ALPS_ROOT}"

./install-tool --build-krell-root --krell-root-install-prefix ${KROOT_DIR} --with-mpich2
${MPICH2_ROOT} --with-papi ${PAPI_ROOT} --with-alps ${ALPS_ROOT}

# Build offline for login nodes
echo "./install-tool --build-offline --openss-prefix ${OSSOFF_DIR} --with-runtime-dir
```

```
${OSSOFF_DIR}/compute --krell-root-install-prefix ${KROOT_DIR} --with-mpich2
${MPICH2_ROOT} --with-papi ${PAPI_ROOT}"

./install-tool --build-offline --openss-prefix ${OSSOFF_DIR} --with-runtime-dir
${OSSOFF_DIR}/compute --krell-root-install-prefix ${KROOT_DIR} --with-mpich2
${MPICH2_ROOT} --with-papi ${PAPI_ROOT}

# Build cbtf and openss for login nodes
echo "./install-tool --runtime-target-arch cray --build-cbtf-all --cbtf-prefix ${CBTFALL_DIR}
--krell-root-prefix ${KROOT_DIR} --with-mpich2 ${MPICH2_ROOT} --with-cn-boost
${KROOT_DIR}/compute --with-cn-mrnet ${KROOT_DIR}/compute --with-cn-xercesc
${KROOT_DIR}/compute --with-cn-libmonitor ${KROOT_DIR}/compute --with-cn-
libunwind ${KROOT_DIR}/compute --with-cn-dyninst ${KROOT_DIR}/compute --with-cn-
papi ${PAPI_ROOT} --with-cn-cbtf-krell ${CBTFALL_DIR}/compute --with-cn-cbtf
${CBTFALL_DIR}/compute --with-binutils ${KROOT_DIR} --with-boost ${KROOT_DIR} --
with-mrnet ${KROOT_DIR} --with-xercesc ${KROOT_DIR} --with-libmonitor ${KROOT_DIR}
--with-libunwind ${KROOT_DIR} --with-dyninst ${KROOT_DIR} --with-papi ${PAPI_ROOT}
--with-alps ${ALPS_ROOT}"

./install-tool --runtime-target-arch cray --build-cbtf-all --cbtf-prefix ${CBTFALL_DIR} --
krell-root-prefix ${KROOT_DIR} --with-mpich2 ${MPICH2_ROOT} --with-cn-boost
${KROOT_DIR}/compute --with-cn-mrnet ${KROOT_DIR}/compute --with-cn-xercesc
${KROOT_DIR}/compute --with-cn-libmonitor ${KROOT_DIR}/compute --with-cn-
libunwind ${KROOT_DIR}/compute --with-cn-dyninst ${KROOT_DIR}/compute --with-cn-
papi ${PAPI_ROOT} --with-cn-cbtf-krell ${CBTFALL_DIR}/compute --with-cn-cbtf
${CBTFALL_DIR}/compute --with-binutils ${KROOT_DIR} --with-boost ${KROOT_DIR} --
with-mrnet ${KROOT_DIR} --with-xercesc ${KROOT_DIR} --with-libmonitor ${KROOT_DIR}
--with-libunwind ${KROOT_DIR} --with-dyninst ${KROOT_DIR} --with-papi ${PAPI_ROOT}
--with-alps ${ALPS_ROOT}

echo "./install-tool --target-arch cray --build-onlyosscbtf --openss-prefix ${OSSCBTF_DIR}
--with-cn-cbtf-krell ${CBTFALL_DIR}/compute --krell-root-prefix ${KROOT_DIR} --with-
mpich2 ${MPICH2_ROOT} --with-papi ${PAPI_ROOT} --with-boost ${KROOT_DIR} --with-
mrnet ${KROOT_DIR} --with-xercesc ${KROOT_DIR} --with-libmonitor ${KROOT_DIR} --
with-libunwind ${KROOT_DIR} --with-dyninst ${KROOT_DIR} --with-libelf ${KROOT_DIR}
--with-libdwarf ${KROOT_DIR} --with-binutils ${KROOT_DIR} --cbtf-prefix
${CBTFALL_DIR}"

./install-tool --target-arch cray --build-onlyosscbtf --openss-prefix ${OSSCBTF_DIR} --
with-cn-cbtf-krell ${CBTFALL_DIR}/compute --krell-root-prefix ${KROOT_DIR} --with-
mpich2 ${MPICH2_ROOT} --with-papi ${PAPI_ROOT} --with-boost ${KROOT_DIR} --with-
mrnet ${KROOT_DIR} --with-xercesc ${KROOT_DIR} --with-libmonitor ${KROOT_DIR} --
with-libunwind ${KROOT_DIR} --with-dyninst ${KROOT_DIR} --with-libelf ${KROOT_DIR}
--with-libdwarf ${KROOT_DIR} --with-binutils ${KROOT_DIR} --cbtf-prefix ${CBTFALL_DIR}
```

```
# Compiling with a module loaded non-default installed compiler requires that the
libstc++ library be made available on the compute nodes
cp /opt/gcc/4.9.2/snos/lib64/libstdc++.so.6
${KROOT_DIR}/compute/lib64/libstdc++.so.6
chmod 755 ${KROOT_DIR}/compute/lib64/libstdc++.so.6
cp /opt/gcc/4.9.2/snos/lib64/libstdc++.so.6 ${KROOT_DIR}/lib64/libstdc++.so.6
chmod 755 ${KROOT_DIR}/lib64/libstdc++.so.6
```

Gordon **gordon.navydsrc.hpc.mil** **NAVY**

```
#!/bin/bash
. /opt/modules/default/init/bash
module unload PrgEnv-pgi PrgEnv-cray PrgEnv-intel
module load PrgEnv-gnu
module unload gcc; module load gcc/4.9.2
export SYSROOT_DIR=/opt/cray/xc-sysroot/default

#BASE_IDIR=/p/home/galarowi/OSS
BASE_IDIR=/p/home/app/PET/pkgs/openss
TOOL_VERS="_v2.2.3"
KROOT_IDIR=${BASE_IDIR}/krellroot${TOOL_VERS}
CBTF_IDIR=${BASE_IDIR}/cbtf${TOOL_VERS}
OSSCBTF_IDIR=${BASE_IDIR}/osscbtf${TOOL_VERS}
OSSOFF_IDIR=${BASE_IDIR}/ossoff${TOOL_VERS}
PAPI_IDIR=/opt/cray/papi/default
MPICH_IDIR=/opt/cray/mpt/7.3.0/gni/mpich-gnu/5.1
ALPS_IDIR=/opt/cray/alps/default

module load /p/home/galarowi/privatemodules/cmake-3.2.2

export cc=gcc
export CC=gcc
export CXX=g++

# Build root components runtime only
echo  Build root components runtime only

echo Step 1 ./install-tool --runtime-only --target-arch cray --target-shared --build-krell-
root --krell-root-prefix ${KROOT_IDIR}/compute --with-mpich ${MPICH_IDIR} --with-papi
${PAPI_IDIR} --with-alps ${ALPS_IDIR}
./install-tool --runtime-only --target-arch cray --target-shared --build-krell-root --krell-
root-prefix ${KROOT_IDIR}/compute --with-mpich ${MPICH_IDIR} --with-papi
${PAPI_IDIR} --with-alps ${ALPS_IDIR}
```

```
echo Step 2 ./install-tool --build-offline --target-shared --target-arch cray --runtime-only
--openss-prefix ${OSSOFF_IDIR}/compute --krell-root-prefix  ${KROOT_IDIR}/compute  --
with-mpich ${MPICH_IDIR} --with-papi ${PAPI_IDIR}
./install-tool --build-offline --target-shared --target-arch cray --runtime-only --openss-
prefix ${OSSOFF_IDIR}/compute --krell-root-prefix  ${KROOT_IDIR}/compute  --with-
mpich ${MPICH_IDIR} --with-papi ${PAPI_IDIR}

./install-tool --build-cbtf-all --runtime-only --target-arch cray --target-shared --cbtf-prefix
${CBTF_IDIR}/compute --krell-root-prefix  ${KROOT_IDIR}/compute --with-mpich
${MPICH_IDIR} --with-papi ${PAPI_IDIR}


# ---------------------------------
# For LOGIN NODE builds
# ---------------------------------

module unload PrgEnv-pgi PrgEnv-cray PrgEnv-intel PrgEnv-gnu craype-haswell
module load craype-target-native
module unload gcc
module load gcc/4.9.2
module load /p/home/galarowi/privatemodules/cmake-3.2.2

export SYSROOT_DIR=/opt/cray/xc-sysroot/default

# Build root components for login nodes
echo Step 3 ./install-tool --build-krell-root --krell-root-prefix ${KROOT_IDIR} --with-
mpich ${MPICH_IDIR} --with-papi ${PAPI_IDIR} --with-alps ${ALPS_IDIR}
./install-tool --build-krell-root --krell-root-prefix ${KROOT_IDIR} --with-mpich
${MPICH_IDIR} --with-papi ${PAPI_IDIR} --with-alps ${ALPS_IDIR}

# Build offline for login nodes
echo Step 4 ./install-tool --build-offline --openss-prefix ${OSSOFF_IDIR} --with-runtime-
dir ${OSSOFF_IDIR}/compute --krell-root-prefix ${KROOT_IDIR} --with-mpich
${MPICH_IDIR} --with-papi ${PAPI_IDIR}
./install-tool --build-offline --openss-prefix ${OSSOFF_IDIR} --with-runtime-dir
${OSSOFF_IDIR}/compute --krell-root-prefix ${KROOT_IDIR} --with-mpich ${MPICH_IDIR}
--with-papi ${PAPI_IDIR}

./install-tool --runtime-target-arch cray --build-cbtf-all --cbtf-prefix ${CBTF_IDIR} --krell-
root-prefix ${KROOT_IDIR} --with-mpich ${MPICH_IDIR} --with-cn-boost
${KROOT_IDIR}/compute --with-cn-mrnet ${KROOT_IDIR}/compute --with-cn-xercesc
${KROOT_IDIR}/compute --with-cn-libmonitor ${KROOT_IDIR}/compute --with-cn-
libunwind ${KROOT_IDIR}/compute --with-cn-dyninst ${KROOT_IDIR}/compute --with-
```

```
cn-papi ${PAPI_IDIR} --with-cn-cbtf-krell ${KROOT_IDIR}/compute --with-cn-cbtf
${CBTF_IDIR}/compute --with-binutils ${KROOT_IDIR} --with-boost ${KROOT_IDIR} --
with-mrnet ${KROOT_IDIR} --with-xercesc ${KROOT_IDIR} --with-libmonitor
${KROOT_IDIR} --with-libunwind ${KROOT_IDIR} --with-dyninst ${KROOT_IDIR} --with-
papi ${PAPI_IDIR} --with-alps ${ALPS_IDIR}

./install-tool --target-arch cray --build-onlyosscbtf --openss-prefix ${OSSCBTF_IDIR} --
with-cn-cbtf-krell ${CBTF_IDIR}/compute --krell-root-prefix ${KROOT_IDIR} --with-mpich
${MPICH_IDIR} --with-boost ${KROOT_IDIR} --with-mrnet ${KROOT_IDIR} --with-xercesc
${KROOT_IDIR} --with-libmonitor ${KROOT_IDIR} --with-libunwind ${KROOT_IDIR} --
with-dyninst ${KROOT_IDIR} --with-libelf ${KROOT_IDIR} --with-libdwarf ${KROOT_IDIR}
--with-binutils ${KROOT_IDIR} --cbtf-prefix ${CBTF_IDIR} --with-papi ${PAPI_IDIR}

# Compiling with a module loaded non-default installed compiler requires that the
libstc++ library be made available on the compute nodes
cp /opt/gcc/4.9.2/snos/lib64/libstdc++.so.6 ${KROOT_IDIR}/lib64/libstdc++.so.6
chmod 755 ${KROOT_IDIR}/lib64/libstdc++.so.6
cp /opt/gcc/4.9.2/snos/lib64/libstdc++.so.6
${KROOT_IDIR}/compute/lib64/libstdc++.so.6
chmod 755 ${KROOT_IDIR}/compute/lib64/libstdc++.so.6
```

Haise **haise.navo.hpc.mil** **NAVY**

```
#!/bin/bash


. /site/modules/modules-3.2.9c/Modules/3.2.9/init/bash


module purge

module load gcc/4.8.2


module load /u/home/galarowi/privatemodules/cmake-3.2.2

module unload compiler/intel mpi/intel/impi/4.1.3


export BASE_IDIR=/site/PET/pkgs/openss

export TOOL_VERS="_v2.2.3"

export MPICH_IDIR=/site/intel/impi/4.1.3.048/intel64
export OPENMPI_IDIR=/site/openmpi/gnu/1.8.4
export PYTHON_IDIR=${BASE_IDIR}/python-2.7.3
export PYTHON_VERS="2.7"
export LTDL_IDIR=${BASE_IDIR}/autotools${TOOL_VERS}
export KROOT_IDIR=${BASE_IDIR}/krellroot${TOOL_VERS}
export CBTF_IDIR=${BASE_IDIR}/cbtf${TOOL_VERS}
export OSSCBTF_IDIR=${BASE_IDIR}/osscbtf${TOOL_VERS}
export OSSOFF_IDIR=${BASE_IDIR}/ossoff${TOOL_VERS}

export cc=gcc
export CC=gcc
export CXX=g++

./install-tool --build-autotools --krell-root-prefix ${LTDL_IDIR}
./install-tool --build-python --krell-root-prefix ${PYTHON_IDIR}

# Build root components
./install-tool --build-krell-root --krell-root-prefix ${KROOT_IDIR} --with-mpich
${MPICH_IDIR} --with-ltdl ${LTDL_IDIR} --force-boost-build --with-openmpi
${OPENMPI_IDIR} --with-python ${PYTHON_IDIR} --with-python-vers ${PYTHON_VERS}

./install-tool --build-offline --openss-prefix ${OSSOFF_IDIR} --krell-root-prefix
${KROOT_IDIR} --with-mpich ${MPICH_IDIR} --with-ltdl ${LTDL_IDIR} --with-openmpi
${OPENMPI_IDIR} --with-python ${PYTHON_IDIR} --with-python-vers ${PYTHON_VERS}

./install-tool --build-cbtf-all --cbtf-prefix ${CBTF_IDIR} --krell-root-prefix  ${KROOT_IDIR}
--with-mpich ${MPICH_IDIR} --with-ltdl ${LTDL_IDIR} --with-openmpi ${OPENMPI_IDIR} -
-with-python ${PYTHON_IDIR} --with-python-vers ${PYTHON_VERS}

./install-tool --build-onlyosscbtf --openss-prefix ${OSSCBTF_IDIR} --krell-root-prefix
${KROOT_IDIR} --cbtf-prefix ${CBTF_IDIR} --with-mpich ${MPICH_IDIR} --with-ltdl
${LTDL_IDIR} --with-openmpi ${OPENMPI_IDIR} --with-python ${PYTHON_IDIR} --with-
```

```
python-vers ${PYTHON_VERS}


# Compiling with a module loaded non-default installed compiler requires that the
libstc++ library be made available on the compute nodes

cp /site/unsupported/gcc/gcc-4.8.2/lib64/libstdc++.so.6
${KROOT_IDIR}/lib64/libstdc++.so.6
```

Kilrain kilrain.navo.hpc.mil NAVY

```
#!/bin/bash

. /site/modules/modules-3.2.9c/Modules/3.2.9/init/bash
module purge
module load gcc/4.8.2

export cc=gcc
export CC=gcc
export CXX=g++

export BASE_IDIR=/site/PET/pkgs/openss
export TOOL_VERS="_v2.2.3"
export OPENMPI_ROOT=/site/openmpi/gnu/1.6.5
export MPICH_ROOT=/site/intel/impi/4.1.3.048/intel64


./install-tool --build-cmake --krell-root-install-prefix ${BASE_IDIR}/cmake-3.2.2
export PATH=${BASE_IDIR}/cmake-3.2.2/bin:$PATH

./install-tool --build-autotools --krell-root-install-prefix
${BASE_IDIR}/autotools_root${TOOL_VERS}

./install-tool --build-python --krell-root-install-prefix
${BASE_IDIR}/python_root${TOOL_VERS}

# Build root components
./install-tool --build-krell-root --krell-root-install-prefix
${BASE_IDIR}/krellroot${TOOL_VERS} --with-mpich ${MPICH_ROOT} --with-ltdl
${BASE_IDIR}/autotools_root${TOOL_VERS} --force-boost-build --with-openmpi
${OPENMPI_ROOT} --with-python ${BASE_IDIR}/python_root${TOOL_VERS}


./install-tool --build-offline --openss-prefix ${BASE_IDIR}/ossoffline${TOOL_VERS} --krell-
```

```
root-install-prefix  ${BASE_IDIR}/krellroot${TOOL_VERS} --with-mpich ${MPICH_ROOT} -
-with-ltdl ${BASE_IDIR}/autotools_root${TOOL_VERS} --with-openmpi
${OPENMPI_ROOT} --with-python ${BASE_IDIR}/python_root${TOOL_VERS}

./install-tool --build-cbtf-all --cbtf-prefix ${BASE_IDIR}/cbtf${TOOL_VERS} --krell-root-
prefix  ${BASE_IDIR}/krellroot${TOOL_VERS} --with-mpich ${MPICH_ROOT} --with-ltdl
${BASE_IDIR}/autotools_root${TOOL_VERS} --with-openmpi ${OPENMPI_ROOT} --with-
python ${BASE_IDIR}/python_root${TOOL_VERS}

./install-tool --build-onlyosscbtf --openss-prefix  ${BASE_IDIR}/osscbtf${TOOL_VERS} --
krell-root-prefix ${BASE_IDIR}/krellroot${TOOL_VERS} --cbtf-prefix
${BASE_IDIR}/cbtf${TOOL_VERS}  --with-mpich ${MPICH_ROOT} --with-ltdl
${BASE_IDIR}/autotools_root${TOOL_VERS} --with-openmpi ${OPENMPI_ROOT} --with-
python ${BASE_IDIR}/python_root${TOOL_VERS}

# Compiling with a module loaded non-default installed compiler requires that the
libstc++ library be made available on the compute nodes
cp /site/unsupported/gcc/gcc-4.8.2/lib64/libstdc++.so.6
${BASE_IDIR}/krellroot${TOOL_VERS}/lib64/libstdc++.so.6
chmod 755 ${BASE_IDIR}/krellroot${TOOL_VERS}/lib64/libstdc++.so.6
```

Lightning   lightning.afrl.hpc.mil   AFRL

```
#!/bin/bash

# ----------------------------
# First Build the compute node version (root and oss versions if you use the
recommended krell externals root build):
# For COMPUTE NODE builds:
# ---------------------------------

. /opt/modules/default/init/bash

module unload PrgEnv-cray PrgEnv-pgi PrgEnv-gnu PrgEnv-intel
module load PrgEnv-gnu gcc/5.1.0
module load cmake

export cc=gcc
export CC=gcc
export CXX=g++
export BASE_IDIR=/app/comenv/pkgs/openss
export TOOL_VERS="_v2.2.3"
#export CMAKE_IDIR=${CMAKE_IDIR}/cmake-3.2.2
```

```
export PATH=${CMAKE_IDIR}/bin:$PATH
export MPICH_IDIR=/opt/cray/mpt/7.2.6/gni/mpich-gnu/5.1
export PAPI_IDIR=/opt/cray/papi/5.4.1.2
export BINUTILS_IDIR=/opt/cray/xc-sysroot/default/usr
export KROOT_IDIR=${BASE_IDIR}/krellroot${TOOL_VERS}
export CBTF_IDIR=${BASE_IDIR}/cbtf${TOOL_VERS}
export OSSCBTF_IDIR=${BASE_IDIR}/osscbtf${TOOL_VERS}
export OSSOFF_IDIR=${BASE_IDIR}/ossoff${TOOL_VERS}
export PYTHON_IDIR=${BASE_IDIR}/python-2.7.3
export PYTHON_VERS="2.7"
export ALPS_IDIR=/opt/cray/alps/5.2.4-2.0502.9774.31.11.ari

./install-tool --runtime-only --target-arch cray --target-shared --build-krell-root --krell-
root-prefix ${KROOT_IDIR}/compute --with-mpich ${MPICH_IDIR} --with-papi
${PAPI_IDIR} --with-binutils ${BINUTILS_IDIR} --with-alps ${ALPS_IDIR}

./install-tool --runtime-only --target-arch cray --target-shared --build-offline --openss-
prefix ${OSSOFF_IDIR}/compute --krell-root-prefix ${KROOT_IDIR}/compute --with-
mpich ${MPICH_IDIR} --with-papi ${PAPI_IDIR} --with-binutils ${BINUTILS_IDIR} --with-
alps ${ALPS_IDIR}

./install-tool --build-cbtf-all --runtime-only --target-arch cray --target-shared --cbtf-prefix
${CBTF_IDIR}/compute --krell-root-prefix  ${KROOT_IDIR}/compute --with-mpich
${MPICH_IDIR} --with-papi ${PAPI_IDIR} --with-alps ${ALPS_IDIR}

# ----------------------------
# Next build the login node version (root and oss versions if you use the recommended
krell externals root build):
# For LOGIN NODE builds:
# ----------------------------------

module unload PrgEnv-cray PrgEnv-gnu craype-ivybridge
module load craype-target-native
module load gcc/5.1.0
module load cmake

export cc=gcc
export CC=gcc
export CXX=g++

# BUILD PYTHON because I didn't see one available
./install-tool --build-python --krell-root-prefix ${PYTHON_IDIR}
export LD_LIBRARY_PATH=${PYTHON_IDIR}/lib:$LD_LIBRARY_PATH
```

```
./install-tool --build-krell-root --krell-root-prefix ${KROOT_IDIR} --with-mpich
${MPICH_IDIR} --with-python ${PYTHON_IDIR} --with-python-vers ${PYTHON_VERS} --
with-papi ${PAPI_IDIR} --force-binutils-build --with-alps ${ALPS_IDIR}

./install-tool --runtime-target-arch cray --build-cbtf-all --cbtf-prefix ${CBTF_IDIR} --krell-
root-prefix ${KROOT_IDIR} --with-mpich ${MPICH_IDIR} --with-cn-boost
${KROOT_IDIR}/compute --with-cn-mrnet ${KROOT_IDIR}/compute --with-cn-xercesc
${KROOT_IDIR}/compute --with-cn-libmonitor ${KROOT_IDIR}/compute --with-cn-
libunwind ${KROOT_IDIR}/compute --with-cn-dyninst ${KROOT_IDIR}/compute --with-
cn-papi ${PAPI_IDIR} --with-cn-cbtf-krell ${KROOT_IDIR}/compute --with-cn-cbtf
${CBTF_IDIR}/compute --with-binutils ${KROOT_IDIR} --with-boost ${KROOT_IDIR} --
with-mrnet ${KROOT_IDIR} --with-xercesc ${KROOT_IDIR} --with-libmonitor
${KROOT_IDIR} --with-libunwind ${KROOT_IDIR} --with-dyninst ${KROOT_IDIR} --with-
papi ${PAPI_IDIR} --with-alps ${ALPS_IDIR}

./install-tool --target-arch cray --build-onlyosscbtf --openss-prefix ${OSSCBTF_IDIR} --
with-cn-cbtf-krell ${CBTF_IDIR}/compute --krell-root-prefix ${KROOT_IDIR} --with-mpich
${MPICH_IDIR} --with-boost ${KROOT_IDIR} --with-mrnet ${KROOT_IDIR} --with-xercesc
${KROOT_IDIR} --with-libmonitor ${KROOT_IDIR} --with-libunwind ${KROOT_IDIR} --
with-dyninst ${KROOT_IDIR} --with-libelf ${KROOT_IDIR} --with-libdwarf ${KROOT_IDIR}
--with-binutils ${KROOT_IDIR} --cbtf-prefix ${CBTF_IDIR} --with-papi ${PAPI_IDIR}

./install-tool --build-offline --openss-prefix ${OSSOFF_IDIR} --with-runtime-dir
${OSSOFF_IDIR}/compute --krell-root-prefix ${KROOT_IDIR} --with-mpich ${MPICH_IDIR}
--with-python ${PYTHON_IDIR} --with-python-vers ${PYTHON_VERS} --with-papi
${PAPI_IDIR}

# Compiling with a module loaded non-default installed compiler requires that the
libstc++ library be made available on the compute nodes

cp /opt/gcc/5.1.0/snos/lib64/libstdc++.so.6 ${KROOT_IDIR}/lib64/libstdc++.so.6
chmod 755 ${KROOT_IDIR}/lib64/libstdc++.so.6

cp /opt/gcc/5.1.0/snos/lib64/libstdc++.so.6
${KROOT_IDIR}/compute/lib64/libstdc++.so.6
chmod 755 ${KROOT_IDIR}/compute/lib64/libstdc++.so.6
```

Riptide riptide.mhpcc.hpc.mil MHPCC

```
#!/bin/bash

. /usr/share/Modules/init/bash

module unload intel/13.0 impi/4.1
module load gcc/gnu/4.9.3

export BASE_IDIR=/gpfs/pkgs/hpcmp/PTOOLS/pkgs/openss
export TOOL_VERS="_v2.2.3"
export MPICH_IDIR=/gpfs/pkgs/mhpcc/intel/2013/impi/4.1.0.024/intel64
export OPENMPI_IDIR=/gpfs/pkgs/mhpcc/openmpi/tm/gnu/1.8.5
export PYTHON_IDIR=${BASE_IDIR}/python-2.7.3
export PYTHON_VERS="2.7"
export LTDL_IDIR=${BASE_IDIR}/autotools${TOOL_VERS}
export KROOT_IDIR=${BASE_IDIR}/krellroot${TOOL_VERS}
export CBTF_IDIR=${BASE_IDIR}/cbtf${TOOL_VERS}
export OSSCBTF_IDIR=${BASE_IDIR}/osscbtf${TOOL_VERS}
export OSSOFF_IDIR=${BASE_IDIR}/ossoff${TOOL_VERS}

export cc=gcc
export CC=gcc
export CXX=g++

./install-tool --build-autotools --krell-root-prefix ${LTDL_IDIR}
./install-tool --build-python --krell-root-prefix ${PYTHON_IDIR}

# Build root components
./install-tool --build-krell-root --krell-root-prefix ${KROOT_IDIR} --with-mpich
${MPICH_IDIR} --with-ltdl ${LTDL_IDIR} --force-boost-build --with-openmpi
${OPENMPI_IDIR} --with-python ${PYTHON_IDIR} --with-python-vers ${PYTHON_VERS}

./install-tool --build-offline --openss-prefix ${OSSOFF_IDIR} --krell-root-prefix
${KROOT_IDIR} --with-mpich ${MPICH_IDIR} --with-ltdl ${LTDL_IDIR} --with-openmpi
${OPENMPI_IDIR} --with-python ${PYTHON_IDIR} --with-python-vers ${PYTHON_VERS}

./install-tool --build-cbtf-all --cbtf-prefix ${CBTF_IDIR} --krell-root-prefix  ${KROOT_IDIR}
--with-mpich ${MPICH_IDIR} --with-ltdl ${LTDL_IDIR} --with-openmpi ${OPENMPI_IDIR} -
-with-python ${PYTHON_IDIR} --with-python-vers ${PYTHON_VERS}

./install-tool --build-onlyosscbtf --openss-prefix ${OSSCBTF_IDIR} --krell-root-prefix
${KROOT_IDIR} --cbtf-prefix ${CBTF_IDIR} --with-mpich ${MPICH_IDIR} --with-ltdl
${LTDL_IDIR} --with-openmpi ${OPENMPI_IDIR} --with-python ${PYTHON_IDIR} --with-
python-vers ${PYTHON_VERS}

# Compiling with a module loaded non-default installed compiler requires that the
libstc++ library be made available on the compute nodes
cp /gpfs/pkgs/mhpcc/gcc-4.9.3//lib64/libstdc++.so.6
${KROOT_IDIR}/lib64/libstdc++.so.6
```

Shepard shepard.navydsrc.hpc.mil NAVY

```
#!/bin/bash

# ----------------------------
# First Build the compute node version (root and oss versions if you use the
recommended krell externals root build):
# For COMPUTE NODE builds:
# ---------------------------------

. /opt/modules/default/init/bash

module unload PrgEnv-cray PrgEnv-gnu craype-ivybridge
module load craype-target-native
module load gcc/4.8.2

export BASE_IDIR=/p/home/app/PET/pkgs/openss
export CMAKE_IDIR=${BASE_IDIR}/cmake-3.2.2

./install-tool --build-cmake --krell-root-prefix ${CMAKE_IDIR}
export PATH=${CMAKE_IDIR}/bin:$PATH

module unload PrgEnv-cray PrgEnv-pgi PrgEnv-gnu PrgEnv-intel
module load PrgEnv-gnu gcc/4.8.2
module unload craype-target-native
module load craype-ivybridge

export CC=gcc
export CXX=g++
export TOOL_VERS="_v2.2.3"
export CMAKE_IDIR=${BASE_IDIR}/cmake-3.2.2
export PATH=${CMAKE_IDIR}/bin:$PATH
export MPICH_IDIR=/opt/cray/mpt/6.3.1/gni/mpich2-gnu/48
export PAPI_IDIR=/opt/cray/papi/5.3.2
export BINUTILS_IDIR=/opt/cray/xc-sysroot/default/usr
export KROOT_IDIR=${BASE_IDIR}/krellroot${TOOL_VERS}
export CBTF_IDIR=${BASE_IDIR}/cbtf${TOOL_VERS}
export OSSCBTF_IDIR=${BASE_IDIR}/osscbtf${TOOL_VERS}
export OSSOFF_IDIR=${BASE_IDIR}/ossoff${TOOL_VERS}
export PYTHON_IDIR=${BASE_IDIR}/python-2.7.3
export ALPS_IDIR=/opt/cray/alps/5.2.1-2.0502.9041.11.6.ari

# Build compute node portions of the tool
```

```
./install-tool --build-krell-root --runtime-only --target-arch cray --target-shared --krell-
root-prefix ${KROOT_IDIR}/compute --with-mpich ${MPICH_IDIR} --with-papi
${PAPI_IDIR} --with-binutils ${BINUTILS_IDIR} --with-alps ${ALPS_IDIR}

./install-tool --build-offline --runtime-only --target-arch cray --target-shared --openss-
prefix ${OSSOFF_IDIR}/compute --krell-root-prefix ${KROOT_IDIR}/compute --with-
mpich ${MPICH_IDIR} --with-papi ${PAPI_IDIR} --with-binutils ${BINUTILS_IDIR} --with-
alps ${ALPS_IDIR}

./install-tool --build-cbtf-all --runtime-only --target-arch cray --target-shared --cbtf-prefix
${CBTF_IDIR}/compute --krell-root-prefix  ${KROOT_IDIR}/compute --with-mpich
${MPICH_IDIR} --with-papi ${PAPI_IDIR} --with-alps ${ALPS_IDIR}


# ----------------------------
# Next build the login node version (root and oss versions if you use the recommended
krell externals root build):
# For LOGIN NODE builds:
# ---------------------------------

module unload PrgEnv-cray PrgEnv-gnu craype-ivybridge
module load craype-target-native
module load gcc/4.8.2
export LD_LIBRARY_PATH=${PYTHON_IDIR}/lib:$LD_LIBRARY_PATH
export PATH=${CMAKE_IDIR}/bin:$PATH

export cc=gcc
export CC=gcc
export CXX=g++

# Build python because I didn't see a version of python with python-devel available
./install-tool --build-python --krell-root-prefix ${PYTHON_IDIR}
export LD_LIBRARY_PATH=${PYTHON_IDIR}/lib:$LD_LIBRARY_PATH

./install-tool --build-krell-root --krell-root-prefix ${KROOT_IDIR} --with-mpich
${MPICH_IDIR} --with-python ${PYTHON_IDIR} --with-papi ${PAPI_IDIR} --force-binutils-
build --with-alps ${ALPS_IDIR}

./install-tool --runtime-target-arch cray --build-cbtf-all --cbtf-prefix ${CBTF_IDIR} --krell-
root-prefix ${KROOT_IDIR} --with-mpich ${MPICH_IDIR} --with-cn-boost
${KROOT_IDIR}/compute --with-cn-mrnet ${KROOT_IDIR}/compute --with-cn-xercesc
${KROOT_IDIR}/compute --with-cn-libmonitor ${KROOT_IDIR}/compute --with-cn-
libunwind ${KROOT_IDIR}/compute --with-cn-dyninst ${KROOT_IDIR}/compute --with-
```

```
cn-papi ${PAPI_IDIR} --with-cn-cbtf-krell ${KROOT_IDIR}/compute --with-cn-cbtf
${CBTF_IDIR}/compute --with-binutils ${KROOT_IDIR} --with-boost ${KROOT_IDIR} --
with-mrnet ${KROOT_IDIR} --with-xercesc ${KROOT_IDIR} --with-libmonitor
${KROOT_IDIR} --with-libunwind ${KROOT_IDIR} --with-dyninst ${KROOT_IDIR} --with-
papi ${PAPI_IDIR} --with-alps ${ALPS_IDIR}

./install-tool --target-arch cray --build-onlyosscbtf --openss-prefix ${OSSCBTF_IDIR} --
with-cn-cbtf-krell ${CBTF_IDIR}/compute --krell-root-prefix ${KROOT_IDIR} --with-mpich
${MPICH_IDIR} --with-boost ${KROOT_IDIR} --with-mrnet ${KROOT_IDIR} --with-xercesc
${KROOT_IDIR} --with-libmonitor ${KROOT_IDIR} --with-libunwind ${KROOT_IDIR} --
with-dyninst ${KROOT_IDIR} --with-libelf ${KROOT_IDIR} --with-libdwarf ${KROOT_IDIR}
--with-binutils ${KROOT_IDIR} --cbtf-prefix ${CBTF_IDIR} --with-papi ${PAPI_IDIR}

./install-tool --build-offline --openss-prefix ${OSSOFF_IDIR} --with-runtime-dir
${OSSOFF_IDIR}/compute --krell-root-prefix ${KROOT_IDIR} --with-mpich ${MPICH_IDIR}
--with-python ${PYTHON_IDIR} --with-papi ${PAPI_IDIR}

# Compiling with a module loaded non-default installed compiler requires that the
libstc++ library be made available on the compute nodes
cp /opt/gcc/4.8.2/snos/lib64/libstdc++.so.6 ${KROOT_IDIR}/lib64/libstdc++.so.6
chmod 755 ${KROOT_IDIR}/lib64/libstdc++.so.6
cp /opt/gcc/4.8.2/snos/lib64/libstdc++.so.6
${KROOT_IDIR}/compute/lib64/libstdc++.so.6
chmod 755 ${KROOT_IDIR}/compute/lib64/libstdc++.so.6
```

Spirit **spirit.afrl.hpc.mil** **AFRL**

```
#!/bin/bash

export MODULESHOME=/usr/share/Modules
. $MODULESHOME/init/bash

#export BASE_IDIR=/work1/home/galarowi/OSS2
export BASE_IDIR=/work1/app/comenv/pkgs/openss
export TOOL_VERS="_v2.2.3"
export KROOT_IDIR=${BASE_IDIR}/krellroot${TOOL_VERS}
export CBTF_IDIR=${BASE_IDIR}/cbtf${TOOL_VERS}
export OSSCBTF_IDIR=${BASE_IDIR}/osscbtf${TOOL_VERS}
export OSSOFF_IDIR=${BASE_IDIR}/ossoff${TOOL_VERS}
export MPT_IDIR=/app/sgi/mpt/mpt-2.12

module unload compiler/intel
module load /work1/home/galarowi/privatemodules/cmake-3.2.2
module load gcc-compilers/4.8.4

export CC=gcc
export cc=gcc
export CXX=g++

./install-tool --build-krell-root --krell-root-prefix ${KROOT_IDIR} --with-mpt ${MPT_IDIR}
--force-sqlite-build

./install-tool --build-offline --openss-prefix ${OSSOFF_IDIR} --krell-root-prefix
${KROOT_IDIR} --with-mpt ${MPT_IDIR}

./install-tool --build-cbtf-all --cbtf-prefix ${CBTF_IDIR} --krell-root-prefix ${KROOT_IDIR} -
-with-mpt ${MPT_IDIR}

./install-tool --build-onlyosscbtf --openss-prefix ${OSSCBTF_IDIR} --cbtf-prefix
${CBTF_IDIR} --krell-root-prefix ${KROOT_IDIR} --with-mpt ${MPT_IDIR}

# Need this when executing OSS on compute nodes.   Non-standard installation compiler
usage requires
# the module be loaded at run time or having the libstdc++ library available
cp /app/gmpapp/gcc/platform/gcc-4.8.4/lib64/libstdc++.so.6
${KROOT_IDIR}/lib64/libstdc++.so.6
```

Thunder **thunder.afrl.hpc.mil** **AFRL**

```
#!/bin/bash

export MODULESHOME=/hafs_x86_64/devel/share/Modules
. $MODULESHOME/init/bash

module purge
module load gcc-compilers/4.8.4
module load /p/home/galarowi/privatemodules/cmake-3.2.2

export cc=gcc
export CC=gcc
export CXX=g++

BASE_DIR=/app/comenv/pkgs/openss
TOOL_VERS="_v2.2.3"
MPT_DIR=/app/sgi/mpt-2.12-sgi712r26
PTYHON_IROOT=${BASE_DIR}/python-2.7.3
```

```
PYTHON_VERS="2.7"
KROOT_IDIR=${BASE_DIR}/krellroot${TOOL_VERS}
CBTF_IDIR=${BASE_DIR}/cbtf${TOOL_VERS}
OSSCBTF_IDIR=${BASE_DIR}/osscbtf${TOOL_VERS}
OSSOFF_IDIR=${BASE_DIR}/ossoff${TOOL_VERS}

# build once before running the rest of the script, also need module file or set PATH to
point to installation directory.
# Or if administrator, install cmake 3.0.2 or above for use
./install-tool --build-cmake --krell-root-prefix ${BASE_DIR}/cmake-3.2.2
export PATH=${BASE_DIR}/cmake-3.2.2/bin:$PATH

./install-tool --build-python --krell-root-prefix ${PTYHON_IROOT}

./install-tool --build-krell-root --krell-root-prefix ${KROOT_IDIR} --with-mpt ${MPT_DIR} -
-with-python ${PTYHON_IROOT} --with-python-vers ${PYTHON_VERS}

./install-tool --build-offline --openss-prefix ${OSSOFF_IDIR} --krell-root-prefix
${KROOT_IDIR} --with-mpt ${MPT_DIR} --with-python ${PTYHON_IROOT} --with-python-
vers ${PYTHON_VERS}

./install-tool --build-cbtf-all --cbtf-prefix ${CBTF_IDIR} --krell-root-prefix ${KROOT_IDIR} -
-with-mpt ${MPT_DIR} --with-python ${PTYHON_IROOT} --with-python-vers
${PYTHON_VERS}

./install-tool --build-onlyosscbtf --openss-prefix ${OSSCBTF_IDIR} --cbtf-prefix
${CBTF_IDIR} --krell-root-prefix ${KROOT_IDIR} --with-mpt ${MPT_DIR} --with-python
${PTYHON_IROOT} --with-python-vers ${PYTHON_VERS}

# Need this when executing OSS on compute nodes.   Non-standard installation compiler
usage requires the module be loaded at run time or having the libstdc++ library
available
cp /app/gmpapp/gcc/platform/gcc-4.8.4/lib64/libstdc++.so.6
${KROOT_IDIR}/lib64/libstdc++.so.6
```

Topaz **topaz.erdc.hpc.mil** **ERDC**

```
#!/bin/bash

export MODULESHOME=/usr/share/modules
. $MODULESHOME/init/bash
```

```
module unload compiler/intel
module load compiler/gcc/4.9.3

export cc=gcc
export CC=gcc
export CXX=g++

BASE_DIR=/app/unsupported/PETtools/CE/pkgs/openss
TOOL_VERS="_v2.2.3"
MPT_DIR=/p/home/apps/sgi/mpt-2.12-sgi712r26

./install-tool --build-krell-root --krell-root-prefix ${BASE_DIR}/krellroot${TOOL_VERS} --
with-mpt ${MPT_DIR}

./install-tool --build-offline --openss-prefix ${BASE_DIR}/ossoff${TOOL_VERS} --krell-
root-prefix ${BASE_DIR}/krellroot${TOOL_VERS} --with-mpt ${MPT_DIR}

./install-tool --build-cbtf-all --cbtf-prefix ${BASE_DIR}/cbtf${TOOL_VERS} --krell-root-
prefix ${BASE_DIR}/krellroot${TOOL_VERS} --with-mpt /${MPT_DIR}

./install-tool --build-onlyosscbtf --openss-prefix ${BASE_DIR}/osscbtf${TOOL_VERS} --
cbtf-prefix ${BASE_DIR}/cbtf${TOOL_VERS} --krell-root-prefix
${BASE_DIR}/krellroot${TOOL_VERS} --with-mpt ${MPT_DIR}

# Need this when executing OSS on compute nodes.   Non-standard installation compiler
usage requires the module be loaded at run time or having the libstdc++ library
available
cp /apps/gnu_compiler/4.9.3/lib64/libstdc++.so.6
${BASE_DIR}/krellroot${TOOL_VERS}/lib64/libstdc++.so.6
```