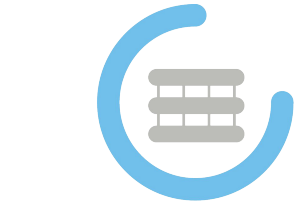
Getting Started with OpenHPC



https://openhpc.community

Reese Baird, Karl Schulz, Derek Simmel,
Nirmala Sundararajan, and Eric Van Hensbergen
OpenHPC Technical Steering Committee Members

PEARC'17 Tutorial
July 10, 2017 ◆ New Orleans, LA

PEARC17

Agenda

- Phase 1 (9-10:30am)
  - presentation and Q/A
  - ssh setup to Lab hardware

<-- Break (10:30am-11am) -->

- Phase 2 (11-12:30pm)
  - Hands-on system installations

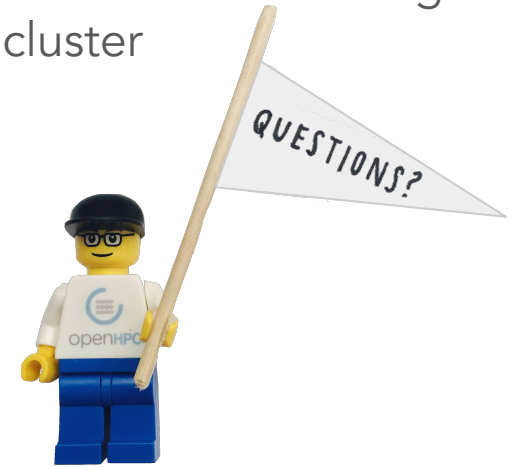
<-- Lunch (12:30 -1:30pm) -->

Goals for the Tutorial

What do we hope to achieve?

- Provide a reasonable overview of the project
- Highlight software available and elements of the hierarchical build/user/test environment
- Complete a basic installation so you feel comfortable using the available recipes to install a local HPC cluster

This is intended to be interactive throughout. Please feel free to ask any questions along the way.



Lab Requirements

- We are going to be leveraging community hardware housed at TACC
- Basic requirement is to have a working ssh client
  - will be breaking up into small groups for cluster installs
  - login credentials will be provided for assigned cluster hardware
  - you should be comfortable with Linux command-line interaction

Conventions and survey

- [key takeaways]
  - we will occasionally be noting key takeaways we think are important
- package manager conventions
  - most of the examples contained herein assume `yum` as the underlying package manager
  - SLES uses `zypper` instead, the discussion also applies to SLES with minor changes to package manager syntax (and SLES specific docs call out zypper commands)
- note: this presentation is available online at: https://goo.gl/NyiDmr
- surveys available at the end of the tutorial

Available Online Resources

- OpenHPC Home: http://www.openhpc.community/
- Primary GitHub Site: https://github.com/openhpc/ohpc
- Package Repositories: http://build.openhpc.community/OpenHPC:/
- OBS Frontend: https://build.openhpc.community
- Component Submission: https://github.com/openhpc/submissions
- System Registry: System Registration Form
- CI Infrastructure: http://test.openhpc.community:8080
- OpenHPC Wiki: https://github.com/openhpc/ohpc/wiki
  - includes links to overview paper and past presentations

- Mailing Lists: http://www.openhpc.community/support/mail-lists/
  - openhpc-announce
  - openhpc-users
  - openhpc-devel

Phase I - Outline

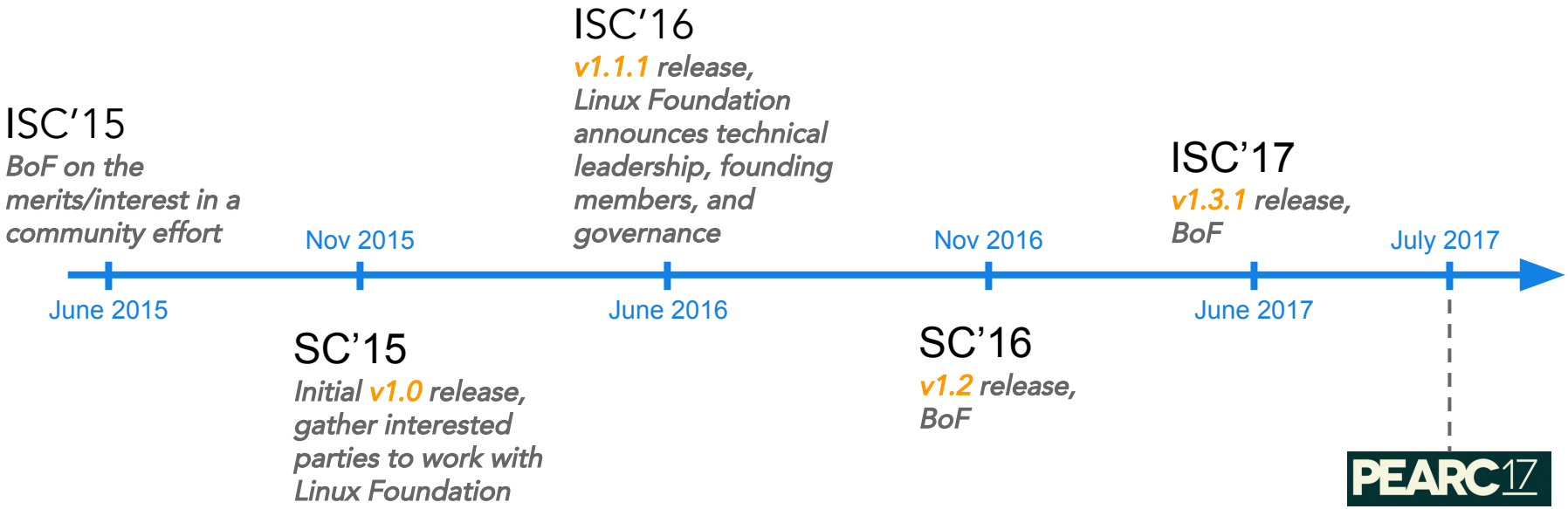
- Project mission/vision
- Brief history
- Target system architecture
- Stack overview
- Packaging conventions
- Module conventions
- Development environment
- Integration testing
- Upgrade paths

OpenHPC - Mission and Vision

Mission: to provide a reference collection of open-source HPC software components and best practices, lowering barriers to deployment, advancement, and use of modern HPC methods and tools.

Vision: OpenHPC components and best practices will enable and accelerate innovation and discoveries by broadening access to state-of-the-art, open-source HPC methods and tools in a consistent environment, supported by a collaborative, worldwide community of HPC users, developers, researchers, administrators, and vendors.

OpenHPC: a brief History…

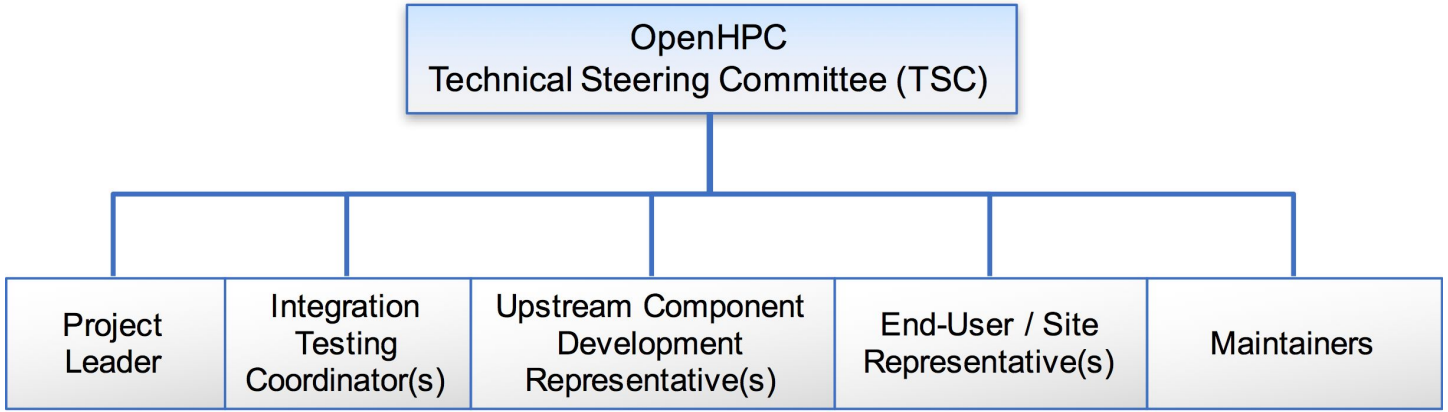


Current Project Members

Mixture of academics, labs, and industry

OpenHPC Technical Steering Committee (TSC)

Role Overview



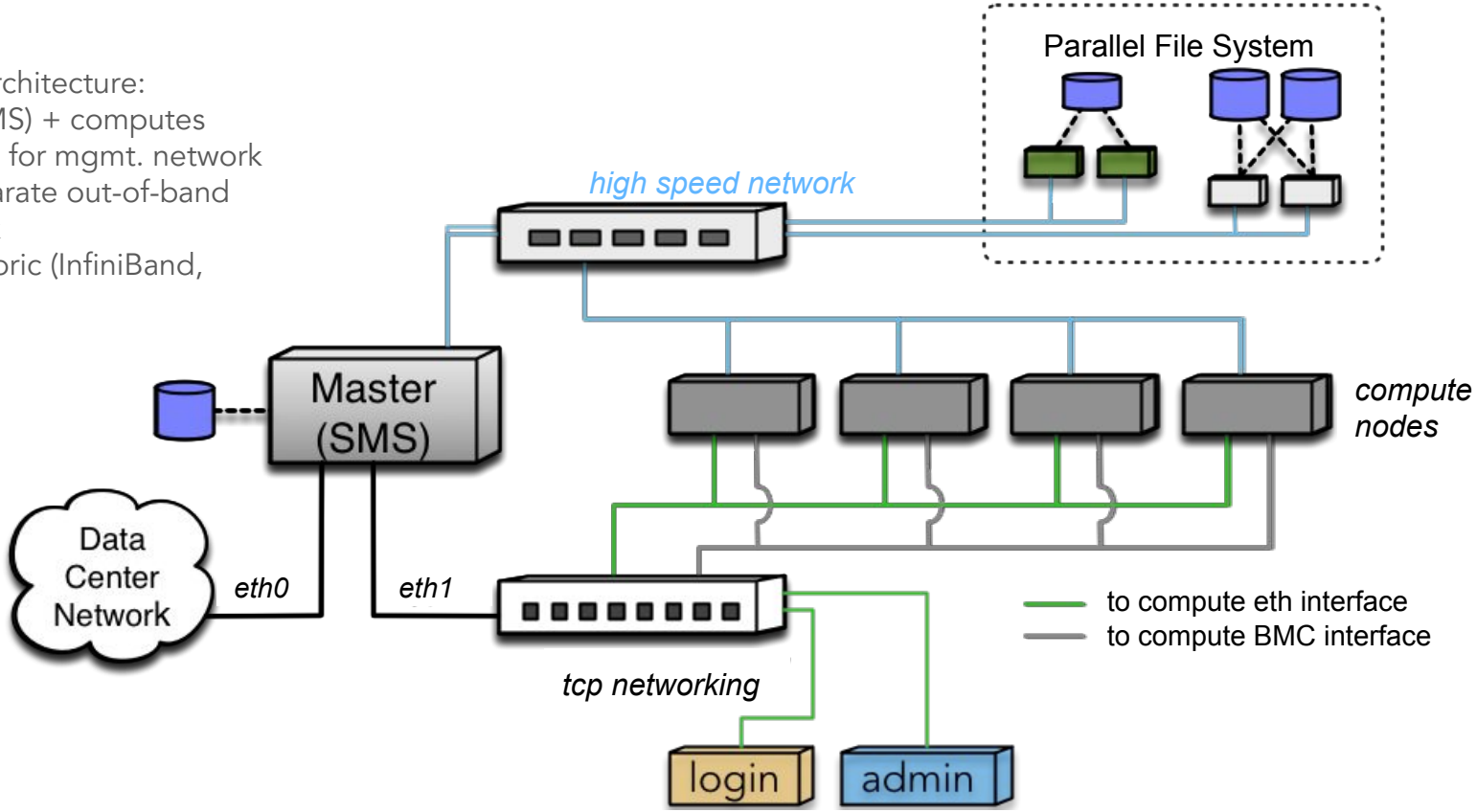
Note: we are currently accepting nominees for TSC volunteers for the next year:

- see recent post to openhpc-users list or email openhpc-tsc-nominations@lists.openhpc.community
- deadline is July 12

LINUX FOUNDATION
COLLABORATIVE PROJECTS

Target system architecture

- Basic cluster architecture: head node (SMS) + computes
- Ethernet fabric for mgmt. network
- Shared or separate out-of-band (BMC) network
- High speed fabric (InfiniBand, Omni-Path)



OpenHPC v1.3.1 - Current S/W components

Functional Areas	Components components available 67 new additions 3 updates 44%
Base OS	CentOS 7.3, SLES12 SP2
Architecture	x86_64, aarch64 (Tech Preview)
Administrative Tools	Conman, Ganglia, Lmod, LosF, Nagios, pdsh, pdsh-mod-slurm, prun, EasyBuild, ClusterShell, mrsh, Genders, Shine, Spack, test-suite
Provisioning	Warewulf, xCAT
Resource Mgmt.	SLURM, Munge, PBS Professional
Runtimes	OpenMP, OCR, Singularity
I/O Services	Lustre client (community version), BeeGFS client
Numerical/Scientific Libraries	Boost, GSL, FFTW, Metis, PETSc, Trilinos, Hypre, SuperLU, SuperLU_Dist, Mumps, OpenBLAS, Scalapack
I/O Libraries	HDF5 (pHDF5), NetCDF (including C++ and Fortran interfaces), Adios
Compiler Families	GNU (gcc, g++, gfortran),
MPI Families	MVAPICH2, OpenMPI, MPICH
Development Tools	Autotools (autoconf, automake, libtool), Valgrind,R, SciPy/NumPy, hwloc
Performance Tools	PAPI, IMB, mpiP, pdtoolkit TAU, Scalasca, ScoreP, SIONLib

new with v1.3.1

- Additional dependencies not provided by BaseOS or community repos are also included

Future additions approved for inclusion:

- PLASMA
- SLEPc
- pNetCDF
- Scotch
- PMIx
- Clang/LLVM

OpenHPC: a building block repository

[Key takeaway #1]

- OpenHPC provides a collection of pre-built ingredients common in HPC environments; fundamentally it is a software repository
- The repository is published for use with Linux distro package managers
  - yum (CentOS/RHEL)
  - zypper (SLES)
- You can pick relevant bits of interest for your site
  - if you prefer a resource manager that is not included, you can build that locally and still leverage the scientific libraries and development environment
  - similarly, you might prefer to utilize a different provisioning system

Enabling an OpenHPC Repo

We currently provide 4 package repositories:

- 2 distros (CentOS7, SLES12)
- 2 architectures (aarch64,x86_64)

# of pre-packaged RPMs available (v1.3.1)

Base OS	x86_64	aarch64	noarch
CentOS 7.3	583	361	60
SLES 12 SP2	587	363	60

Repository Enablement

- Public repositories are available at: http://build.openhpc.community/OpenHPC:/
- There are two primary ways to access the available RPMs
  - enable public OpenHPC repo(s) on head node that can route to internet
  - host a local mirror of the OpenHPC repo(s) within your datacenter
- To facilitate the remote network access option, we provide an "ohpc-release" RPM
  - downloadable from front page of GitHub site (https://github.com/openhpc/ohpc)
  - here is an example install for CentOS/x86 and latest v1.3.x release:

```
# export OHPC_GITHUB=https://github.com/openhpc/ohpc/releases/download
# yum -y install $OHPC_GITHUB/v1.3.GA/ohpc-release-1.3-1.el7.x86_64.rpm
```

** We will use this approach in the lab*

Repository Enablement (external network access)

- Note that ohpc-release configures two repos:

```
# yum repolist
Loaded plugins: langpacks
repo id                                  repo name                                                status
OpenHPC                                  OpenHPC-1.3 - Base                                         821
OpenHPC-updates                          OpenHPC-1.3 - Updates                                      686
base                                     CentOS-7 - Base                                          9,363
epel                                     Extra Packages for Enterprise Linux 7 - x86_64          11,854
```

- With the initial release of a minor branch (e.g. v1.3), the [Updates] repo will be empty
- As subsequent micro releases are available (e.g. v1.3.1, v1.3.2, etc), the [Updates] repo will be populated to match latest release

Repository Enablement (local mirror)

- If your system does not have external network access, or you just prefer to mirror things locally, standalone tar archives are available at:

  http://build.openhpc.community/dist

- A "make_repo.sh" script is provided that will set up a locally hosted OpenHPC repository using the contents from downloaded tarball

Index of /dist/1.3.1

Name	Last modified	Size
Parent Directory		-
OpenHPC-1.3.1.CentOS_7.aarch64.tar	2017-06-16 20:54	1.1G
OpenHPC-1.3.1.CentOS_7.src.tar	2017-06-16 20:57	9.2G
OpenHPC-1.3.1.CentOS_7.x86_64.tar	2017-06-16 20:55	2.8G
OpenHPC-1.3.1.SLE_12.aarch64.tar	2017-06-16 20:50	839M
OpenHPC-1.3.1.SLE_12.src.tar	2017-06-16 20:52	8.8G
OpenHPC-1.3.1.SLE_12.x86_64.tar	2017-06-16 20:50	2.3G
OpenHPC-1.3.1.md5s	2017-06-16 21:03	1.6K

```
# tar xf OpenHPC-1.3.1.CentOS_7.x86_64.tar
# ./make_repo.sh
--> Creating OpenHPC.local.repo file in /etc/yum.repos.d
--> Local repodata stored in /root/repo

# yum repolist | grep OpenHPC
OpenHPC-local           OpenHPC-1.3 - Base
OpenHPC-local-updates   OpenHPC-1.3.1 - Updates
```

Quick installation example

- Once an OpenHPC repo is enabled, package installation follows standard package manager semantics
- Package interdependencies maintained within RPMs
- Consider example install of PETSc for gnu7/mvapich2 on bare bones head node

```
# yum install petsc-gnu7-mvapich2-ohpc
…
================================================================================
 Package                    Arch          Version                Repository   Size
================================================================================
Installing:
 petsc-gnu7-mvapich2-ohpc   x86_64        3.7.6-104.1            OpenHPC-updates  13 M
Installing for dependencies:
 gnu7-compilers-ohpc        x86_64        7.1.0-29.3             OpenHPC-updates  51 M
 mvapich2-gnu7-ohpc         x86_64        2.2-26.2               OpenHPC-updates 8.6 M
 openblas-gnu7-ohpc         x86_64        0.2.19-25.1            OpenHPC-updates 3.4 M
 phdf5-gnu7-mvapich2-ohpc   x86_64        1.10.0-83.1            OpenHPC-updates 2.9 M
 prun-ohpc                  noarch        1.1-21.1               OpenHPC         9.5 k
 scalapack-gnu7-mvapich2-ohpc x86_64      2.0.2-39.1             OpenHPC-updates 3.4 M

Transaction Summary
================================================================================
```

Hierarchical Software Environment

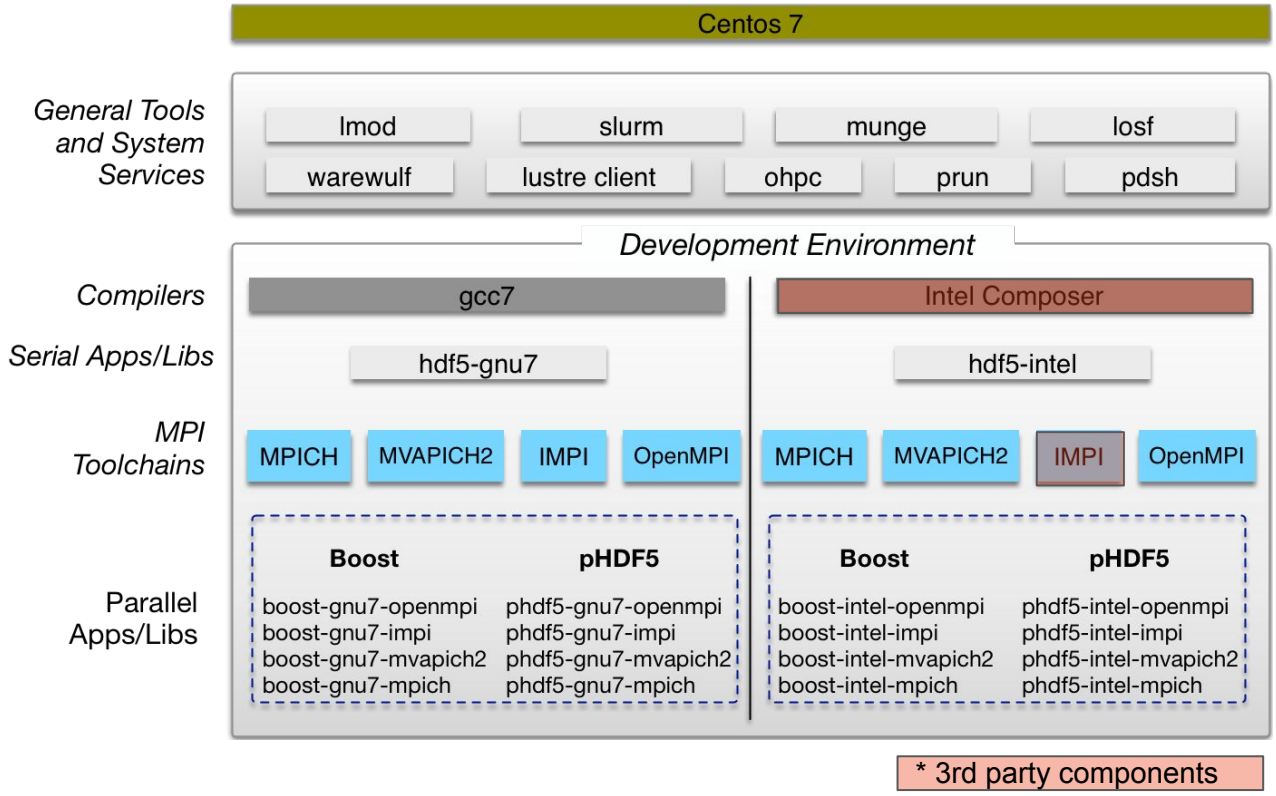
Environment modules

- OpenHPC uses the Lmod implementation of environmental modules to provide standard user development and runtime environments.
- Lmod is a Lua based module system that easily handles the MODULEPATH Hierarchical problem. Environment Modules provide a convenient way to dynamically change the users' environment through modulefiles.
- https://www.tacc.utexas.edu/research-development/tacc-projects/lmod

Hierarchical Software Environment

[Key takeaway #2]

- The designation of compiler/MPI variants are fundamental to OpenHPC's build, test, and end-user environment



Hierarchical Software Environment

- Packaging adopts a consistent environment variable schema in modulefiles:
  - `${PACKAGE_DIR}` - top level install path of package
  - `${PACKAGE_BIN}` - path to binaries supplied by package
  - `${PACKAGE_INC}` - path to include headers for package
  - `${PACKAGE_LIB}` - path to dynamic libraries for package
- Recommend using this in build scripts, Makefiles, and interactive execution
- Let's consider an end user interaction example with previous PETSc install: assume we are a user wanting to build a PETSC hello world in C

```
$ module load gnu7
$ module load petsc
$ mpicc -I$PETSC_INC petsc_hello.c -L$PETSC_LIB -lpetsc
```

Hierarchical Software - the user experience

Goal is to have a component hierarchy reflected in the user environment

- End user sees compatible software based on the currently loaded environment
- Modulefiles for components relying on compiler/MPI variants leverage the "family" capability of Lmod
- If you switch to a different MPI variant, other related modules are updated automatically....

```
$ module list
Currently Loaded Modules:
    1) autotools   2) prun/1.1   3) gnu7/7.1.0   4) mvapich2/2.2   5) ohpc   6) boost/1.63.0
$ module avail
------------------------------ /opt/ohpc/pub/moduledeps/gnu7-mvapich2 ------------------------------
   adios/1.11.0        imb/4.1             netcdf-fortran/4.4.4    scalapack/2.0.2    sionlib/1.7.1
   boost/1.63.0 (L)    mpiP/3.4.1          netcdf/4.4.1.1          scalasca/2.3.1     superlu_dist/4.2
   fftw/3.3.6          mumps/5.1.1         petsc/3.7.6             scipy/0.19.0       tau/2.26.1
   hypre/2.11.1        netcdf-cxx/4.3.0    phdf5/1.10.0            scorep/3.0         trilinos/12.10.1

---------------------------------- /opt/ohpc/pub/moduledeps/gnu7 ----------------------------------
   R/3.3.3    hdf5/1.10.0    mpich/3.2           numpy/1.12.1    openblas/0.2.19    pdtoolkit/3.23
   gsl/2.3    metis/5.1.0    mvapich2/2.2 (L)    ocr/1.0.1       openmpi/1.10.7     superlu/5.2.1

-------------------------------------- /opt/ohpc/pub/modulefiles --------------------------------------
   EasyBuild/3.2.1        clustershell/1.7.3        hwloc/1.11.6        papi/5.5.1        singularity/2.3
   autotools       (L)    gnu7/7.1.0         (L)    ohpc         (L)    prun/1.1   (L)    valgrind/3.12.0

$ echo $BOOST_LIB
/opt/ohpc/pub/libs/gnu7/mvapich2/boost/1.63.0/lib
$ module swap mvapich2 openmpi
Due to MODULEPATH changes the following have been reloaded:
1) boost/1.63.0
$ echo $BOOST_LIB
/opt/ohpc/pub/libs/gnu7/openmpi/boost/1.63.0/lib
```

Hierarchical Software Environment

Another goal is support the coexistence of multiple variants:

- gnu7 variant introduced in latest v1.3.1 release
- previous gnu variant based on gcc 5.x
- example: admin upgraded from v1.3 to v1.3.1 and opted-in to install gnu7 variant

```
$ module list
Currently Loaded Modules:
  1) autotools   2) prun/1.1   3) gnu7/7.1.0   4) mvapich2/2.2   5) ohpc

$ module avail

--------------------------- /opt/ohpc/pub/moduledeps/gnu7-mvapich2 -----------------------------
   adios/1.11.0    imb/4.1           netcdf-cxx/4.3.0        scalapack/2.0.2    sionlib/1.7.1
   boost/1.63.0    mpiP/3.4.1        netcdf-fortran/4.4.4    scalasca/2.3.1     superlu_dist/4.2
   fftw/3.3.6      mumps/5.1.1       petsc/3.7.6             scipy/0.19.0       tau/2.26.1
   hypre/2.11.1    netcdf/4.4.1.1    phdf5/1.10.0            scorep/3.0         trilinos/12.10.1

------------------------------ /opt/ohpc/pub/moduledeps/gnu7 -------------------------------
   R_base/3.3.3    metis/5.1.0         numpy/1.12.1       openmpi/1.10.7
   gsl/2.3         mpich/3.2           ocr/1.0.1          pdtoolkit/3.23
   hdf5/1.10.0     mvapich2/2.2 (L)    openblas/0.2.19    superlu/5.2.1

---------------------------------- /opt/ohpc/pub/modulefiles ----------------------------------
   EasyBuild/3.2.1           gnu/5.4.0           ohpc       (L)    singularity/2.3
   autotools          (L)    gnu7/7.1.0   (L)    papi/5.5.1        valgrind/3.12.0
   clustershell/1.7.3        hwloc/1.11.6        prun/1.1   (L)
```

previously installed from 1.3 release

Packaging Conventions

General package naming conventions

- All RPMs provided via OpenHPC repos include "-ohpc" as the suffix for the package name, for example:

```
lmod-ohpc-7.4.8-11.1.aarch64.rpm
```

- Convention for packages built against specific compiler variant:

```
package-<compiler_family>-ohpc-<package_version>-<release>.<arch>.rpm
```

- Convention for packages built against compiler/MPI combination:

```
package-<compiler_family>-<mpi family>-ohpc-<package_version>-<release>.<arch>.rpm
```

- "ohpc" designation useful for queries (also helpful during upgrades)

```
[sms]# yum search petsc-gnu7 ohpc
========================== N/S matched: petsc-gnu7, ohpc ==========================
petsc-gnu7-impi-ohpc.x86_64 : Portable Extensible Toolkit for Scientific Computation
petsc-gnu7-mpich-ohpc.x86_64 : Portable Extensible Toolkit for Scientific Computation
petsc-gnu7-mvapich2-ohpc.x86_64 : Portable Extensible Toolkit for Scientific Computation
petsc-gnu7-openmpi-ohpc.x86_64 : Portable Extensible Toolkit for Scientific Computation
```

example search for ohpc provided versions of PETSc for gnu7

Meta packages

- Meta RPM packages introduced with v1.3.1 release
  - these replace previous use of groups/patterns
  - general naming convention remains the same
  - names that begin with “ohpc-*” are typically metapackages
  - intended to group related collections of RPMs by functionality
- There is no requirement to use, but we leverage them in the recipes for convenience
- Available package list shown in Appendix E

Table 2: **Available OpenHPC Meta-packages**

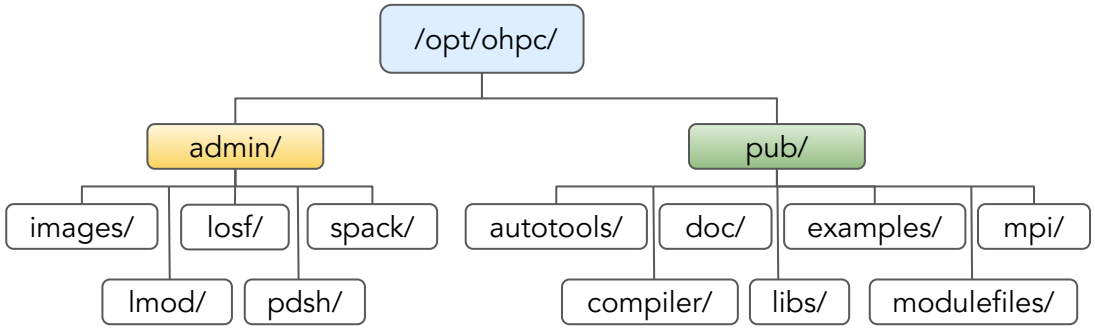
Group Name	Description
ohpc-autotools	Collection of GNU autotools packages.
ohpc-base	Collection of base packages.
ohpc-base-compute	Collection of compute node base packages.
ohpc-ganglia	Collection of Ganglia monitoring and metrics packages.
ohpc-gnu7-io-libs	Collection of IO library builds for use with GNU compiler toolchain.
ohpc-gnu7-mpich-parallel-libs	Collection of parallel library builds for use with GNU compiler toolchain and the MPICH runtime.
ohpc-gnu7-mvapich2-parallel-libs	Collection of parallel library builds for use with GNU compiler toolchain and the MVAPICH2 runtime.
ohpc-gnu7-openmpi-parallel-libs	Collection of parallel library builds for use with GNU compiler toolchain and the OpenMPI runtime.
ohpc-gnu7-parallel-libs	Collection of parallel library builds for use with GNU compiler toolchain.
ohpc-gnu7-perf-tools	Collection of performance tool builds for use with GNU compiler toolchain.
ohpc-gnu7-python-libs	Collection of python related library builds for use with GNU compiler toolchain.
ohpc-gnu7-runtimes	Collection of runtimes for use with GNU compiler toolchain.
ohpc-gnu7-serial-libs	Collection of serial library builds for use with GNU compiler toolchain.
ohpc-intel-impi-parallel-libs	Collection of parallel library builds for use with Intel(R) Parallel Studio XE toolchain and the Intel(R) MPI Library.
ohpc-intel-io-libs	Collection of IO library builds for use with Intel(R) Parallel Studio XE software suite.
ohpc-intel-mpich-parallel-libs	Collection of parallel library builds for use with Intel(R) Parallel Studio XE toolchain and the MPICH runtime.
ohpc-intel-mvapich2-parallel-libs	Collection of parallel library builds for use with Intel(R) Parallel Studio XE toolchain and the MVAPICH2 runtime.
ohpc-intel-openmpi-parallel-libs	Collection of parallel library builds for use with Intel(R) Parallel Studio XE toolchain and the OpenMPI runtime.
ohpc-intel-perf-tools	Collection of performance tool builds for use with Intel(R) Parallel Studio XE toolchain.
ohpc-intel-python-libs	Collection of python related library builds for use with Intel(R) Parallel Studio XE toolchain.
ohpc-intel-runtimes	Collection of runtimes for use with Intel(R) Parallel Studio XE toolchain.
ohpc-intel-serial-libs	Collection of serial library builds for use with Intel(R) Parallel Studio XE toolchain.
ohpc-nagios	Collection of Nagios monitoring and metrics packages.
ohpc-slurm-client	Collection of client packages for SLURM.
ohpc-slurm-server	Collection of server packages for SLURM.
ohpc-warewulf	Collection of base packages for Warewulf provisioning.

Installation Paths

- Installation of end-user oriented components housed under /opt/ohpc/pub
  - shared file-system in the recipes
  - could also be local to nodes for stateful
- RPM packaging relies on macros to aid in top-level path definitions

snippet from OHPC_macros

```
# Top-level OpenHPC installation paths
%define PROJ_NAME ohpc
%define OHPC_HOME /opt/%{PROJ_NAME}
%define OHPC_ADMIN %{OHPC_HOME}/admin
%define OHPC_PUB %{OHPC_HOME}/pub
%define OHPC_COMPILERS %{OHPC_PUB}/compiler
%define OHPC_MPI_STACKS %{OHPC_PUB}/mpi
%define OHPC_APPS %{OHPC_PUB}/apps
%define OHPC_LIBS %{OHPC_PUB}/libs
%define OHPC_MODULES %{OHPC_PUB}/modulefiles
%define OHPC_MODULEDEPS %{OHPC_PUB}/moduledeps
```



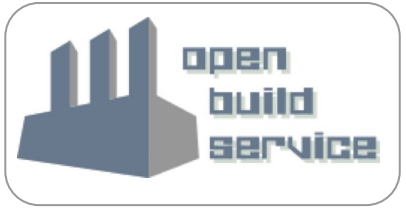
Development Infrastructure

Development Infrastructure

What are we using to get the job done....?

- The usual software engineering stuff
  - GitHub (SCM and issue tracking, milestone planning)
  - continuous integration (CI) testing (Jenkins)
  - documentation (Latex)

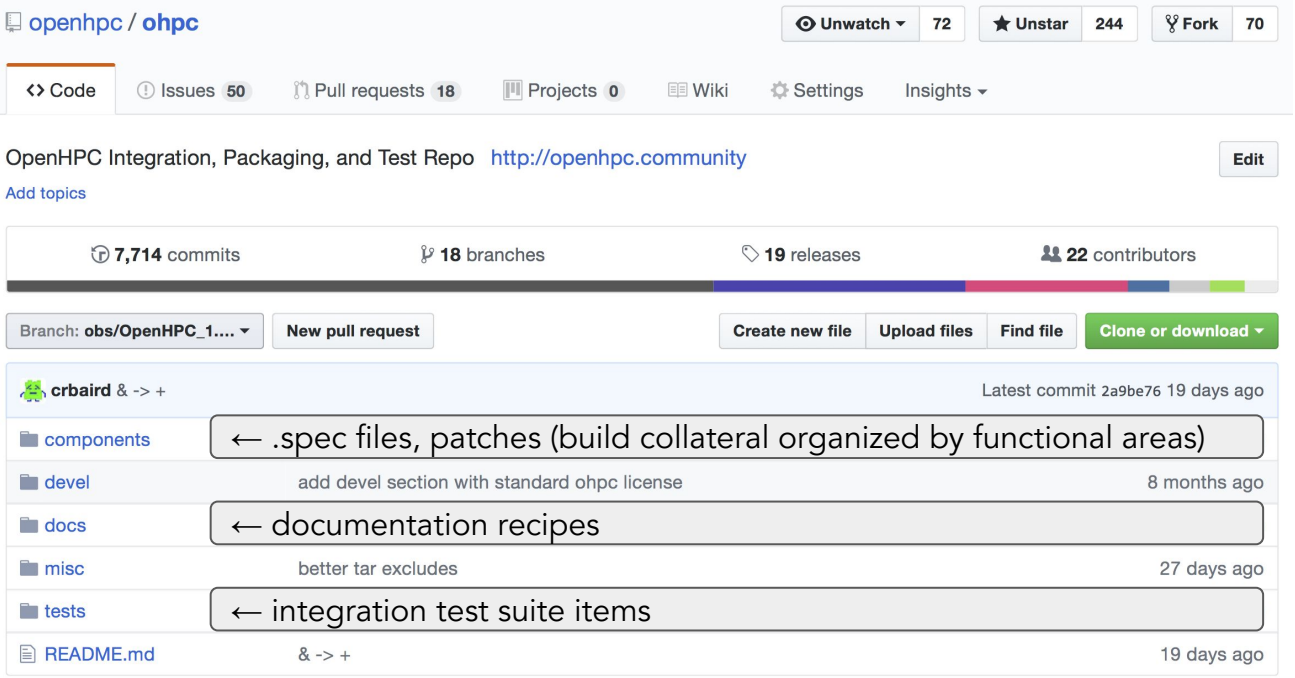
- Capable build/packaging system
  - recall we target a common delivery/access mechanism that adopts Linux sysadmin familiarity - ie. yum/zypper repositories for supported distros
  - require a flexible system to manage builds for multiple distros, multiple compiler/MPI family combinations, and dependencies across packages
  - have engineered a system using Open Build Service (OBS) which is supported by back-end git
    - git houses .spec files, tarballs, patches, documentation recipes, and integration tests
    - OBS performs automated builds and dependency analysis



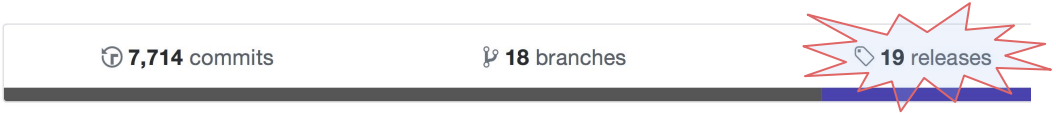
GitHub Overview: Top-level repo layout

Top-level directories of potential interest:

- components/
- docs/
- tests/



GitHub Overview: Release notes



The "releases" section on GitHub is a good source for perusing changes particular to a given release:

- important notices
- general updates
- version changes

https://github.com/openhpc/ohpc/releases

Component Version Changes

* EasyBuild-ohpc	(3.1.2 -> 3.2.1)
* clustershell-ohpc	(1.7.2 -> 1.7.3)
* conman-ohpc	(0.2.7 -> 0.2.8)
* docs-ohpc	(1.3 -> 1.3.1)
* hdf5-intel-ohpc	(1.8.17 -> 1.10.0)
* lmod-defaults-intel-impi-ohpc	(1.2 -> 1.3.1)
* lmod-defaults-intel-mpich-ohpc	(1.2 -> 1.3.1)
* lmod-defaults-intel-mvapich2-ohpc	(1.2 -> 1.3.1)
* lmod-defaults-intel-openmpi-ohpc	(1.2 -> 1.3.1)
* lmod-ohpc	(6.5.11 -> 7.4.8)
* losf-ohpc	(0.53.0 -> 0.54.0)
* mumps-intel-impi-ohpc	(5.0.2 -> 5.1.1)
* mumps-intel-mpich-ohpc	(5.0.2 -> 5.1.1)
* mumps-intel-mvapich2-ohpc	(5.0.2 -> 5.1.1)
* mumps-intel-openmpi-ohpc	(5.0.2 -> 5.1.1)
* nagios-plugins-all-ohpc	(2.1.1 -> 2.2.0)
* nagios-plugins-apt-ohpc	(2.1.1 -> 2.2.0)
* nagios-plugins-breeze-ohpc	(2.1.1 -> 2.2.0)

OpenHPC v1.3.1 (16 June 2017)

Edit

koomie released this 19 days ago · **0 commits** to obs/OpenHPC_1.3_Factory since this release

Release Notes

Important Highlights/Notices

- A new compiler variant (`gnu7`) is introduced with this release. In the case of a fresh install, OpenHPC recipes default to installing the new variant along with matching runtimes and libraries. However, if upgrading a previously installed system, administrators can opt-in to enable the `gnu7` variant . This procedure is detailed in Appendix B of the OpenHPC Install

uded in the `warewulf-httpd.conf` file that ships with the ackage. If upgrading from a version prior to 1.3, the updated ttpd/conf.d/warewulf-httpd.conf.rpmnew locally. You will need oduction file and restart the web server to ensure correct ple for CentOS:

lf-httpd.conf.rpmnew /etc/httpd/conf.d/warewulf-httpd.conf

ecipe (#323)
t enablement for installation recipes (#340)
a packages versus groups/patterns used in previous releases. can be found in Appendix E of the OpenHPC Install Guide(s).

as changed from `R_base-ohpc` to `R-gnu7-ohpc` , and the lmod to R (#472).
(#436)
emplate paths (#423)
optional SLURM add-on (#435)
istency of several administrative packages

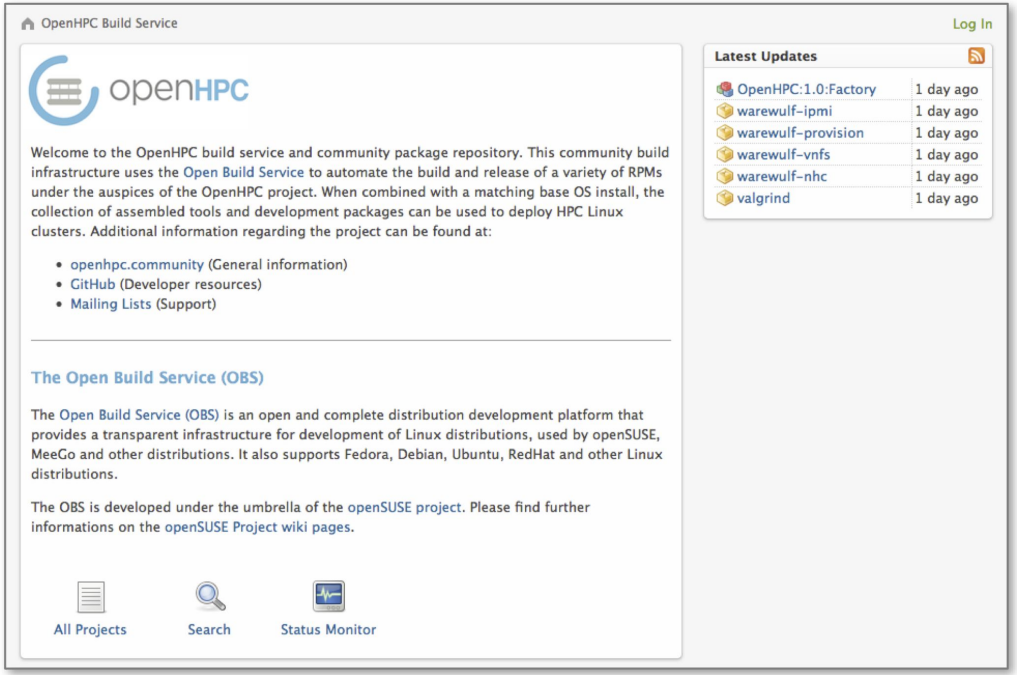
- Lmod packaging updated to conflict SLES Modules package (#440)
- updated .spec files to further centralize compiler/mpi family setup (#447)
- updates to OHPC_macros to support builds outside of OBS (#459)
- variety of component version updates and other additions highlighted below

Build System - OBS

Manages build Process end-to-end:

- OBS can drive builds for multiple OS distros from single input
- Repeatable builds carried out in chroot environment
- Generates binary and src RPMs
- Publishes corresponding package repositories
- Client/server architecture supports distributed build slaves and multiple architectures
- *Note: we can create OBS accounts for developers and folks interested in contributing new packages. Allows testing builds in a standalone HOME project.*

https://build.openhpc.community



Build System Conventions

[Key takeaway #3]

- A single component .spec file is maintained for all compiler/MPI variants
- When changes are made to this .spec file, the build workflow is setup to automatically build all variants (and dependencies)
- macro variables are used within a given package .spec to call out the dependency type per build

Packages with a **compiler** dependency

```
%define ohpc_compiler_dependent 1
%include %{_sourcedir}/OHPC_macros
```

Packages with a **compiler/mpi** dependency

```
%define ohpc_compiler_dependent 1
%define ohpc_mpi_dependent 1
%include %{_sourcedir}/OHPC_macros
```

dependency snippet from OHPC_macros

```
%if "%{compiler_family}" == "gnu7"
BuildRequires: gnu7-compilers%{PROJ_DELIM} >= 7.1.0
Requires:      gnu7-compilers%{PROJ_DELIM} >= 7.1.0
%endif

%if "%{mpi_family}" == "openmpi"
BuildRequires: openmpi-%{compiler_family}%{PROJ_DELIM}
Requires:      openmpi-%{compiler_family}%{PROJ_DELIM}
%endif
```

Build System - package links

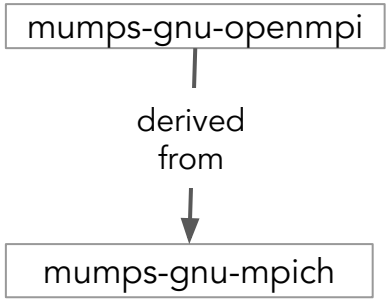
To drive multiple related builds from one .spec file, we leverage OBS's ability to link related packages via a "_link" file:

- default parent build typically set up for gcc compiler variant and OpenMPI
- alternate child builds use same source but override compiler_family and mpi_family macros

mumps-gnu-mpich OBS definition: _link

```
<!-- OpenHPC OBS configuration files -->

<link project='OpenHPC:1.3:Update1:Factory' package='mumps-gnu-openmpi'>
<patches>
  <topadd>%define compiler_family gnu</topadd>
  <topadd>%define mpi_family mpich</topadd>
</patches>
</link>
```

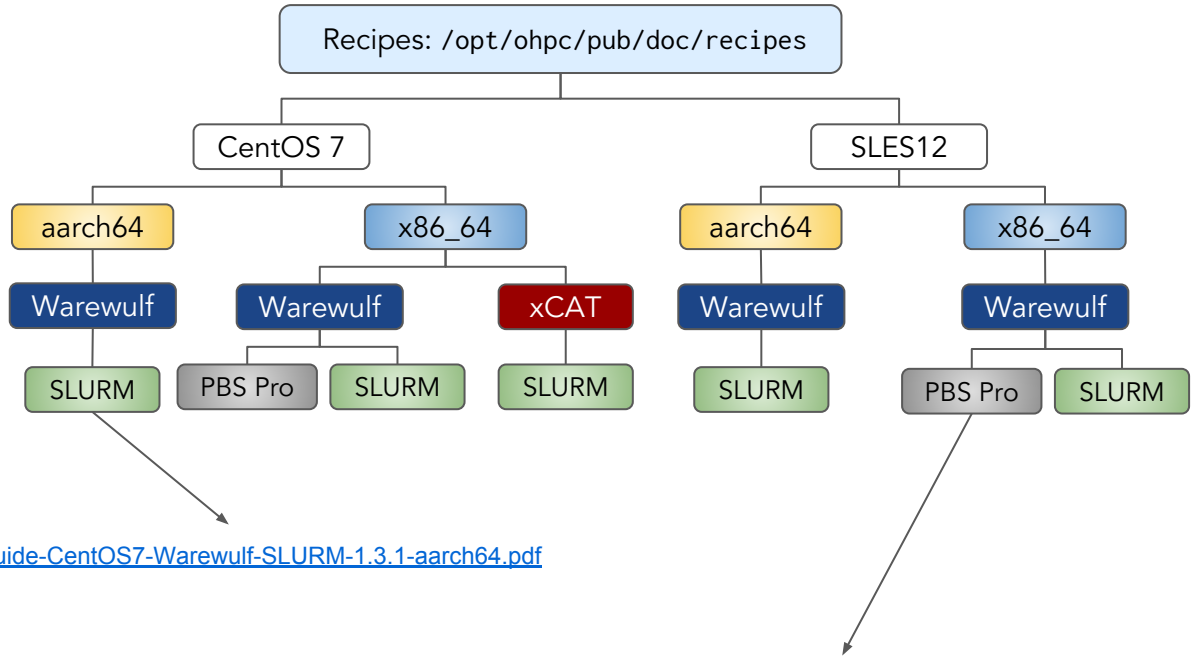


OpenHPC: validated recipes for system install

[Key takeaway #4]

- In addition to being a package repository, OpenHPC provides validated recipes for bare-metal system installs
- Recipes organized by OS, architecture, and key administrative components
- Latest guides always available on main GitHub site:

https://github.com/openhpc/ohpc

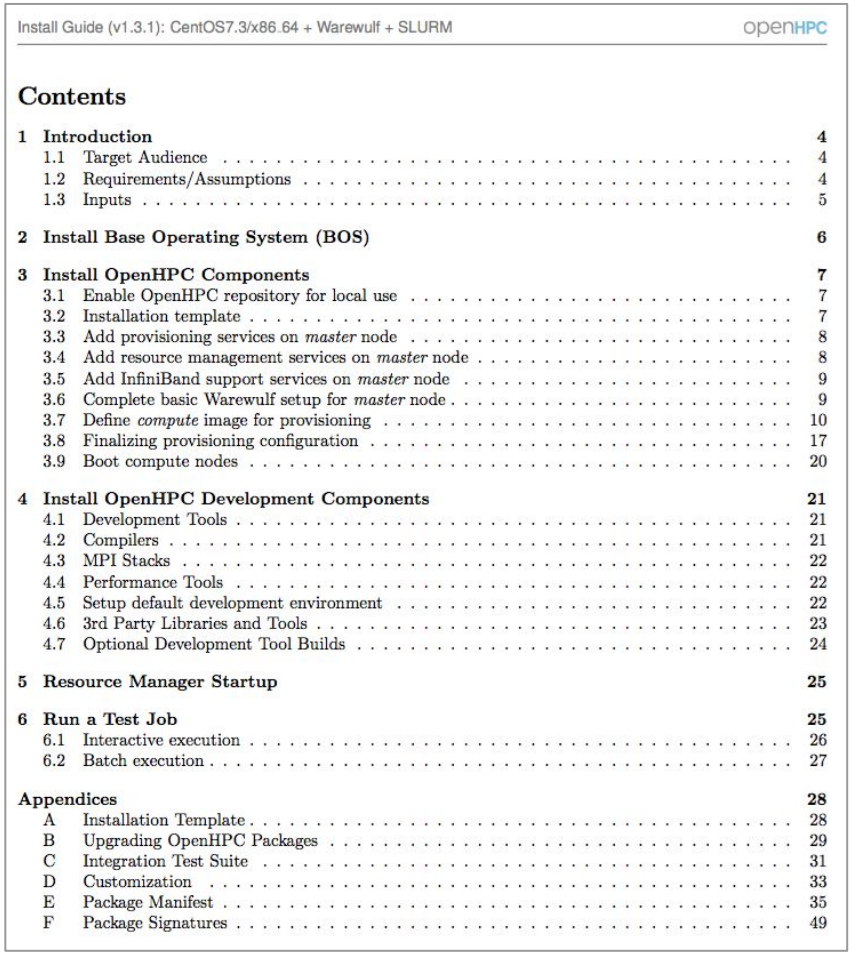


Install_guide-CentOS7-Warewulf-SLURM-1.3.1-aarch64.pdf

Install_guide-SLE_12-Warewulf-PBSPro-1.3.1-x86_64.pdf

OpenHPC: Installation Recipes

- Current recipes are purposefully very transparent on config file edits and assumes Linux familiarity
- Starts by assuming compatible Base OS is installed on system management server (SMS) and proceeds to:
  - install provisioning services on SMS
  - create/customize a compute image (chroot based)
  - install development tools on SMS (exported to compute nodes via NFS)
  - startup resource manager and run test job



Install Guide (v1.3.1): CentOS7.3/x86_64 + Warewulf + SLURM openHPC

Contents

OpenHPC: Installation Recipes (cont.)

- There are a number of optional customizations sprinkled throughout the docs, e.g
  - include parallel file system client
  - syslog forwarding
  - monitoring tools
- Potentially useful tips or optional config options are called out via separate tooltip boxes, e.g.

> **Tip**
>
> By default, Warewulf is configured to provision over the `eth1` interface and the steps below include updating this setting to override with a potentially alternatively-named interface specified by `${sms_eth_internal}`.

- Meta-package convenience packages leveraged to install many of the available development tools, e.g.

```
# Install perf-tools meta-package
[sms]# yum -y install ohpc-gnu7-perf-tools
```

← installs collection of PAPI, TAU, IMB, mpiP, and Scalasca

OpenHPC: Installation Recipes (cont.)

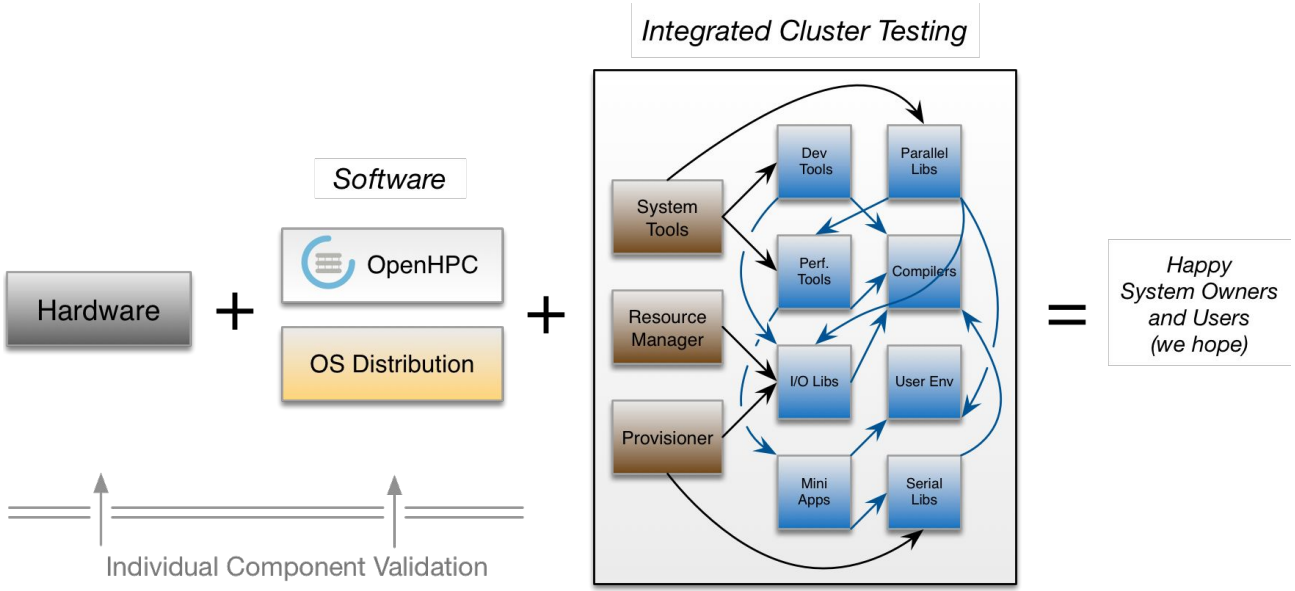
- Recipe guides necessarily have a number of things to "cut-and-paste" if you want to reproduce them locally
- There are also a number of local site-specific settings, e.g.
  - MAC addresses
  - ethernet interface names
- We understand that this can be cumbersome and we also want to have an automated capability to validate the recipes in our CI system:
  - one of the motivations for using a typesetting system like LaTeX
  - we parse all of the commands referenced in the recipes into standalone Bash scripts
  - these template scripts are included in the `docs-ohpc` package
    - can be used with companion `input.local` file to configure site-specific settings
    - provided as convenience for local installation and customization

⬅ we will be leveraging these scripts for the cluster installs in the lab

```
# ls /opt/ohpc/pub/doc/recipes/centos7/*/*/*/recipe.sh
/opt/ohpc/pub/doc/recipes/centos7/aarch64/warewulf/slurm/recipe.sh
/opt/ohpc/pub/doc/recipes/centos7/x86_64/warewulf/pbspro/recipe.sh
/opt/ohpc/pub/doc/recipes/centos7/x86_64/warewulf/slurm/recipe.sh
/opt/ohpc/pub/doc/recipes/centos7/x86_64/xcat/slurm/recipe.sh
```

Integration Testing

- As part of the release workflow, we rely on an integration tests suite
- Intent is to build upon existing validation efforts and augment component-level validation with targeted cluster-level validation:
  - install recipes
  - cross-package interaction
  - development environment
  - upgrade mechanisms



Integration Testing

- Where do we get the tests? Ideally, we leverage directly from the packages we are testing:
  - as an example, we went down this path originally with HDF5
    - discovered the tests that ship with their "make check" actually test internal (non-public) API's
    - did not make sense as the internal header files are not part of a normal HDF5 install
    - ended up using separate collection of tests from HDF5 community that are used to illustrate APIs (only C and Fortran though)
    - we integrated these as a subcomponent and added some companion C++ tests
  - in other cases, we have to cook up the tests from scratch (another great opportunity for community participation)
- Dev environment tests are a mixture of flavors:
  - interactive execution to verify certain binaries are in working order
  - successful compilation to test libraries provided via OpenHPC
  - successful interactive execution
  - tests for module usability and consistency
  - successful remote execution under resource manager

← more than 1,000 jobs submitted to RM when all tests enabled

Integration Test Modularity

** We will use the test-suite in the lab*

- test-suite available as **standalone RPM**
- major components have configuration options to enable/disable
- harness includes capability to repeat tests with different compiler/MPI environments
  - gcc/Intel compiler toolchains
  - MPICH, OpenMPI, MVAPICH2, Intel MPI families

```
Package version.............. : test-suite-1.0.0
Build user................... : jilluser
Build host................... : master4-centos71.localdomain
Configure date............... : 2015-10-26 09:23
Build architecture........... : x86_64-unknown-linux
Test suite configuration..... : long
Submodule Configuration:
User Environment:
        RMS test harness..... : enabled
        Munge................ : enabled
        Apps................. : enabled
        Compilers............ : enabled
        MPI.................. : enabled
        HSN.................. : enabled
        Modules.............. : enabled
        OOM.................. : enabled
Dev Tools:
        Valgrind............. : enabled
        R base package....... : enabled
        TBB.................. : enabled
        CILK................. : enabled
Performance Tools:
        mpiP Profiler........ .... : enabled
        Papi................. : enabled
        PETSc................ : enabled
        TAU.................. : enabled
```

```
Libraries:
        Adios ............... : enabled
        Boost ............... : enabled
        Boost MPI............ : enabled
        FFTW................. : enabled
        GSL.................. : enabled
        HDF5................. : enabled
        HYPRE................ : enabled
        IMB.................. : enabled
        Metis................ : enabled
        MUMPS................ : enabled
        NetCDF............... : enabled
        Numpy................ : enabled
        OPENBLAS............. : enabled
        PETSc................ : enabled
        PHDF5................ : enabled
        ScaLAPACK............ : enabled
        Scipy................ : enabled
        Superlu.............. : enabled
        Superlu_dist......... : enabled
        Trilinos ............ : enabled
Apps:
        MiniFE............... : enabled
        MiniDFT.............. : enabled
        HPCG................. : enabled
```

Community Test System (CI) - Jenkins

openHPC Jenkins search LOG IN

Jenkins ▸ 1.3.x ▸ ENABLE AUTO REFRESH

1.3.x

http://test.openhpc.community:8080

OpenHPC CI Infrastructure
Thanks to the Texas Advanced Computing Center (TACC) for hosting support and to Intel, Cavium, and Dell for hardware donations.

1.3 | **1.3.1** | All

S	Name ↓	Last Success	Last Duration
✓	(1.3 to 1.3.1) - (centos7.3,x86_64) - (warewulf+slurm)	20 days - #41	1 hr 12 min
✓	(1.3.1) - (centos7.3,x86_64) - (warewulf+pbspro) - UEFI	20 days - #390	56 min
✓	(1.3.1) - (centos7.3,x86_64) - (warewulf+slurm) - long cycle	20 days - #892	1 hr 2 min
✓	(1.3.1) - (centos7.3,x86_64) - (warewulf+slurm) - tarball REPO	20 days - #48	1 hr 2 min
✓	(1.3.1) - (centos7.3,x86_64) - (warewulf+slurm+PSXE)	20 days - #726	2 hr 14 min
✓	(1.3.1) - (centos7.3,x86_64) - (warewulf+slurm+PSXE+OPA)	20 days - #80	1 hr 51 min
✓	(1.3.1) - (centos7.3,x86_64) - (xcat+slurm)	20 days - #271	1 hr 1 min
✓	(1.3.1) - (sles12sp2,x86_64) - (warewulf+pbspro) - tarball REPO	20 days - #45	44 min
✓	(1.3.1) - (sles12sp2,x86_64) - (warewulf+pbspro) - UEFI	20 days - #70	45 min
✓	(1.3.1) - (sles12sp2,x86_64) - (warewulf+slurm)	20 days - #780	53 min
✓	(1.3.1) - (sles12sp2,x86_64) - (warewulf+slurm+PSXE) - long cycle	20 days - #114	1 hr 53 min
✓	(1.3.1) - (sles12sp2,x86_64) - (warewulf+slurm+PSXE+OPA)	20 days - #36	1 hr 33 min
✓	(1.3.1) - aarch64 - (centos7.3) - (warewulf+slurm)	21 days - #4	0.35 sec

- All recipes exercised in CI system (Jenkins)
- Start w/ bare-metal installs and then run integration test suite
- We capture results in Jenkins using TAP/Junit logging

All Tests

Package	Duration	Fail	(diff)	Skip	(diff)	Pass	(diff)	Total	(diff)
InstallTests	0.58 sec	0		0		9		9	
RootLevelTests	1 min 10 sec	0		0		37		37	
UserLevelTests	27 min	0		0		984		984	

Multiple Architecture Builds Available

- Starting with v1.2 release, we also include builds for aarch64
  - both SUSE and RHEL/CentOS now have aarch64 variants available for latest versions (SLES 12 SP2, CentOS 7.3)
- Recipes/packages being made available as a Tech Preview
  - some additional work required for provisioning
  - significant majority of development packages testing OK, but there are a few known caveats
  - please see https://github.com/openhpc/ohpc/wiki/ARM-Tech-Preview for latest info
- *Hoping to have full release with compatible provisioning option for SC'17*

Dev Environment Consistency

x86_64

```
[train01@sms001 ~]$ module avail

---------------------- /opt/ohpc/pub/moduledeps/gnu7-mpich ----------------------
   adios/1.11.0    mpiP/3.4.1             petsc/3.7.6        scorep/3.0
   boost/1.63.0    mumps/5.1.1            phdf5/1.10.0       sionlib/1.7.1
   fftw/3.3.6      netcdf-cxx/4.3.0       scalapack/2.0.2    superlu_dist/4.2
   hypre/2.11.1    netcdf-fortran/4.4.4   scalasca/2.3.1     tau/2.26.1
   imb/4.1         netcdf/4.4.1.1         scipy/0.19.0       trilinos/12.10.1

------------------------ /opt/ohpc/pub/moduledeps/gnu7 -------------------------
   R/3.3.3        metis/5.1.0         numpy/1.12.1       openmpi/1.10.7
   gsl/2.3        mpich/3.2    (L)    ocr/1.0.1          pdtoolkit/3.23
   hdf5/1.10.0    mvapich2/2.2        openblas/0.2.19    superlu/5.2.1

-------------------------
   EasyBuild/3.2.1
   autotools          (L)
```

aarch64

```
train01@cavium1:~> module avail

---------------------- /opt/ohpc/pub/moduledeps/gnu7-mpich ----------------------
   adios/1.11.0    mpiP/3.4.1             petsc/3.7.6        scorep/3.0
   boost/1.63.0    mumps/5.1.1            phdf5/1.10.0       sionlib/1.7.1
   fftw/3.3.6      netcdf-cxx/4.3.0       scalapack/2.0.2    superlu_dist/4.2
   hypre/2.11.1    netcdf-fortran/4.4.4   scalasca/2.3.1     tau/2.26.1
   imb/4.1         netcdf/4.4.1.1         scipy/0.19.0       trilinos/12.10.1

------------------------ /opt/ohpc/pub/moduledeps/gnu7 -------------------------
   R/3.3.3        metis/5.1.0         ocr/1.0.1          pdtoolkit/3.23
   gsl/2.3        mpich/3.2    (L)    openblas/0.2.19    superlu/5.2.1
   hdf5/1.10.0    numpy/1.12.1        openmpi/1.10.7

-------------------------- /opt/ohpc/pub/modulefiles --------------------------
   EasyBuild/3.2.1           hwloc/1.11.6        singularity/2.3
   autotools          (L)    ohpc         (L)    valgrind/3.12.0
```

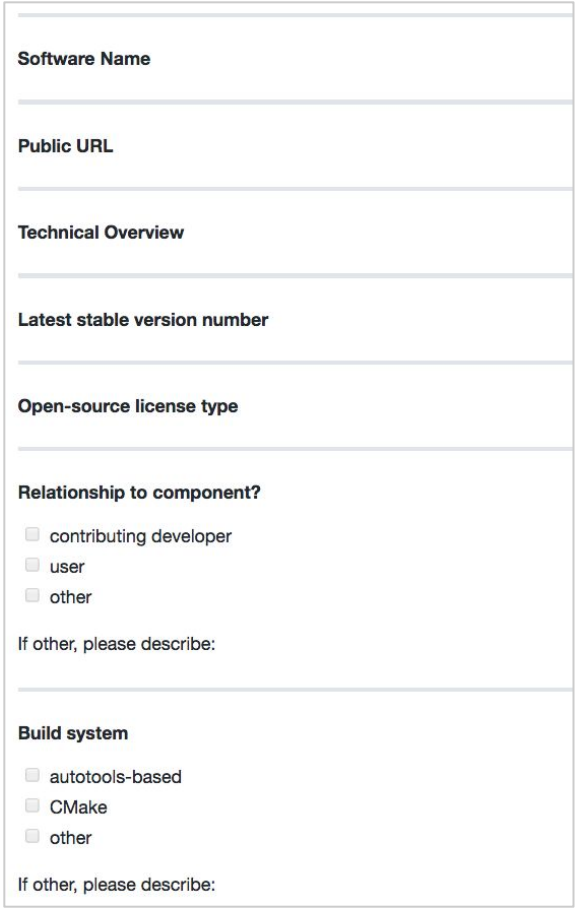
OpenHPC providing consistent development environment to the end user across multiple architectures.

Miscellaneous Items

How to Request Additional Software

- A common question posed to the project has been how to request new software components?
- There is a simple submission site for new requests:
  - https://github.com/openhpc/submissions
  - requests reviewed on rolling basis at roughly a quarterly cadence

Next Submission Deadline: Aug 4, 2017



Software Name

Public URL

Technical Overview

Latest stable version number

Open-source license type

Relationship to component?

- contributing developer
- user
- other

If other, please describe:

Build system

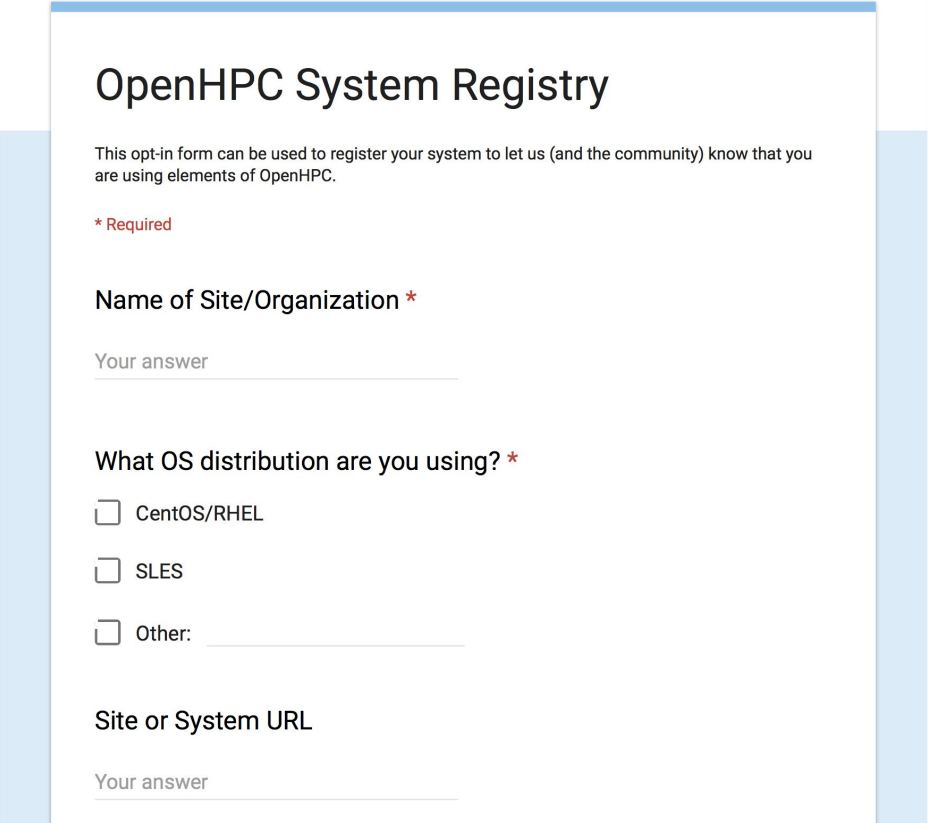
- autotools-based
- CMake
- other

If other, please describe:

System Registry

- Interested sites can register their their usage on a public system registry
- Helpful for us to have an idea as to who is potentially benefitting from the community effort
- Accessible from top-level GitHub page and here:

  System Registry Form



OpenHPC System Registry

This opt-in form can be used to register your system to let us (and the community) know that you are using elements of OpenHPC.

* Required

Name of Site/Organization *

Your answer

What OS distribution are you using? *

- [ ] CentOS/RHEL
- [ ] SLES
- [ ] Other:

Site or System URL

Your answer

Upgrading Installed Systems

OpenHPC was designed with upgradability in mind:

- A rolling updates repository is enabled by default with each minor release (1.2, 1.3, …)
  - Bug fixes and new components are made available in the updates repository
- When upgrading between micro releases on the same branch (e.g. from v1.3 to 1.3.1), there is no need to adjust local package manager configurations
- Upgrading between major or minor releases does require enabling a new repository, but consistent package release numbers are maintained through all releases

Example upgrade steps highlighted in docs (Appendix B)

```
[root@sms001 ~]# yum clean expire-cache
Loaded plugins: fastestmirror
Cleaning repos: OpenHPC OpenHPC-updates base epel
4 metadata files removed
[root@sms001 ~]# yum --installroot=/opt/ohpc/admin/images/centos7.3 clean expire-cache
...

[root@sms001 ~]# yum -y upgrade "*-ohpc"
...
[root@sms001 ~]# yum -y --installroot=/opt/ohpc/admin/images/centos7.3 upgrade "*-ohpc"
...

[root@sms001 ~]# wwvnfs --chroot /opt/ohpc/admin/images/centos7.3
Using 'centos7.3' as the VNFS name
Creating VNFS image from centos7.3
Compiling hybridization link tree                          : 0.08 s
Building file list                                         : 0.34 s
Compiling and compressing VNFS                             : 4.96 s
Adding image to datastore                                  : 9.32 s
Total elapsed time                                         : 14.71 s
```

Availability of 3rd party development libs for PXSE

- If deploying on x86_64, you may also have a desire to use the Intel® Parallel Studio (PXSE) toolchain (compilers and MPI)
- Most of the 3rd party development components are available for use with the recent versions of PXSE
- To enable usage, you would first obtain and install PXSE separately on your system:
  - by default it installs into /opt/intel, but can be overridden to match OHPC paths
  - once installed, use the provided compatibility packages to created OpenHPC style modulefiles
  - can then install desired 3rd party libraries/tools

```
[sms]# yum install intel-compilers-devel-ohpc
...
Scanning top-level dir = /opt/ohpc/pub/intel
--> Installing OpenHPC-style modulefile for version=17.0.4.196

[sms]# yum install intel-mpi-devel-ohpc
[sms]# yum -y install ohpc-intel-mvapich2-parallel-libs
```


See "3rd Party Libraries and Tools" section in docs for more info

Release Roadmap

- Have had some requests for a roadmap for future releases
- High-level roadmap now maintained on GitHub wiki

Release	Target Release Date	Expectations
1.3.2	August 2017	New component additions and version upgrades.
1.3.3	November 2017	New component additions and version upgrades.

Previous Releases

A history of previous OpenHPC releases is highlighted below. Clicking on the version string will take you to the `Release Notes` for more detailed information on the changes in a particular release.

Release	Date
1.3.1	June 16, 2017
1.3	March 31, 2017
1.2.1	January 24, 2017
1.2	November 12, 2016
1.1.1	June 21, 2016
1.1	April 18, 2016
1.0.1	February 05, 2016
1.0	November 12, 2015

https://github.com/openhpc/ohpc/wiki/Release-History-and-Roadmap

Phase II - Hands-on system installations

Phase II (Lab) - Outline

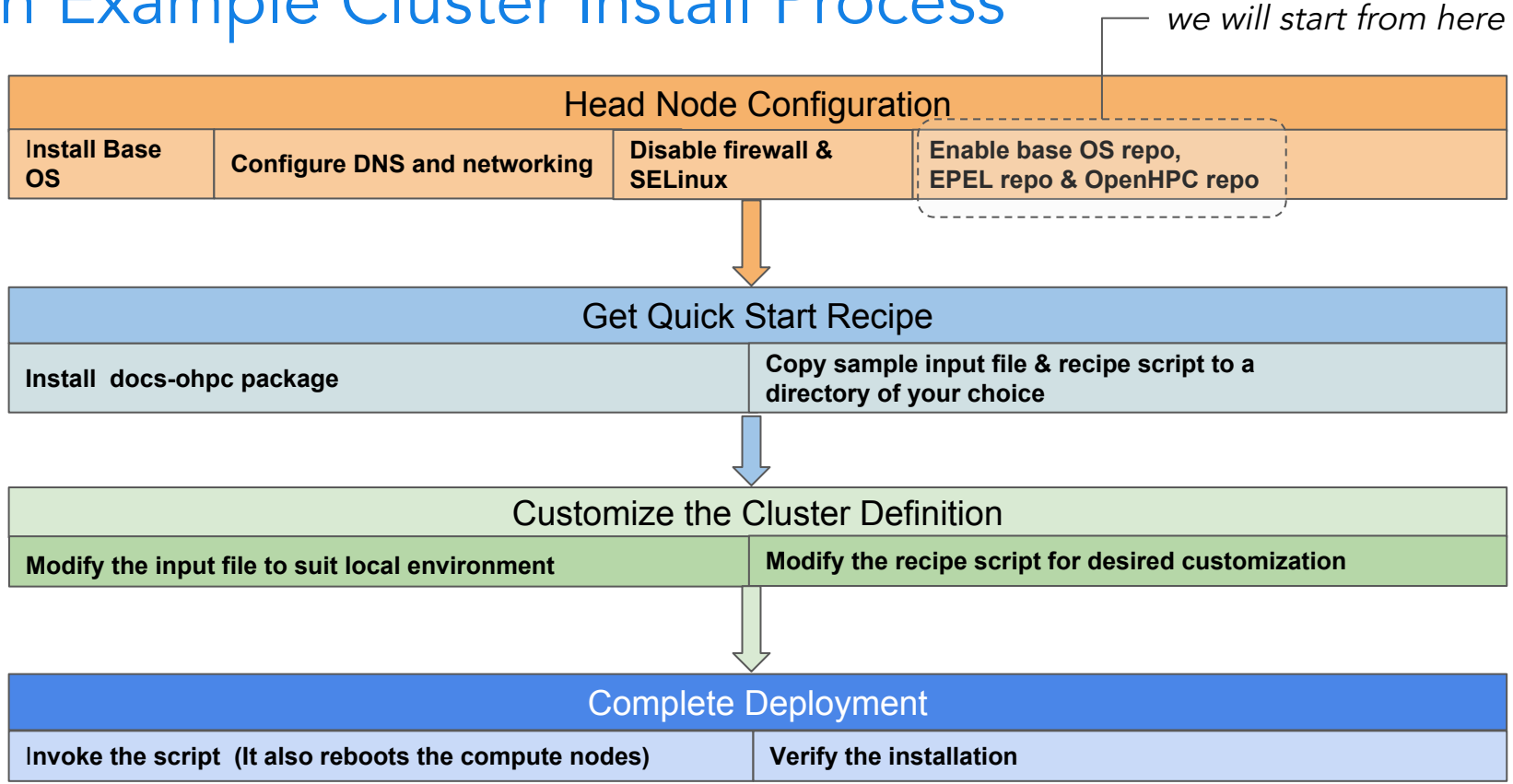
- Cluster hardware overview
- Divvy up into groups
  - each group will install a small cluster
- Run some MPI bandwidth tests
- Modify existing compute image and redeploy
- Rebuild development library from .src.rpm
  - install rebuilt RPM
  - install integration test suite and run for relevant package
- Additional items (time permitting)

Team Groupings and Selections

- Let's break up into 4-5 teams and choose high-level config

	Account	Assigned SMS host	Provisioner	RMS
Team 1	train01	sms001	Warewulf	SLURM
Team 2	train02	sms004	Warewulf	PBSPro
Team 3	train03	sms007	Warewulf	SLURM
Team 4	train04	sms010	xCAT	SLURM
Team 5	train05	sms013		
Instructors		sms020	Warewulf	SLURM

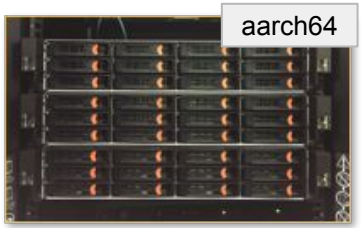
An Example Cluster Install Process



Cluster Hardware

- We will leverage some of the infrastructure hosted at TACC
- Using for build servers and continuous integration (CI) testbed.
- We will use the x86 portions to create 5 clusters
  - each team will be assigned a head node (SMS) + 2 computes
  - account names are train01,train02,...train05
- First, let's establish login access to the CI head node (this will be our access point to the assigned cluster H/W):

```
$ ssh foo@bar
```



Enable OpenHPC repository (CentOS)

1. Login to your assigned SMS node
   a. take note of the BMC user passwd (each cluster is different)
2. Escalate privileges
3. Install ohpc-release RPM from public repo
4. Verify you can query the repo

1
```
[train01@ohpc2 ~]$ ssh sms001
Cluster On Demand: SMS born-on date = 07/07/2017 13:33:02
Cluster On Demand: Base OS          = centos7.3
Cluster On Demand: Interactive      = true
Cluster On Demand: Queue name       = Install Cluster
Cluster On Demand: BUILD number     = 10134
--
Cluster On Demand: Local inputs      = /root/ci_ohpc_inputs
Cluster On Demand: Long test enable? = false
Cluster On Demand: BMC user/passwd   = cod/NWRlZGIxMWQ0

Cluster On Demand: Runlimit         = 480 (minutes)
Cluster On Demand: User             = train01
```

2
```
[train01@sms001 ~]$ sudo su -
```

3
```
[root@sms001 ~]# yum install
http://build.openhpc.community/OpenHPC:/1.3/CentOS_7/x86_64/ohpc-release-1.3-1.el7.x86_64.rpm
...
Installed:
  ohpc-release.x86_64 0:1.3-1.el7
```

4
```
[root@sms001 ~]# yum search ohpc
```

Install documentation and template recipes (CentOS)

1. Install docs-ohpc RPM
2. cp desired recipe.sh to local dir (recall path to available recipes)
   a. spend a few minutes perusing recipe.sh
   b. note a number "enable_*" variables to control optional installation/configuration
   c. high-level header comments map back to specific sections in the docs
3. Edit /root/ci_ohpc_inputs
   a. change bmc_username from "root" to "cod"
   b. change bmc_password from "unknown" to value shown in /etc/motd

1
```
[root@sms001 ~]# yum install docs-ohpc

[root@sms001 ~]# ls /opt/ohpc/pub/doc/recipes/centos7/x86_64/*/*/recipe.sh
/opt/ohpc/pub/doc/recipes/centos7/x86_64/warewulf/pbspro/recipe.sh
/opt/ohpc/pub/doc/recipes/centos7/x86_64/warewulf/slurm/recipe.sh
/opt/ohpc/pub/doc/recipes/centos7/x86_64/xcat/slurm/recipe.sh
```

2
```
[root@sms001 ~]# cp
/opt/ohpc/pub/doc/recipes/centos7/x86_64/warewulf/slurm/recipe.sh .
```

← vi is available by default, feel free to install other available editors (e.g. emacs-nox, nano) if you prefer

Launch install script

1. Override default input file for template install script via OHPC_INPUT_LOCAL env variable
2. Do the deed, run ./recipe.sh
   a. consider saving output via "tee" and timing it
   b. if all is well, should take ~15 minutes to complete
   c. you may want to open up another ssh session to interact with while the template script is running
3. Verify compute nodes are booted

1
```
[root@sms001 ~]# export OHPC_INPUT_LOCAL=/root/ci_ohpc_inputs
```

2
```
[root@sms001 ~]# time ./recipe.sh 2>&1 | tee install.log
Resolving Dependencies
--> Running transaction check
---> Package ohpc-base.x86_64 0:1.3.1-38.1 will be installed
--> Processing Dependency: conman-ohpc for package: ohpc-base-1.3.1-38.1.x86_64
--> Processing Dependency: lmod-ohpc for package: ohpc-base-1.3.1-38.1.x86_64
--> Processing Dependency: examples-ohpc for package: ohpc-base-1.3.1-38.1.x86_64
...
  petsc-gnu7-openmpi-ohpc.x86_64 0:3.7.6-26.1
  scalapack-gnu7-openmpi-ohpc.x86_64 0:2.0.2-12.2
  superlu_dist-gnu7-openmpi-ohpc.x86_64 0:4.2-79.2
  trilinos-gnu7-openmpi-ohpc.x86_64 0:12.10.1-49.1

Complete!
Created symlink from /etc/systemd/system/multi-user.target.wants/slurmctld.service to
/usr/lib/systemd/system/slurmctld.service.
c3: ssh: Could not resolve hostname c3: Name or service not known
pdsh@sms007: c3: ssh exited with exit code 255
c4: ssh: Could not resolve hostname c4: Name or service not known
pdsh@sms007: c4: ssh exited with exit code 255
```

⬅ ok to ignore the errors accessing c3,c4 (we assume a minimum of 4 nodes for CI testing)

3
```
[root@sms001 ~]# pdsh -w c[1-2] uptime
c1  15:30:23 up 4 min,  0 users,  load average: 0.09, 0.22, 0.11
c2  15:30:19 up 4 min,  0 users,  load average: 0.05, 0.15, 0.08
```

A few words on provisioning whilst we wait…

- We are using Warewulf and xCAT today for stateless-based provisioning
  - images generally kept small and are installed into ramdisk on computes

```
[root@sms001 ~]# wwsh vnfs list
VNFS NAME            SIZE (M) CHROOT LOCATION
centos7.3            301.6    /opt/ohpc/admin/images/centos7.3
```

  - leveraging shared file system to expose the development environment
- Warewulf/xCAT leverage standard tools for remote installs:
  - manage DHCP config based on registering nodes/MAC addresses

```
[root@sms001 ~]# wwsh node list
NAME                 GROUPS               IPADDR               HWADDR
================================================================================
c1                   UNDEF                172.16.1.8,172.17.1.8 a4:bf:01:00:9c:89
c2                   UNDEF                172.16.1.9,172.17.1.9 a4:bf:01:00:0b:72
```

  - rely on PXE/tftpboot to push out node-specific images
  - xCAT can setup local DNS config, Warewulf tends to use static routing via /etc/hosts
  - images are maintained on master SMS host in a CHROOT; both have tools to assemble a compressed image for association with different node types (e.g. compute, login)
  - both can synchronize desired config files (we are using for user credentials, RMS config today)

System install times

What did we get?

	Provisioner	RMS	Install Time
Team 1	Warewulf	SLURM	13m 42s
Team 2	Warewulf	PBS	13m 42s
Team 3	Warewulf	SLURM	13m 32s
Team 4	xCAT	SLURM	14m 38s
Team 5			
Instructors	Warewulf	SLURM	15m

Resource manager test job (slurm)

1. Verify queue is open for business
2. Become "test" user that was included in the installation process
3. Copy example hello world source code and compile
4. Request interactive job on 2 compute nodes (4 mpi tasks total)
5. Launch MPI binary on assigned compute(s) using "prun" convenience script

```
[root@sms001 ~]# sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
normal*      up 1-00:00:00      2   idle c[1-2]

[root@sms001 ~]# su - test
[test@sms001 ~]$ module list

Currently Loaded Modules:
  1) autotools   2) prun/1.1   3) gnu7/7.1.0   4) mvapich2/2.2   5) ohpc

[test@sms001 ~]$ cp /opt/ohpc/pub/examples/mpi/hello.c .
[test@sms001 ~]$ mpicc hello.c

[test@sms001 ~]$ srun -N 2 -n 4 --pty /bin/bash
[test@c1 ~]$ prun ./a.out
[prun] Master compute host = c1
[prun] Resource manager = slurm
[prun] Launch cmd = mpiexec.hydra -bootstrap slurm ./a.out

 Hello, world (4 procs total)
    --> Process #   0 of   4 is alive. -> c1
    --> Process #   1 of   4 is alive. -> c1
    --> Process #   2 of   4 is alive. -> c2
    --> Process #   3 of   4 is alive. -> c2
```

Resource manager test job (PBSPro)

1. Set up PBSPro environment
2. Verify queue is open for business
3. Become "test" user that was included in the installation process
4. Copy example hello world source code and compile
5. Request interactive job on 2 compute nodes
6. Launch MPI binary on assigned compute(s) using "prun" convenience script

```
[root@sms001 ~]# . /etc/profile.d/pbs.sh

[root@sms001 ~]# pbsnodes -aSj
                                                          mem       ncpus   nmics   ngpus
vnode           state           njobs   run   susp      f/t        f/t     f/t     f/t   jobs
--------------- --------------- ------ ----- ------ ------------ ------- ------- ------- -------
c1              free                 0     0      0    63gb/63gb   88/88     0/0     0/0 --
c2              free                 0     0      0    63gb/63gb   88/88     0/0     0/0 --

[root@sms001 ~]# su - test
[test@sms001 ~]$ module list

Currently Loaded Modules:
  1) autotools   2) prun/1.1   3) gnu7/7.1.0   4) mvapich2/2.2   5) ohpc

[test@sms001 ~]$ cp /opt/ohpc/pub/examples/mpi/hello.c .
[test@sms001 ~]$ mpicc hello.c

[test@sms001 ~]$ qsub -I -l select=2:mpiprocs=2
[test@c1 ~]$ prun ./a.out
[prun] Master compute host = c1
[prun] Resource manager = pbspro
[prun] Launch cmd = mpiexec.hydra -rmk pbspro ./a.out

 Hello, world (2 procs total)
    --> Process #   0 of   2 is alive. -> c1
    --> Process #   1 of   2 is alive. -> c2
```

Point-to-point MPI measurements (slurm example)

In this exercise, we will run some quick MPI tests using the IMB benchmark suite which is available for each of the installed MPI variants (mpich, mvapich2, and openmpi)

This are run under the "test" user

1. enable IMB for default MPI stack (mvapich2)
2. Request interactive job on 2 compute nodes (2 mpi tasks total this time)
3. Run PingPong test
   a. log the **max** bandwidth
4. Swap to another MPI stack and repeat the test
   a. note the imb module being reloaded automatically
   b. log the max bandwidth again
5. Repeat the process for mpich and log the max bandwidth

1
```
[test@sms001 ~]$ module load imb
```

2
```
[test@sms001 ~]$ srun -N 2 -n 2 --pty /bin/bash
```

3
```
[test@c1 ~]$ prun IMB-MPI1 PingPong
...
#---------------------------------------------------
# Benchmarking PingPong
# #processes = 2
#---------------------------------------------------
       #bytes #repetitions      t[usec]   Mbytes/sec
            0         1000         1.58         0.00
            1         1000         1.68         0.57
            2         1000         1.64         1.16
            4         1000         1.59         2.40
            8         1000         1.59         4.80
...
```

4
```
[test@c1 ~]$ module swap mvapich2 openmpi
[test@c1 ~]$ prun IMB-MPI1 PingPong
```

Point-to-point MPI measurements (results)

What did we get?

	Max Bandwidth (Mbytes/sec)		
	mvapich2	openmpi	mpich
Team 1	5998	5869	110.89
Team 2	5999	5873	110.89
Team 3	5864	5890	110.88
Team 4	5872	6001	110.94
Team 5			
Instructors	11472.39	11465.72	110.88

Point-to-point MPI measurements

Table 1: Available MPI builds

	Ethernet (TCP)	InfiniBand	Intel® Omni-Path
MPICH	✓		
MVAPICH2		✓	
MVAPICH2 (psm2)			✓
OpenMPI	✓	✓	
OpenMPI (psm2)	✓	✓	✓

Modify compute node image and redeploy (Warewulf)

In this exercise, we will work thru augmenting the existing compute image by adding an additional package (gnuplot).

1. Install desired RPM into CHROOT
2. Rebuild VNFS image
3. Reboot compute nodes to deploy
   a. we source the "ci_ohpc_inputs" file to leverage BMC IP addresses
   b. verify that gnuplot is indeed available after compute node reboot

1
```
[root@sms001 ~]# yum --installroot=/opt/ohpc/admin/images/centos7.3 install gnuplot
```

2
```
[root@sms001 ~]# wwvnfs --chroot /opt/ohpc/admin/images/centos7.3
Using 'centos7.3' as the VNFS name
Creating VNFS image from centos7.3
Compiling hybridization link tree                          : 0.08 s
Building file list                                         : 0.34 s
Compiling and compressing VNFS                             : 4.96 s
Adding image to datastore                                  : 9.32 s
Total elapsed time                                         : 14.71 s
```

3
```
[root@sms001 ~]# . ci_ohpc_inputs
[root@sms001 ~]# ipmitool -E -I lanplus -H ${c_bmc[0]} -U cod chassis power reset
Chassis Power Control: Reset
[root@sms001 ~]# ipmitool -E -I lanplus -H ${c_bmc[1]} -U cod chassis power reset
Chassis Power Control: Reset

[root@sms001 ~]# ssh c1 gnuplot -V
gnuplot 4.6 patchlevel 2
```

Modify compute node image and redeploy (xCAT)

1. Install desired RPM in to CHROOT (1)
2. Rebuild VNFS image (2)
3. Reboot compute nodes to deploy (3)
   a. we source the "ci_ohpc_inputs" file to leverage BMC IP addresses
   b. verify that gnuplot is indeed available after compute node reboot

```
[root@sms001 ~]# . /etc/profile.d/xcat.sh
[root@sms001 ~]# yum --installroot=/install/netboot/centos7.3/x86_64/compute/rootimg install gnuplot

[root@sms001 ~]# packimage centos7.3-x86_64-netboot-compute
Packing contents of /install/netboot/centos7.3/x86_64/compute/rootimg
archive method:cpio
compress method:gzip

[root@sms001 ~]# . ci_ohpc_inputs
[root@sms001 ~]# ipmitool -E -I lanplus -H ${c_bmc[0]} -U cod chassis power reset
Chassis Power Control: Reset
[root@sms001 ~]# ipmitool -E -I lanplus -H ${c_bmc[1]} -U cod chassis power reset
Chassis Power Control: Reset

[root@sms001 ~]# ssh c1 gnuplot -V
gnuplot 4.6 patchlevel 2
```

Rebuilding from source RPM

In this exercise, we will highlight building a component (FFTW) from src and upgrading the local install:

1. Install rpmbuild and OpenHPC build macros
2. Install desired source RPM
3. Make functional changes to spec file
4. Bump RPM release number
5. Rebuild
6. Install new binary RPM

```
1 [train01@sms001 ~]$ sudo yum install rpm-build ohpc-buildroot
  ...
2 [train01@sms001 ~]$ rpm -i
  http://build.openhpc.community/OpenHPC:/1.3:/Update1/CentOS_7/src/fftw-gnu7-openmpi-ohpc-3.3.6
  -20.3.src.rpm

  [train01@sms001 ~]$ cd ~/rpmbuild/SPECS

3 [train01@sms001 ~]$ perl -pi -e "s/enable-static=no/enable-avx512=yes/" fftw.spec

4 [train01@sms001 ~]$ perl -pi -e "s/Release:   20.3/Release:   21.3/" fftw.spec

5 [train01@sms001 ~]$ rpmbuild -bb fftw.spec

6 [train01@sms001 ~]$ sudo yum install
  ~/rpmbuild/RPMS/x86_64/fftw-gnu7-openmpi-ohpc-3.3.6-21.3.x86_64.rpm
```

Install and launch integration tests

Next, we will install OpenHPC's test suite and run the suite for the FFTW component we just reinstalled.

1. Install OpenHPC test suite RPM
2. Sync ohpc-test account to compute nodes
3. Become ohpc-test user
4. Configure fftw test
5. Execute test suite

1
```
[train01@sms001 ~]$ sudo yum install test-suite-ohpc
...
```
2
```
[train01@sms001 ~]$ sudo wwsh file resync passwd shadow group
[train01@sms001 ~]$ sudo pdsh -w c[1-2] /warewulf/bin/wwgetfiles
```
3
```
[train01@sms001 ~]$ sudo su - ohpc-test
```
4
```
[ohpc-test@sms001 ~]$ cd ~/tests/libs/fftw
[ohpc-test@sms001 ~]$ ./bootstrap
[ohpc-test@sms001 ~]$ module load fftw
[ohpc-test@sms001 ~]$ ./configure
```
5
```
[ohpc-test@sms001 ~]$ make check
...
PASS: C_test
PASS: C_mpi_test
PASS: C_threads_test
PASS: F_test
PASS: CXX_omp_test
PASS: rm_execution
============================================================================
```

Performance analysis with TAU

In this exercise, we briefly exercise one of the installed performance analysis tools (TAU) that is available thru OpenHPC.

1. Load TAU module
2. Compile source code using TAU wrapper script and set desired profiling environment variables
3. Launch instrumented binary with tau_exec

1
```
[test@sms001 ~]$ module load tau
```

2
```
[test@sms001 ~]$ tau_cxx.sh ~ohpc-test/tests/perf-tools/scalasca/tests/CXX_mpi_test.cpp
[test@sms001 ~]$ export TAU_TRACE=0 TAU_CALLPATH=0 TAU_PROFILE=1
[test@sms001 ~]$ export TAU_METRICS=GET_TIME_OF_DAY:PAPI_L2_DCM
```

3
```
[test@sms001 ~]$ srun -N 2 -n 2 --pty /bin/bash
[test@c1 ~]$ prun tau_exec ./a.out
```

Performance analysis with TAU (Cont.)

View timing profile data

```
train01@c1 ~]$ cd MULTI__GET_TIME_OF_DAY
train01@c1 ~]$ pprof
...

FUNCTION SUMMARY (mean):
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0        0.272          273         1           1     273156 .TAU application
 99.9        0.496          272         1           6     272885 int main(int, char **)
 87.7          239          239         1           0     239678 MPI_Init()
 10.8           13           29         1        5611      29630 void update(int, int)
  3.5            9            9      4400           0          2 double rhs(double, double)
  1.7            4            4       400           0         11 MPI_Recv()
  1.1            2            2         1           0       2968 MPI_Finalize()
  0.5            1            1       400           0          3 MPI_Send()
  0.3        0.912        0.912       400           0          2 double boundary_condition(double, double)
  0.0        0.102        0.102         1           0        102 void timestamp()
  0.0       0.0265       0.0265        11           0          2 double initial_condition(double, double)
  0.0       0.0055       0.0055         1           0          6 MPI_Comm_rank()
  0.0       0.0035       0.0035         1           0          4 MPI_Comm_size()
```

Performance analysis with TAU (Cont.)

View PAPI counter profile data

```
train01@c1 ~]$ cd MULTI__PAPI_L2_DCM/
train01@c1 ~]$ pprof
...

FUNCTION SUMMARY (mean):
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs Count/Call Name
           counts total counts
---------------------------------------------------------------------------------------
100.0        1041   3.314E+05           1           1     331412 .TAU application
 99.7        3726   3.304E+05           1           6     330372 int main(int, char **)
 86.9   2.881E+05   2.881E+05           1           0     288058 MPI_Init()
  7.0        3797   2.321E+04           1        5611      23210 void update(int, int)
  4.4    1.46E+04    1.46E+04           1           0      14600 MPI_Finalize()
  3.1   1.014E+04   1.014E+04         400           0         25 MPI_Send()
  2.7        8783        8783         400           0         22 MPI_Recv()
  0.2       760.5       760.5           1           0        760 void timestamp()
  0.1       292.5       292.5         400           0          1 double boundary_condition(double, double)
  0.1         177         177        4400           0          0 double rhs(double, double)
  0.0        25.5        25.5          11           0          2 double initial_condition(double, double)
  0.0          13          13           1           0         13 MPI_Comm_rank()
  0.0         5.5         5.5           1           0          6 MPI_Comm_size()
```

Wahoo!! You made it.

Backup Slides

Spack compiler registration and module setup

In this exercise, we will use one of the add-on package managers (spack) to build an IO monitoring library (Darshan).

1. As root, load desired compiler family and spack module files
2. Spack will search the path for new compilers
3. Install desired software from pre-packaged spack recipe
4. Source spack environment script to set new module path

```
[root@sms001 ~]# module load gcc7 spack

[root@sms001 ~]# spack compiler find
==> Added 1 new compiler to /root/.spack/linux/compilers.yaml    gcc@7.1.0

[root@sms001 ~]# spack install darshan-runtime
...
[root@sms001 ~]# . /opt/ohpc/admin/spack/0.10.0/share/spack/setup-env.sh

[root@sms001 ~]# module avail
------------ /opt/ohpc/admin/spack/0.10.0/share/spack/modules/linux-sles12-x86_64 ------------
   darshan-runtime-3.1.0-gcc-7.1.0-vhd5hhg    m4-1.4.17-gcc-7.1.0-7jd575i
   hwloc-1.11.4-gcc-7.1.0-u3k6dok             ncurses-6.0-gcc-7.1.0-l3mdumo
   libelf-0.8.13-gcc-7.1.0-tsgwr7j            openmpi-2.0.1-gcc-7.1.0-5imqlfb
   libpciaccess-0.13.4-gcc-7.1.0-33gbduz      pkg-config-0.29.1-gcc-7.1.0-dhbpa2i
   libsigsegv-2.10-gcc-7.1.0-lj5rntg          util-macros-1.19.0-gcc-7.1.0-vkdpa3t
   libtool-2.4.6-gcc-7.1.0-ulicbkz            zlib-1.2.10-gcc-7.1.0-gy4dtna


--------------------------- /opt/ohpc/pub/moduledeps/gnu7-mvapich2 ---------------------------
   adios/1.11.0    imb/4.1         netcdf-fortran/4.4.4    scalapack/2.0.2    sionlib/1.7.1
   boost/1.63.0    mpiP/3.4.1      netcdf/4.4.1.1          scalasca/2.3.1     superlu_dist/4.2
```

Install and launch integration tests

Next, we will install OpenHPC's test suite and run the suite for the FFTW component we just reinstalled.

1. Install OpenHPC test suite RPM
2. Migrate ohpc-test user credentials to compute nodes
3. Become ohpc-test user
4. Configure test run as desired
5. Execute test suite

```
1 [train01@sms001 ~]$ sudo yum install test-suite-ohpc
  ...
2 [train01@sms001 ~]$ sudo wwsh file resync passwd shadow group
  [train01@sms001 ~]$ sudo pdsh -w c[1-2] /warewulf/bin/wwgetfiles

3 [train01@sms001 ~]$ sudo su - ohpc-test

4 [train01@sms001 ~]$ cd ~/tests
  [train01@sms001 ~]$ ./configure --disable-all --enable-fftw

5 [train01@sms001 ~]$ make check
make --no-print-directory check-TESTS
PASS: libs/fftw/ohpc-tests/test_mpi_families
============================================================================
Testsuite summary for test-suite 1.3.0
============================================================================
# TOTAL: 1
# PASS:  1
# SKIP:  0
# XFAIL: 0
# FAIL:  0
# XPASS: 0
# ERROR: 0
============================================================================
```