openHPC
http://openhpc.community

OpenHPC: Community Building Blocks for HPC Systems

Karl W. Schulz, Ph.D.
Scalable Datacenter Solutions Group, Intel

4th Annual MVAPICH User Group (MUG) Meeting
August 17, 2016 ◆ Columbus, Ohio

Outline

- Motivation and brief history for this new community effort
- Overview of project governance
- Stack overview
- What’s coming

Motivation for Community Effort

- Many sites spend considerable effort aggregating a large suite of open-source projects to provide a capable HPC environment for their users:
  - necessary to build/deploy HPC focused packages that are either absent or do not keep pace from Linux distro providers
  - local packaging or customization frequently tries to give software versioning access to users (e.g. via modules or similar equivalent)

- OpenHPC is focused on establishing a centralizing community effort to:
  - provide a collection of pre-packaged binary components that can be used to help install and manage HPC systems throughout their lifecycle
  - implement integration testing to gain validation confidence
  - allow and promote multiple system configuration recipes that leverage community reference designs and best practices
  - provide additional distribution/integration mechanisms for leading research groups releasing open-source software
  - foster identification and development of relevant interfaces between supported components that allows for simple component replacement and customization

OpenHPC: a brief History…

- ISC'15 (June 2015) – BoF discussion on the merits/interest in a *Community Supported HPC Repository and Management Framework*
  - discussed convenience of distribution via standard Linux package managers
  - feedback at the time was that most parties were interested in CentOS/SLES
  - consensus that "modules" or equivalent needed to provide capable end-user development environment
- MUG'15 (August 2015) - shamelessly squeezed in a sidebar about potential interest in a community supported HPC repo
- SC'15 (November 2015) – Follow on BoF for a *Comprehensive Open Community HPC Software Stack*
  - initial seeding of OpenHPC and a 1.0 release
  - variety of interested community members assembled thru the Linux Foundation to work towards establishing a formal, collaborative project
- Nov'15 – May'16
  - Linux Foundation working group collaborating to define participating agreement, initial governance structure and solicit volunteers

➡ *June 16, 2016 – Linux Foundation announces technical, leadership and member investment milestones with founding members and formal governance structure*

OpenHPC: Project Members

Argonne National Laboratory

Altair

ARM

Atos

Avtech Scientific
attaining the vision

BSC
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

UNIVERSITY OF CAMBRIDGE

CEA

CINECA

CRAY

DELL

FUJITSU

GENCI

Hewlett Packard Enterprise

intel

BERKELEY LAB

INDIANA UNIVERSITY BLOOMINGTON

Lawrence Livermore National Laboratory

lrz

Lenovo

Los Alamos
NATIONAL LABORATORY
EST. 1943

PARTEC
CLUSTER COMPETENCE CENTER

PSC
PITTSBURGH SUPERCOMPUTING CENTER

RIKEN

sgi

SUSE

TACC

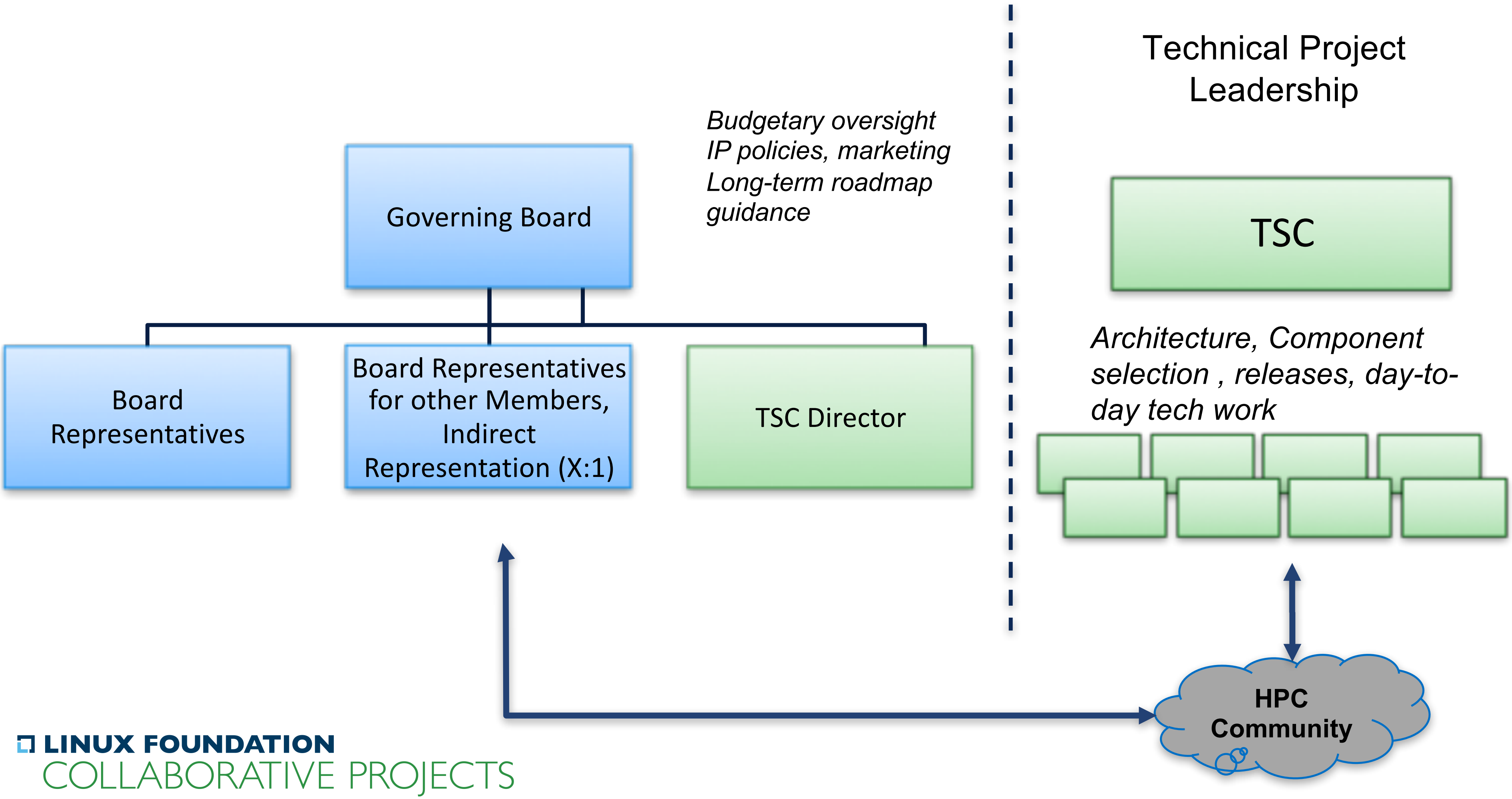
UNIVA

Mixture of Academics, Labs, OEMs, and ISVs/OSVs

Project member participation interest? Please contact Jeff ErnstFriedman jernstfriedman@linuxfoundation.org

Community Governance Overview

Governing Board + Technical Steering Committee



Governing Board

Budgetary oversight
IP policies, marketing
Long-term roadmap guidance

Board Representatives

Board Representatives for other Members, Indirect Representation (X:1)

TSC Director

Technical Project Leadership

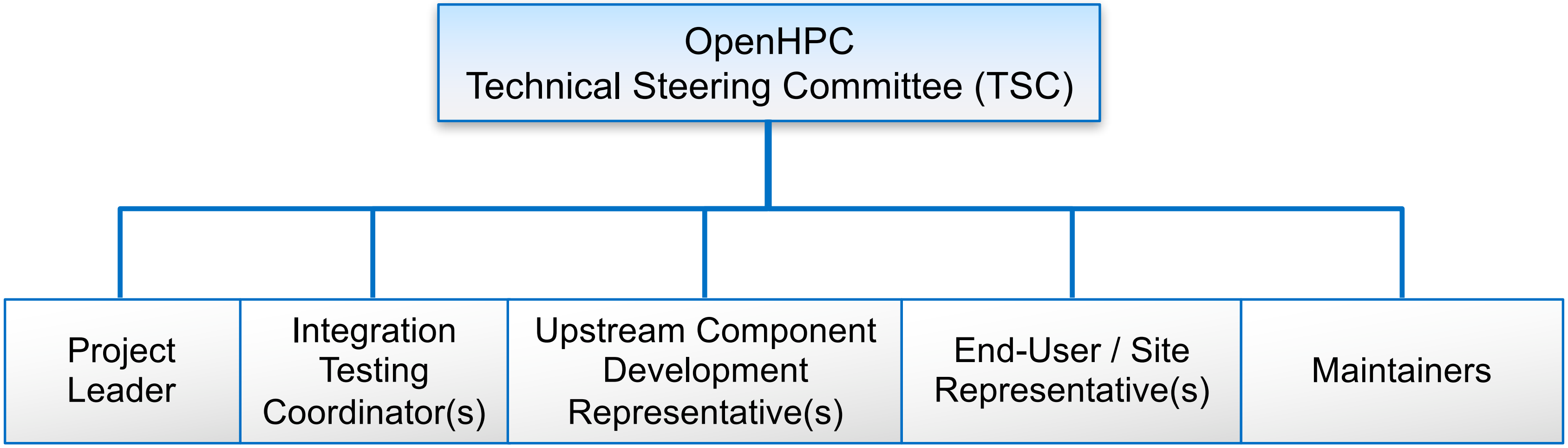
TSC

Architecture, Component selection , releases, day-to-day tech work

HPC Community

LINUX FOUNDATION
COLLABORATIVE PROJECTS

OpenHPC TSC - Role Overview



OpenHPC
Technical Steering Committee (TSC)

- Project Leader
- Integration Testing Coordinator(s)
- Upstream Component Development Representative(s)
- End-User / Site Representative(s)
- Maintainers

LINUX FOUNDATION
COLLABORATIVE PROJECTS

OpenHPC TSC – Individual Members

- Reese Baird, Intel (Maintainer)
- Pavan Balaji, Argonne National Laboratory (Maintainer)
- David Brayford, LRZ (Maintainer)
- Todd Gamblin, Lawrence Livermore National Labs (Maintainer)
- Craig Gardner, SUSE (Maintainer)
- Yiannis Georgiou, ATOS (Maintainer)
- Balazs Gerofi, RIKEN (Component Development Representative)
- Jennifer Green, Los Alamos National Laboratory (Maintainer)
- Eric Van Hensbergen, ARM (Maintainer, Testing Coordinator)
- Douglas Jacobsen, NERSC (End-User/Site Representative)
- Chulho Kim, Lenovo (Maintainer)
- Greg Kurtzer, Lawrence Berkeley National Labs (Component Development Representative)
- Thomas Moschny, ParTec (Maintainer)
- Karl W. Schulz, Intel (Project Lead, Testing Coordinator)
- Derek Simmel, Pittsburgh Supercomputing Center (End-User/Site Representative)
- Thomas Sterling, Indiana University (Component Development Representative)
- Craig Stewart, Indiana University (End-User/Site Representative)
- Scott Suchyta, Altair (Maintainer)
- Nirmala Sundararajan, Dell (Maintainer)

https://github.com/openhpc/ohpc/wiki/Governance-Overview

Stack Overview

- Packaging efforts have HPC in mind and include compatible modules (for use with Lmod) with development libraries/tools
- Endeavoring to provide hierarchical development environment that is cognizant of different compiler and MPI families
- Intent is to manage package dependencies so they can be used as building blocks (e.g. deployable with multiple provisioning systems)
- Include common conventions for env variables
- Development library install example:

```
# yum install petsc-gnu-mvapich2-ohpc
```

- End user interaction example with above install: (assume we are a user wanting to build a PETSC hello world in C)

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

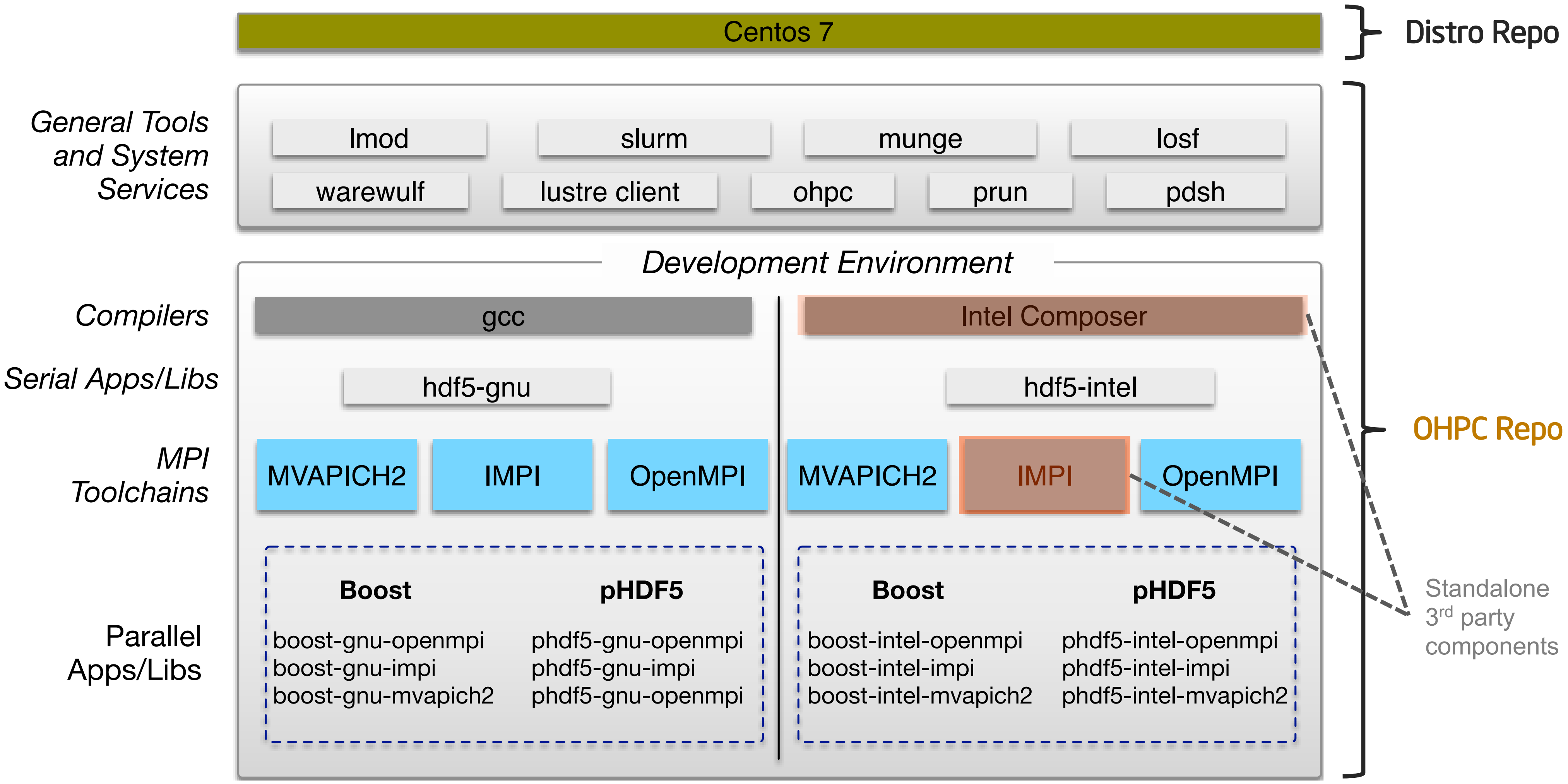
OpenHPC 1.1.1 - Current S/W components

Functional Areas	Components
Base OS	CentOS 7.2, SLES12 SP1
Administrative Tools	Conman, Ganglia, Lmod, LosF, Nagios, pdsh, prun, EasyBuild, ClusterShell, mrsh, Genders, Shine, Spack
Provisioning	Warewulf
Resource Mgmt.	SLURM, Munge
Runtimes	OpenMP, OCR
I/O Services	Lustre client (community version)
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
Development Tools	Autotools (autoconf, automake, libtool), Valgrind,R, SciPy/NumPy
Performance Tools	PAPI, IMB, mpiP, pdtoolkit TAU

Notes:

- Additional dependencies that are not provided by the BaseOS or community repos (e.g. EPEL) are also included
- 3rd Party libraries are built for each compiler/MPI family (6 combinations typically)
- Resulting repositories currently comprised of ~270 RPMs

Hierarchical Overlay for OpenHPC software



Centos 7

Distro Repo

General Tools and System Services

lmod · slurm · munge · losf

warewulf · lustre client · ohpc · prun · pdsh

Development Environment

Compilers

gcc · Intel Composer

Serial Apps/Libs

hdf5-gnu · hdf5-intel

MPI Toolchains

MVAPICH2 · IMPI · OpenMPI · MVAPICH2 · IMPI · OpenMPI

Parallel Apps/Libs

Boost

boost-gnu-openmpi
boost-gnu-impi
boost-gnu-mvapich2

pHDF5

phdf5-gnu-openmpi
phdf5-gnu-impi
phdf5-gnu-openmpi

Boost

boost-intel-openmpi
boost-intel-impi
boost-intel-mvapich2

pHDF5

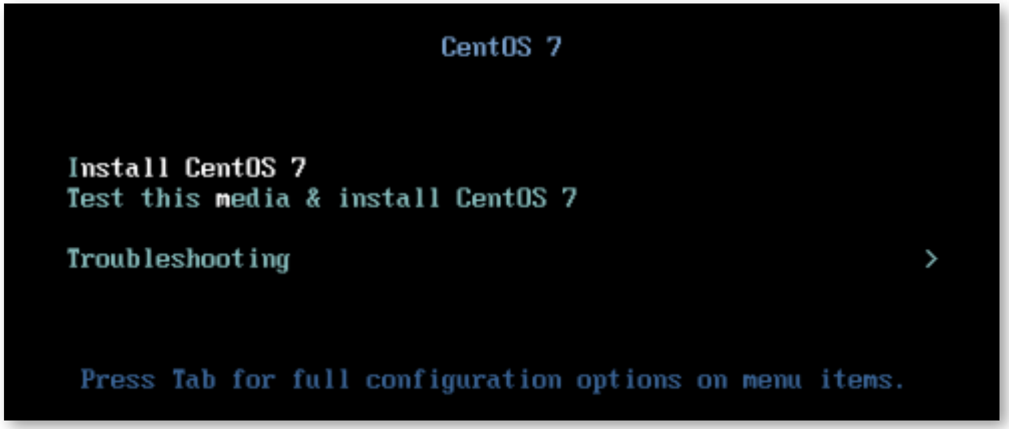
phdf5-intel-openmpi
phdf5-intel-impi
phdf5-intel-mvapich2

OHPC Repo

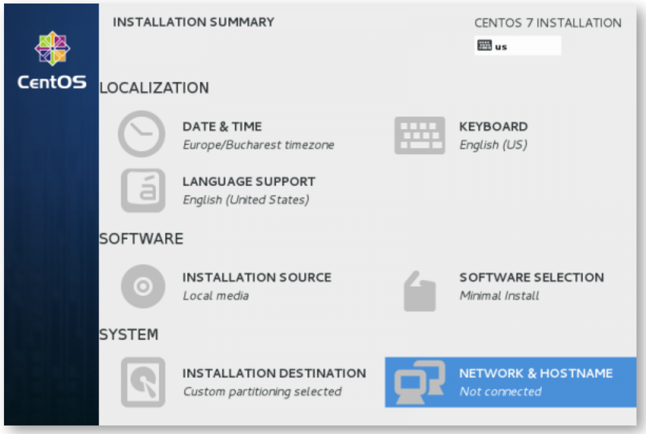
Standalone 3rd party components

Stack Overview: Bare metal install

- Step1: Example OpenHPC 1.1 recipe assumes base OS is first installed on chosen master (SMS) host - e.g. install CentOS7.2 on SMS



```
CentOS 7

Install CentOS 7
Test this media & install CentOS 7

Troubleshooting                          >

Press Tab for full configuration options on menu items.
```



INSTALLATION SUMMARY

CENTOS 7 INSTALLATION

us

LOCALIZATION

DATE & TIME
Europe/Bucharest timezone

KEYBOARD
English (US)

LANGUAGE SUPPORT
English (United States)

SOFTWARE

INSTALLATION SOURCE
Local media

SOFTWARE SELECTION
Minimal Install

SYSTEM

INSTALLATION DESTINATION
Custom partitioning selected

NETWORK & HOSTNAME
Not connected

- Step2: Enable OpenHPC repo using pre-packaged ohpc-release (or mirror repo locally)

```
# export OHPC_GITHUB=https://github.com/openhpc/ohpc/releases/download
# rpm -ivh ${OHPC_GITHUB}/v1.1.GA/ohpc-release-1.1-1.x86_64.rpm
```

Stack Overview: Bare metal install (cont.)

- Note that ohpc-release enables two repos:

```
# yum repolist
repo id                          repo name
OpenHPC                          OpenHPC-1.1 - Base
OpenHPC-updates                  OpenHPC-1.1 - Updates
base                             CentOS-7 - Base
epel                             Extra Packages for Enterprise Linux 7 - x86_64
```

- Step3: install desired building blocks to build cluster or add development tools. Convenience aliases are provided to group related functionality

```
[sms]# yum -y groupinstall ohpc-base
[sms]# yum -y groupinstall ohpc-warewulf
```

Add provisioning components

```
[sms]# yum -y groupinstall ohpc-slurm-server
```

Add SLURM server components

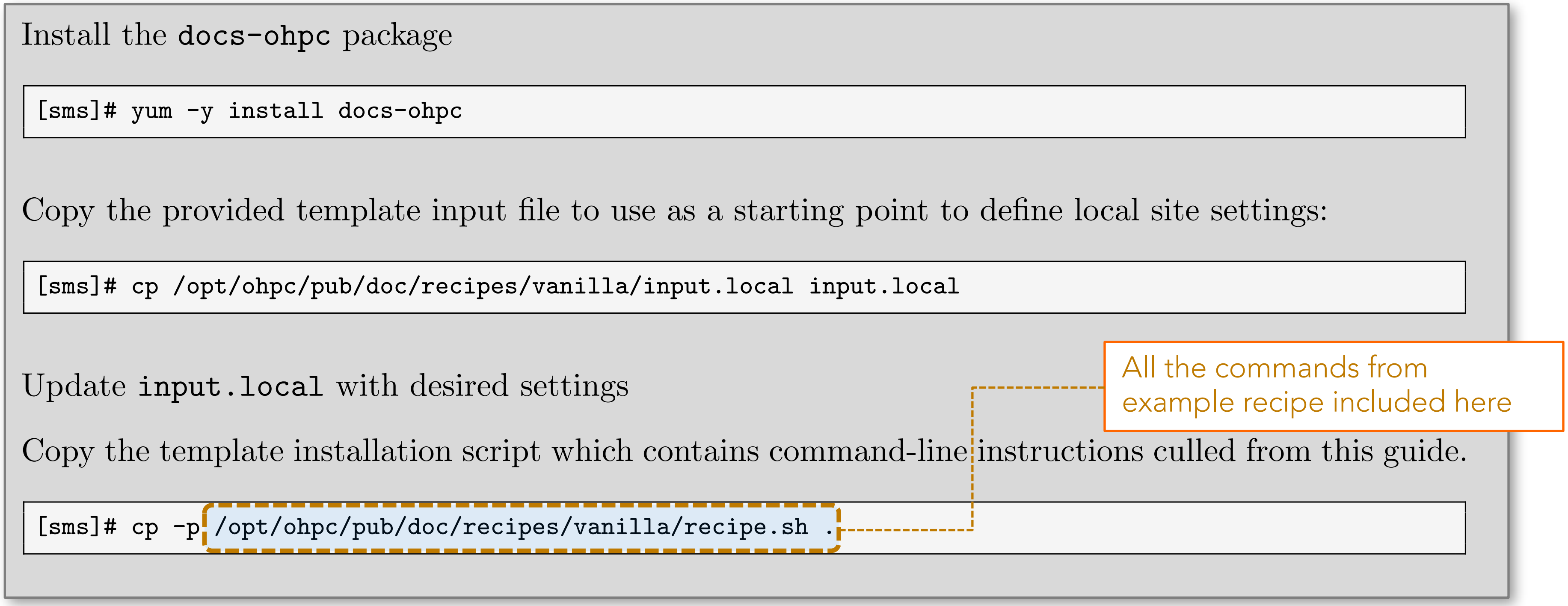
```
[sms]# cp /opt/ohpc/pub/examples/network/centos/ifcfg-ib0 /etc/sysconfig/network-scripts

# Define local IPoIB address and netmask
[sms]# perl -pi -e "s/master_ipoib/${sms_ipoib}/" /etc/sysconfig/network-scripts/ifcfg-ib0
[sms]# perl -pi -e "s/ipoib_netmask/${ipoib_netmask}/" /etc/sysconfig/network-scripts/ifcfg-ib0

# Initiate ib0
[sms]# ifup ib0
```

Example optional configuration enabling IPoIB (e.g. to support Lustre over IB)

*note that community recipe is purposefully very transparent on config file edits and assumes Linux familiarity

Stack Overview: Bare metal install (cont.)

- Recipe guides necessarily have a number of things to “cut-and-paste” if you want to reproduce them
- Have motivating need to automate during the validation process:
  - Cull out relevant commands automatically for use during CI testing
  - Seemed reasonable to make available directly, so there is a template starting script available with the documentation RPM which can be used for local installation and customization



Install the **docs-ohpc** package

```
[sms]# yum -y install docs-ohpc
```

Copy the provided template input file to use as a starting point to define local site settings:

```
[sms]# cp /opt/ohpc/pub/doc/recipes/vanilla/input.local input.local
```

Update **input.local** with desired settings

Copy the template installation script which contains command-line instructions culled from this guide.

```
[sms]# cp -p /opt/ohpc/pub/doc/recipes/vanilla/recipe.sh .
```

All the commands from example recipe included here

Development Infrastructure

OpenHPC Development Infrastructure

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

The usual software engineering stuff:

- GitHub (SCM and issue tracking/planning)
- Continuous Integration (CI) Testing (Jenkins)
- Documentation (Latex)

Capable build/packaging system

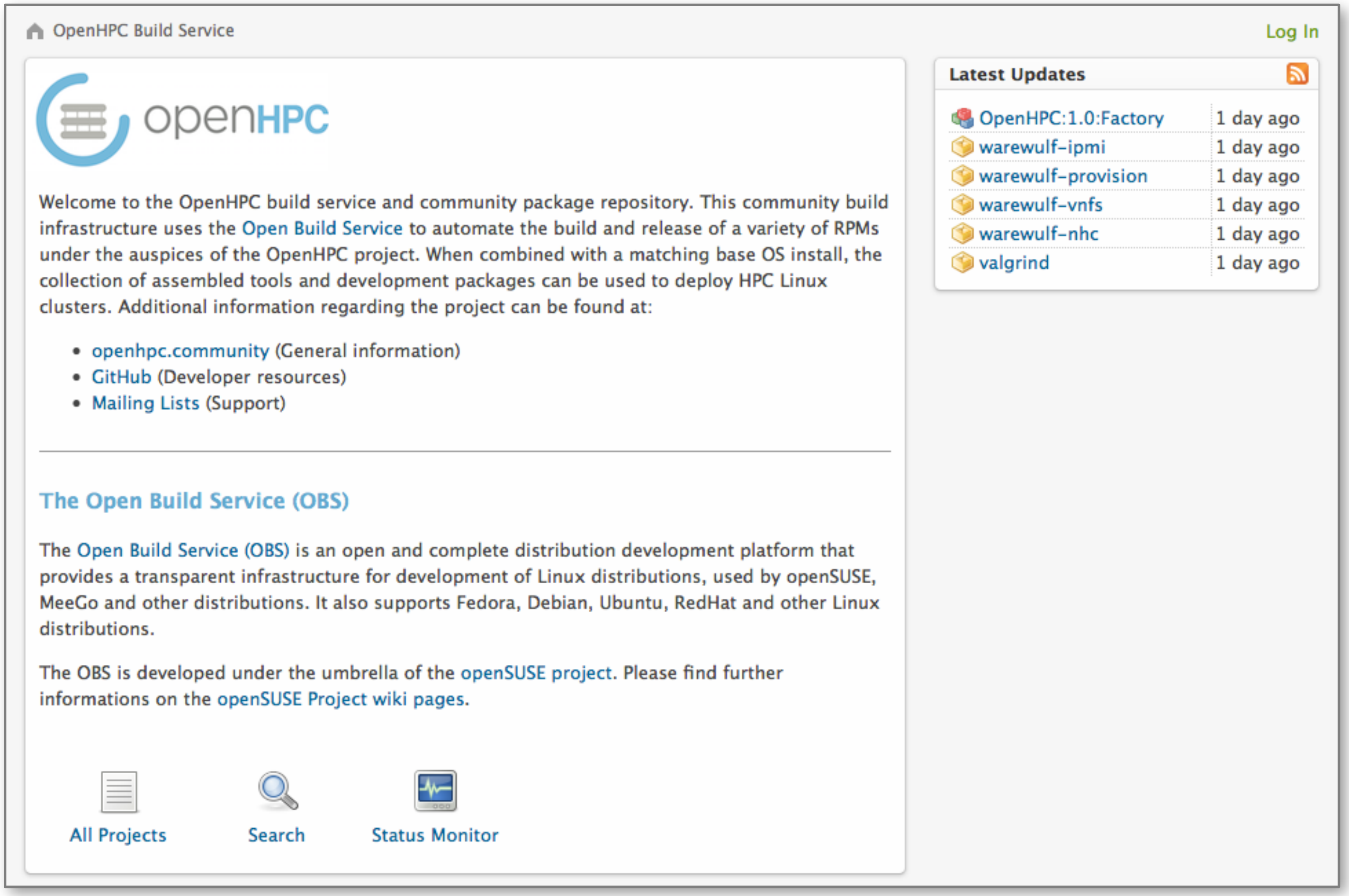
- At present, we target a common delivery/access mechanism that adopts Linux sysadmin familiarity - ie. **yum/zypper** repositories for supported distros
  - ultimately delivering RPMs
  - [base] + [update] repositories to support life-cycle management
- Require flexible system to manage builds for multiple 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

https://github.com/openhpc/ohpc

https://build.openhpc.community

Build System - OBS

https://build.openhpc.community



Welcome to the OpenHPC build service and community package repository. This community build infrastructure uses the Open Build Service to automate the build and release of a variety of RPMs under the auspices of the OpenHPC project. When combined with a matching base OS install, the collection of assembled tools and development packages can be used to deploy HPC Linux clusters. Additional information regarding the project can be found at:

- openhpc.community (General information)
- GitHub (Developer resources)
- Mailing Lists (Support)

The Open Build Service (OBS)

The Open Build Service (OBS) is an open and complete distribution development platform that provides a transparent infrastructure for development of Linux distributions, used by openSUSE, MeeGo and other distributions. It also supports Fedora, Debian, Ubuntu, RedHat and other Linux distributions.

The OBS is developed under the umbrella of the openSUSE project. Please find further informations on the openSUSE Project wiki pages.

Latest Updates	
OpenHPC:1.0:Factory	1 day ago
warewulf-ipmi	1 day ago
warewulf-provision	1 day ago
warewulf-vnfs	1 day ago
warewulf-nhc	1 day ago
valgrind	1 day ago

- Using the **Open Build Service** (OBS) to manage build process
- OBS can drive builds for multiple repositories
- 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

OpenHPC Build Architecture Conventions

- Motivation is to have single input to drive multiple output configurations (e.g. hierarchy for compiler/MPI families)
- Also want to establish baseline install path conventions
- Leverage variety of macros to aid in this effort

Install Path Macros

```
$ cat OHPC_macros
%define OHPC_BUILD 1
%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
```

MPI macros

```
$ cat OHPC_setup_mpi
if [ -z "$OHPC_MPI_FAMILY" ]; then
    echo "Unknown OHPC_MPI_FAMILY"
    exit 1
fi

if [ "$OHPC_MPI_FAMILY" = "openmpi" ]; then
    module load openmpi
elif [ "$OHPC_MPI_FAMILY" = "impi" ]; then
    module load impi
elif [ "$OHPC_MPI_FAMILY" = "mvapich2" ]; then
    module load mvapich2
else
    echo "Unsupported OHPC_MPI_FAMILY -> $OHPC_MPI_FAMILY"
    exit 1
fi
```

/opt/ohpc <--- *Top-level path convention for installs*

OpenHPC Build Architecture Conventions (cont.)

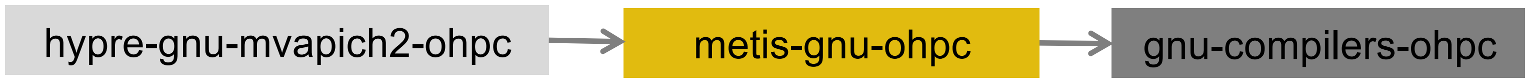
Example of compiler hierarchy template

- Default family choice defined, but can be overridden
- Family dependencies embedded for package managers

Snippet from METIS .spec file

```
%include %{_sourcedir}/OHPC_macros

# OpenHPC convention: the default assumes the gnu compiler family;
# however, this can be overridden by specifying the compiler_family
# variable via rpmbuild or other mechanisms.

%{!?compiler_family: %define compiler_family gnu}

# Compiler dependencies
BuildRequires: lmod%{PROJ_DELIM}
%if %{compiler_family} == gnu
BuildRequires: gnu-compilers%{PROJ_DELIM}
Requires:      gnu-compilers%{PROJ_DELIM}
%endif
%if %{compiler_family} == intel
BuildRequires: gcc-c++ intel-compilers-devel%{PROJ_DELIM}
Requires:      gcc-c++ intel-compilers-devel%{PROJ_DELIM}
%endif
```

yum/zypper install:



hypre-gnu-mvapich2-ohpc → metis-gnu-ohpc → gnu-compilers-ohpc

Build System - OBS

- OBS manages dependency resolution and rebuilds all downstream packages
- Leveraging ability within OBS to link related packages
  - Convenient for packages with **compiler** and **MPI** dependencies
  - Single commit drives all package permutations
- OBS builds automatically triggered via git commit hooks

Snippets from METIS OBS config

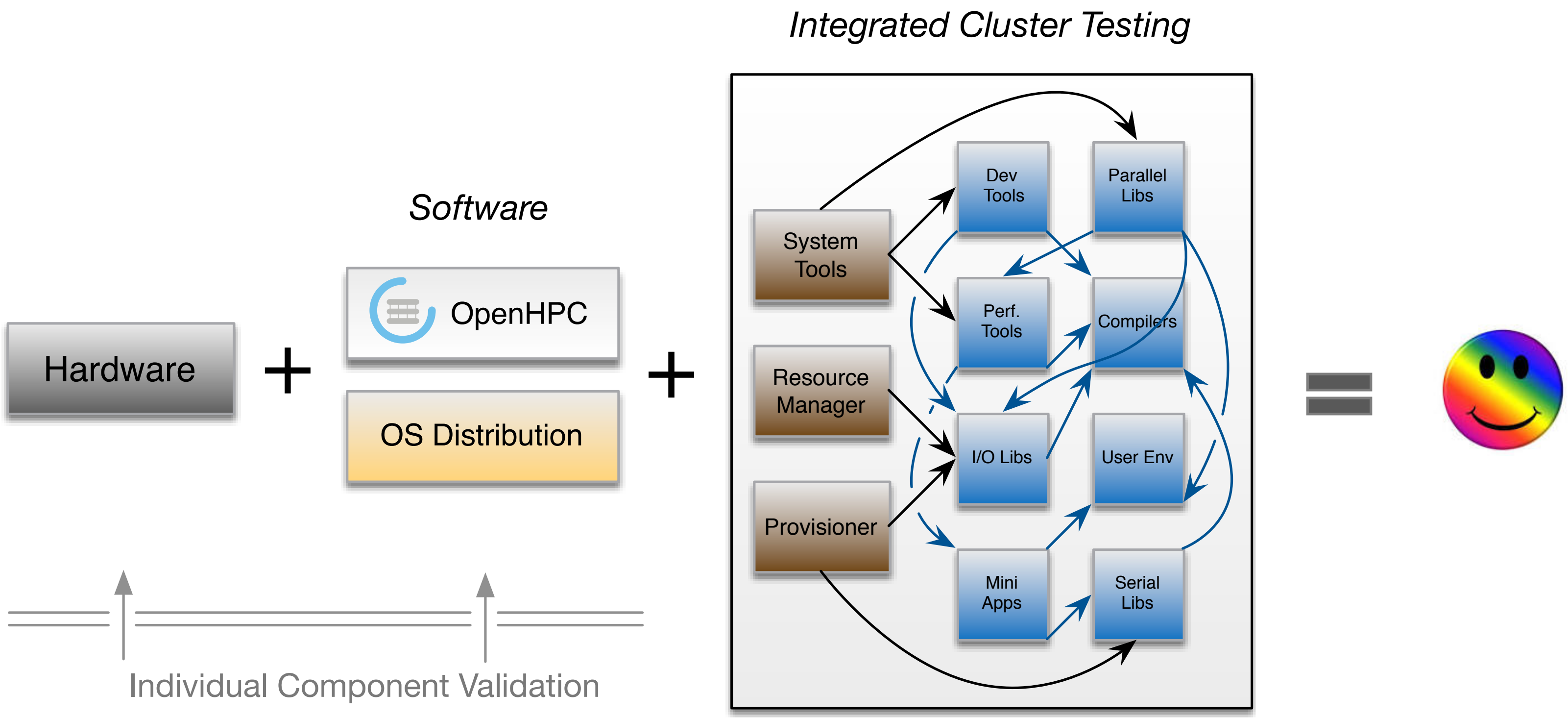
```
$ ls metis-*
metis-gnu:
_service          <-- Parent

metis-intel:      <-- Child
_link

$ cat metis-intel/_link
<link project='OpenHPC:1.1:Factory' package='metis-gnu'>
<patches>
  <topadd>%define compiler_family intel</topadd>
</patches>
</link>
```

Integration Testing

Integration/Test/Validation

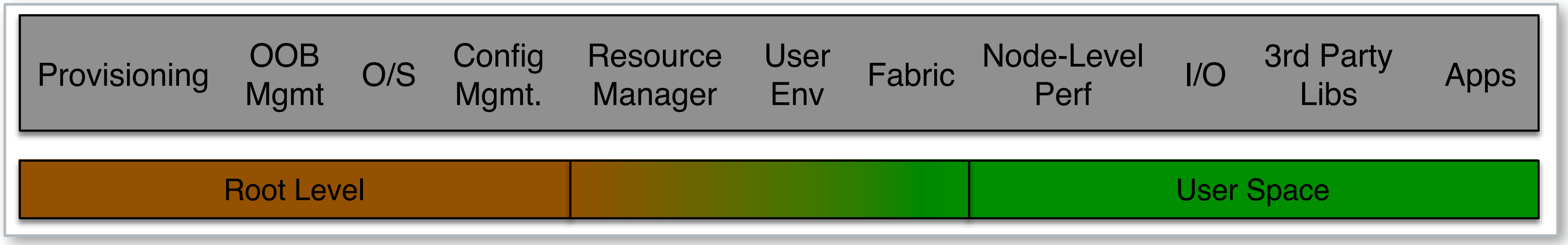
Testing is a key element for us and the intent is to build upon existing validation efforts and augment component-level validation with targeted cluster-validation and scaling initiatives including:

- install recipes
- cross-package interaction
- development environment
- mimic use cases common in HPC deployments
- upgrade mechanism



Integration/Test/Validation

- To facilitate global efforts in diagnostics/validation, we have devised a standalone integration test infrastructure
- Intent was to create families of tests that could be used during:
  - initial install process (can we build a system?)
  - post-install process (does it work?)
  - developing tests that touch all of the major components (can we compile against 3rd party libraries, will they execute under resource manager, etc)
- Expectation is that each new component included will need corresponding integration test collateral
- These integration tests and harness are included in GitHub repo



Provisioning | OOB Mgmt | O/S | Config Mgmt. | Resource Manager | User Env | Fabric | Node-Level Perf | I/O | 3rd Party Libs | Apps

Root Level | User Space

Post Install Integration Tests - Overview

- 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 (great opportunity for community participation)

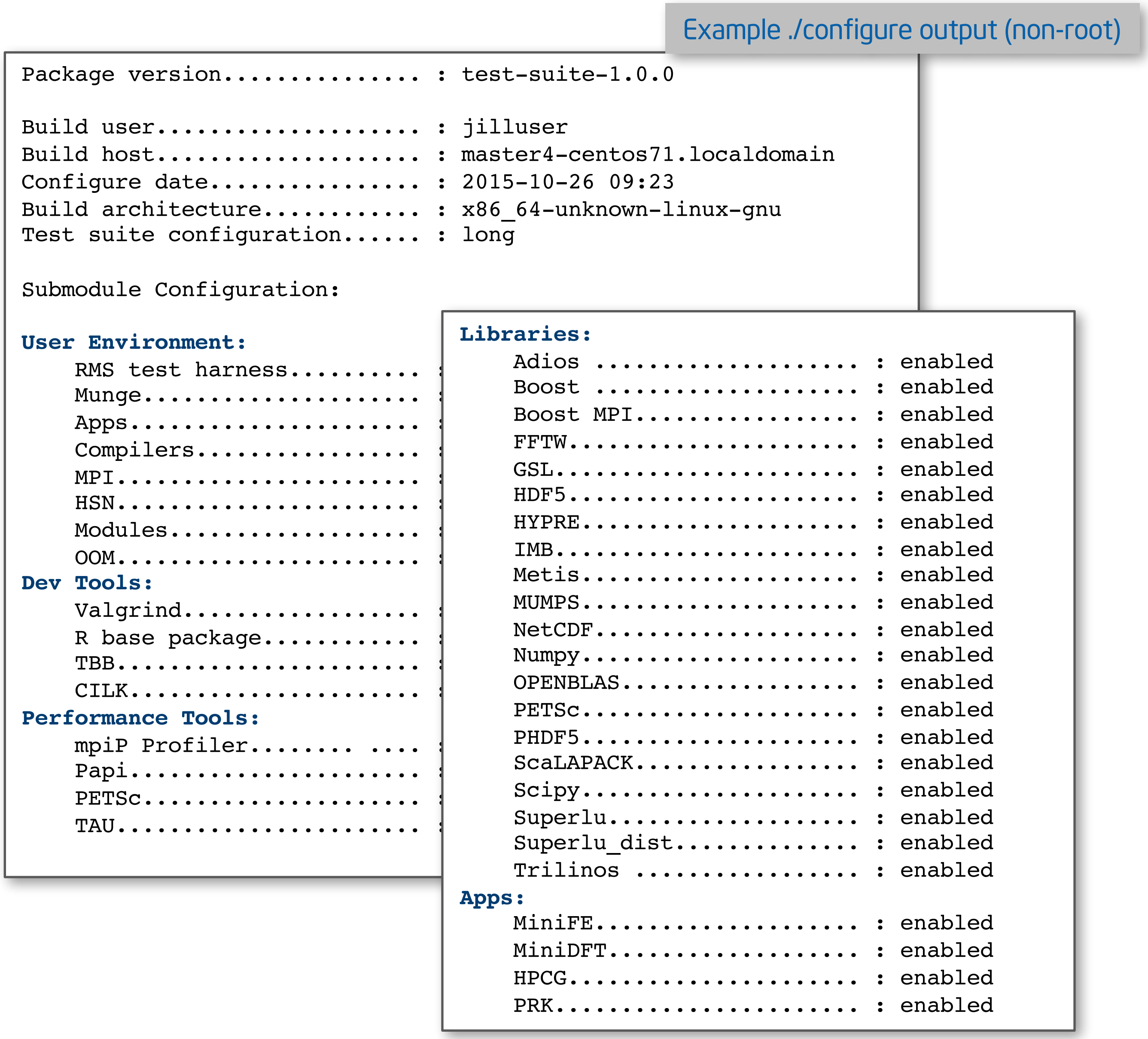
- Dev environment tests are a mixture of flavors:
  - interactive execution to verify certain binaries are in working order
  - successful compilation to test libraries provided via OpenHPC
  - successful interactive execution
  - tests for module usability and consistency
  - successful remote execution under resource manager

> 1,000 jobs submitted to RM as part of the current test suite

Post Install Integration Tests - Overview

Global testing harness includes a number of embedded subcomponents:

- major components have configuration options to enable/disable
- end user tests need to touch all of the supported compiler and MPI families
- we abstract this to repeat the tests with different compiler/MPI environments:
  - gcc/Intel compiler toolchains
  - Intel, OpenMPI, MVAPICH2 MPI families



Example ./configure output (non-root)

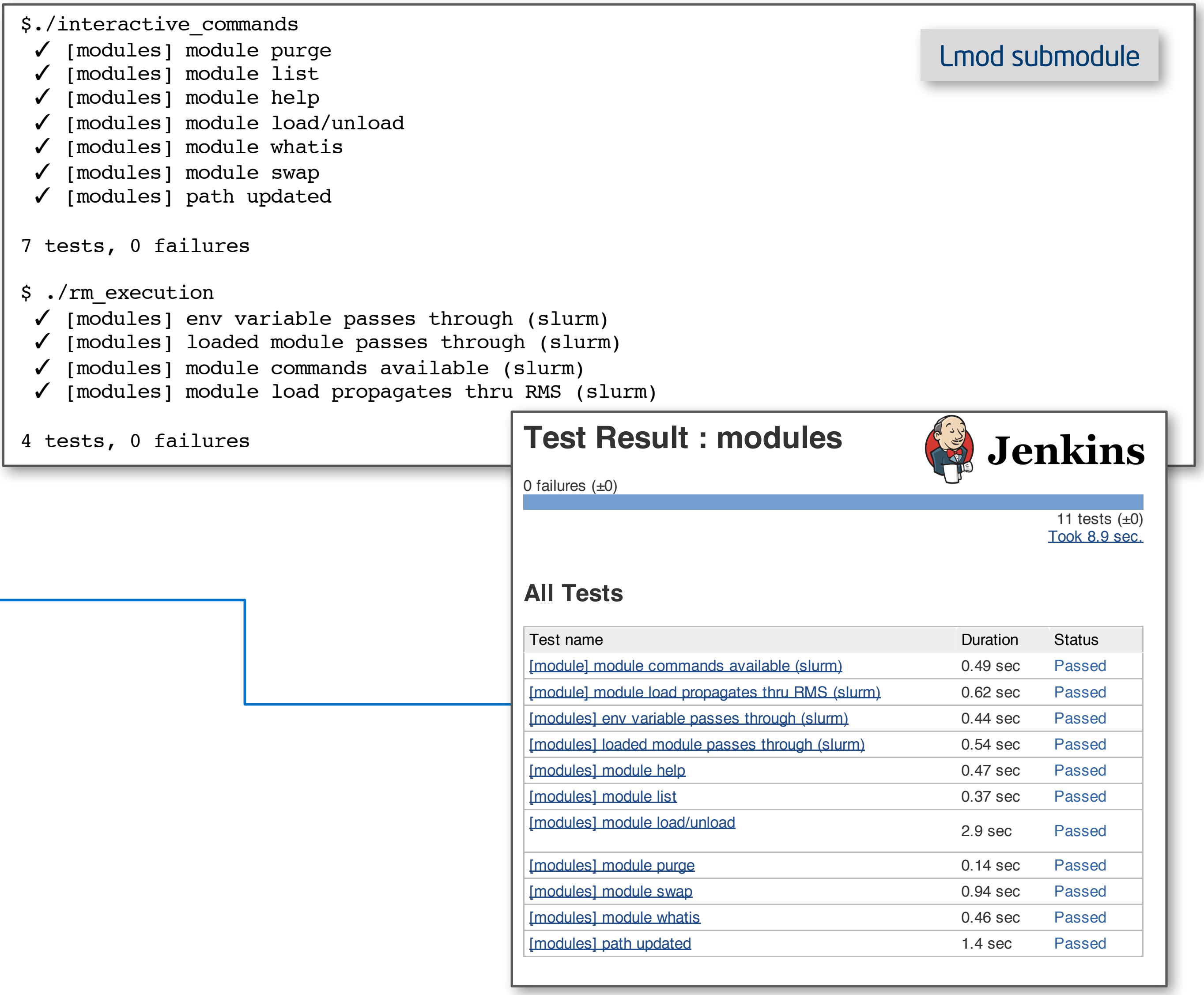
```
Package version............... : test-suite-1.0.0

Build user.................... : jilluser
Build host.................... : master4-centos71.localdomain
Configure date................ : 2015-10-26 09:23
Build architecture............ : x86_64-unknown-linux-gnu
Test suite configuration...... : long

Submodule Configuration:

User Environment:
    RMS test harness.......... :
    Munge..................... :
    Apps...................... :
    Compilers................. :
    MPI....................... :
    HSN....................... :
    Modules................... :
    OOM....................... :
Dev Tools:
    Valgrind.................. :
    R base package............ :
    TBB....................... :
    CILK...................... :
Performance Tools:
    mpiP Profiler........ .... :
    Papi...................... :
    PETSc..................... :
    TAU....................... :
```

```
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
    PRK....................... : enabled
```

Integration Tests - Let’s see one submodule test in action

Lmod user environment

- These are examples that primarily test interactive commands
- We are using the Bash Automated Testing System (Bats) for these tests
  - a TAP-complaint framework for Bash
  - available on GitHub
- We have extended Bats to:
  - create Junit output for parsing into Jenkins CI environment
  - capture execution runtimes



Lmod submodule

```
$./interactive_commands
 ✓ [modules] module purge
 ✓ [modules] module list
 ✓ [modules] module help
 ✓ [modules] module load/unload
 ✓ [modules] module whatis
 ✓ [modules] module swap
 ✓ [modules] path updated

7 tests, 0 failures

$ ./rm_execution
 ✓ [modules] env variable passes through (slurm)
 ✓ [modules] loaded module passes through (slurm)
 ✓ [modules] module commands available (slurm)
 ✓ [modules] module load propagates thru RMS (slurm)

4 tests, 0 failures
```

Test Result : modules

Jenkins

0 failures (±0)

11 tests (±0)
Took 8.9 sec.

All Tests

Test name	Duration	Status
[module] module commands available (slurm)	0.49 sec	Passed
[module] module load propagates thru RMS (slurm)	0.62 sec	Passed
[modules] env variable passes through (slurm)	0.44 sec	Passed
[modules] loaded module passes through (slurm)	0.54 sec	Passed
[modules] module help	0.47 sec	Passed
[modules] module list	0.37 sec	Passed
[modules] module load/unload	2.9 sec	Passed
[modules] module purge	0.14 sec	Passed
[modules] module swap	0.94 sec	Passed
[modules] module whatis	0.46 sec	Passed
[modules] path updated	1.4 sec	Passed

What’s Coming

- Some known big ticket items on the horizon for the TSC
  - establishing a process and prioritization/selection process for including new software components
  - establish minimum integration test expectations
  - establish packaging conventions:
    - naming schemes
    - dependency hierarchy management
    - installation paths
    - upgrade/rollback? mechanisms
  - roadmap timeline for next release (and cadence strategy for future releases)
  - addition of public CI infrastructure, roll out of additional architecture builds (e.g. ARM)

openHPC
http://openhpc.community

Thanks for your Time - Questions?

karl.w.schulz@intel.com

http://openhpc.community
https://github.com/openhpc/ohpc
https://build.openhpc.community (repo)