http://openhpc.community

Cluster Computing with OpenHPC

Karl W. Schulz, Ph.D.
Technical Project Lead, OpenHPC Community
Scalable Datacenter Solutions Group, Intel

HPCSYSPROS16 Workshop, SC'16
November 14 ◆ Salt Lake City, Utah

Acknowledgements

- Co-Authors: Reese Baird, David Brayford, Yiannis Georgiou, Gregory Kurtzer, Derek Simmel, Thomas Sterling, Nirmala Sundararajan, Eric Van Hensbergen
- OpenHPC Technical Steering Committee (TSC)
- Linux Foundation and all of the project members
- Intel, Cavium, and Dell for hardware donations to support community testing efforts
- Texas Advanced Computing Center for hosting support

Outline

- Community project overview
  - mission/vision
  - members
  - governance
- Stack overview
- Infrastructure: build/test
- Summary

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: Project Members

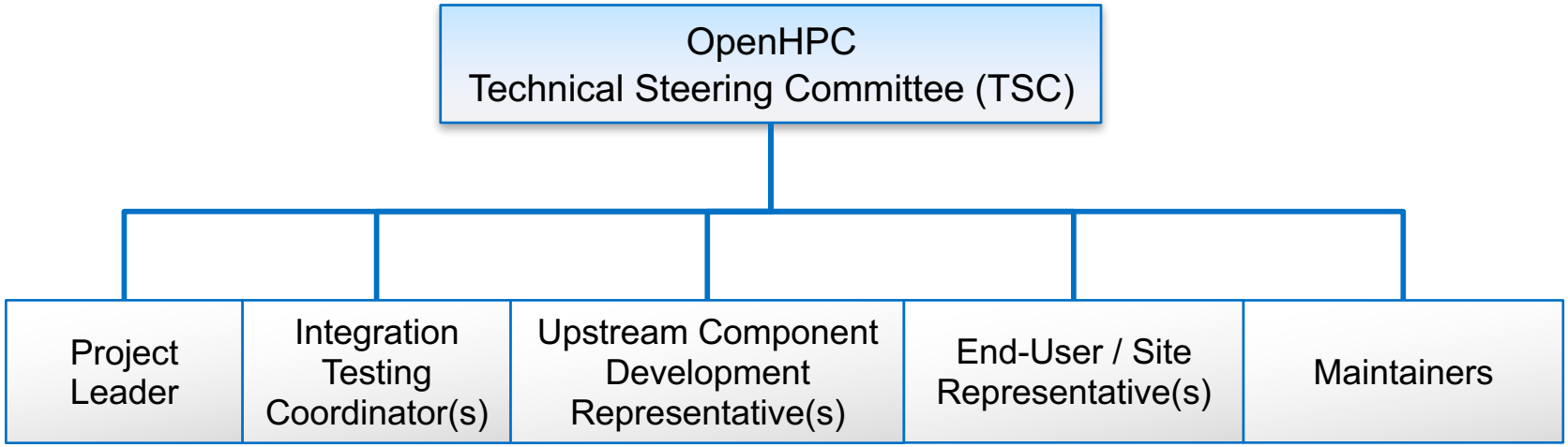
Argonne National Laboratory

CEA

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

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

OpenHPC Technical Steering Committee (TSC)
Role Overview



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

LINUX FOUNDATION
COLLABORATIVE PROJECTS

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
```

Typical Cluster Architecture

- Install guides walk thru bare-metal install
- Leverages image-based provisioner (Warewulf)
  - PXE boot (stateless)
  - optionally connect external Lustre file system
- Obviously need hardware-specific information to support (remote) bare-metal provisioning

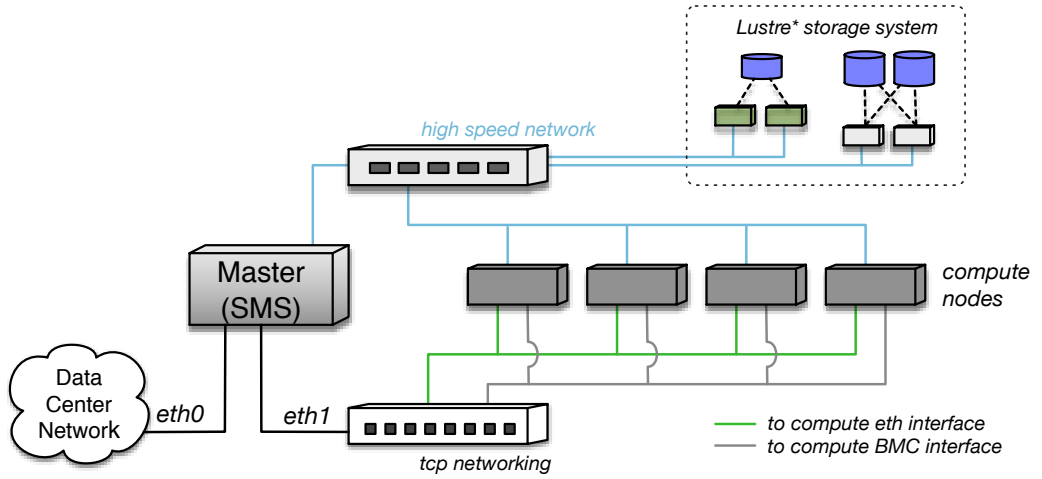


Figure 1: Overview of physical cluster architecture.

- `${sms_name}` # Hostname for SMS server
- `${sms_ip}` # Internal IP address on SMS server
- `${sms_eth_internal}` # Internal Ethernet interface on SMS
- `${eth_provision}` # Provisioning interface for computes
- `${internal_netmask}` # Subnet netmask for internal network
- `${ntp_server}` # Local ntp server for time synchronization
- `${bmc_username}` # BMC username for use by IPMI
- `${bmc_password}` # BMC password for use by IPMI
- `${c_ip[0]}, ${c_ip[1]}, ...` # Desired compute node addresses
- `${c_bmc[0]}, ${c_bmc[1]}, ...` # BMC addresses for computes
- `${c_mac[0]}, ${c_mac[1]}, ...` # MAC addresses for computes
- `${compute_regex}` # Regex for matching compute node names (e.g. c*)

Optional:

- `${mgs_fs_name}` # Lustre MGS mount name
- `${sms_ipoib}` # IPoIB address for SMS server
- `${ipoib_netmask}` # Subnet netmask for internal IPoIB
- `${c_ipoib[0]}, ${c_ipoib[1]}, ...` # IPoIB addresses for computes

OpenHPC v1.2 - Current S/W components

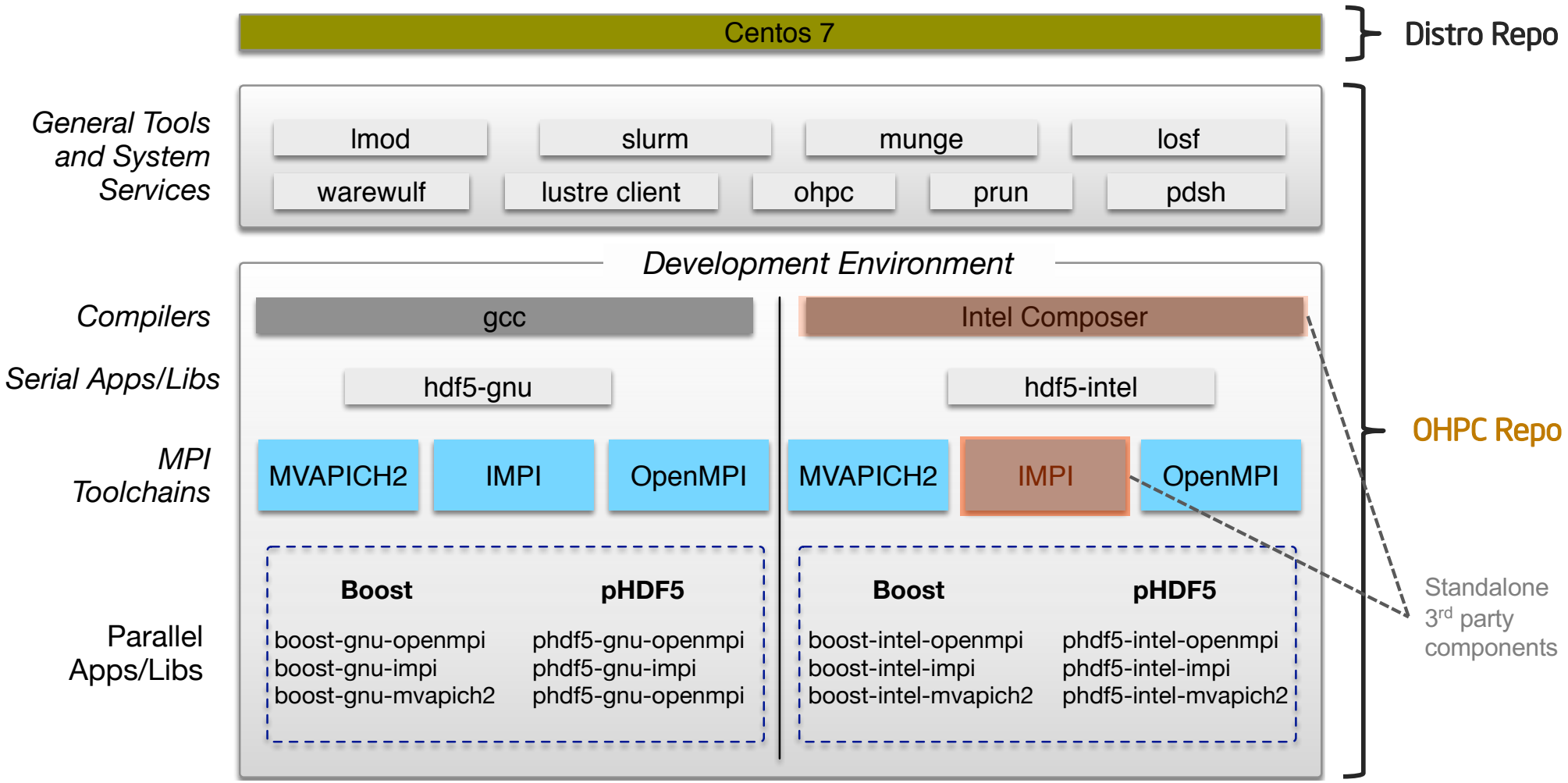
Functional Areas	Components
Base OS	CentOS 7.2, SLES12 SP1
Architecture	x86_64, aarch64 (Tech Preview)
Administrative Tools	Conman, Ganglia, Lmod, LosF, Nagios, pdsh, prun, EasyBuild, ClusterShell, mrsh, Genders, Shine, Spack
Provisioning	Warewulf
Resource Mgmt.	SLURM, Munge, PBS Professional
Runtimes	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, MPICH
Development Tools	Autotools (autoconf, automake, libtool), Valgrind,R, SciPy/NumPy
Performance Tools	PAPI, IMB, mpiP, pdtoolkit TAU, Scalasca, ScoreP, SIONLib

new with v1.2

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 (8 combinations typically)
- Resulting repositories currently comprised of ~300 RPMs

Hierarchical Overlay for OpenHPC software

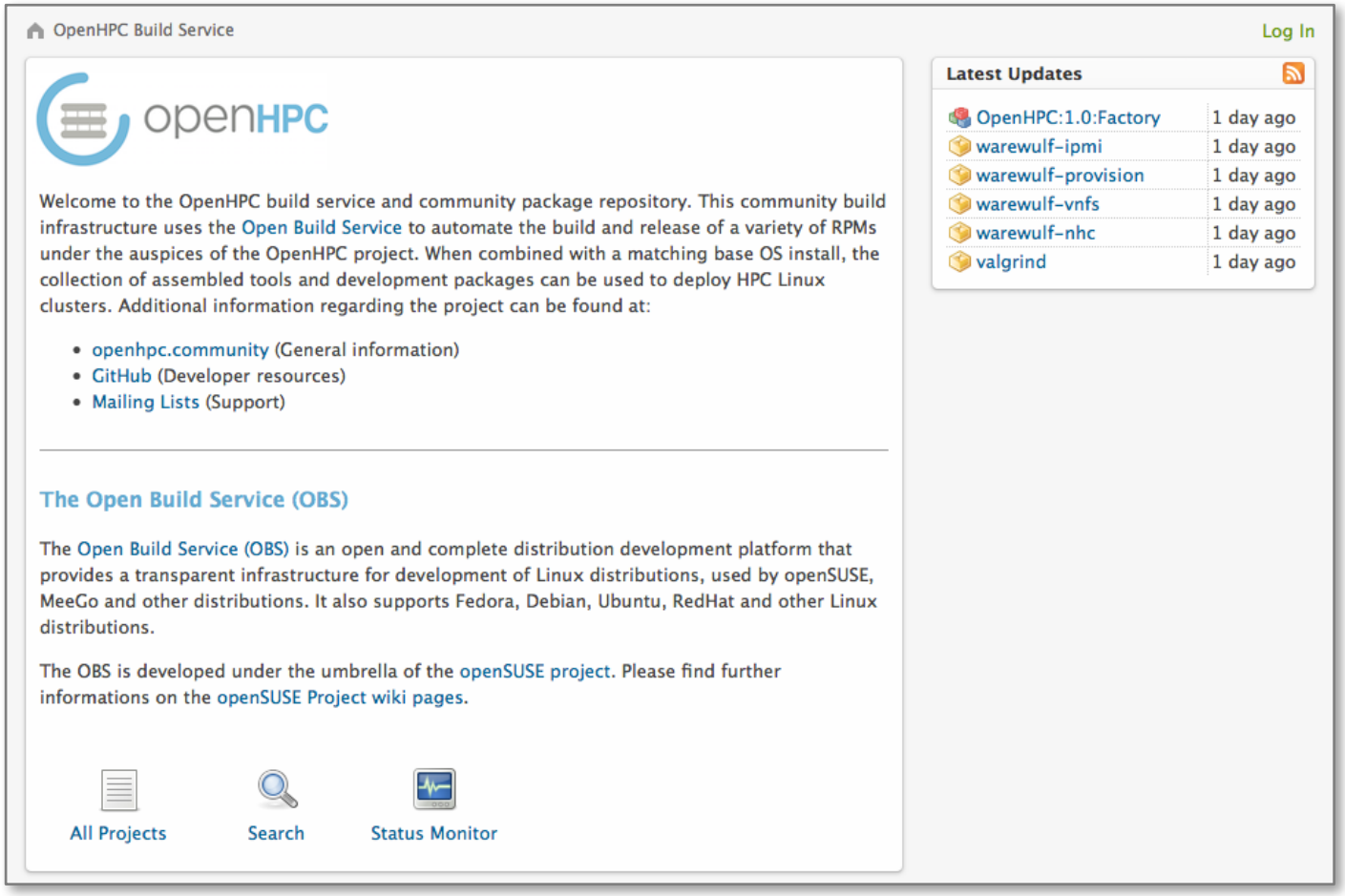


- single input drives all permutations
- packaging conventions highlighted further in paper

Infrastructure

Community Build System - OBS

https://build.openhpc.community

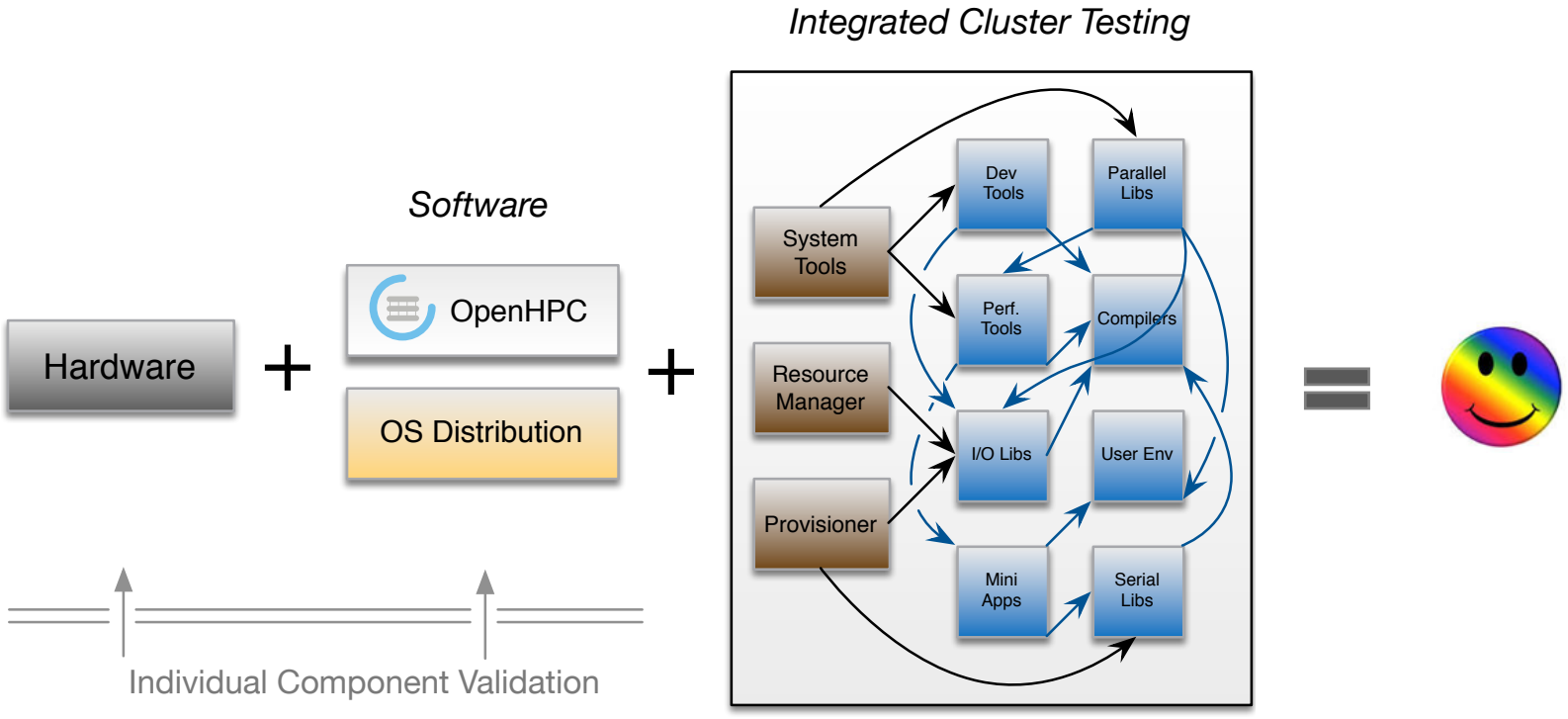


- Using the **Open Build Service** (OBS) to manage build process
- OBS can drive builds for multiple repositories
- Repeatable builds carried out in chroot environment
- **Generates binary and src rpms**
- Publishes corresponding package repositories
- Client/server architecture supports distributed build slaves and multiple architectures

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



Post Install Integration Tests - Overview

Global testing harness includes a number of embedded subcomponents:

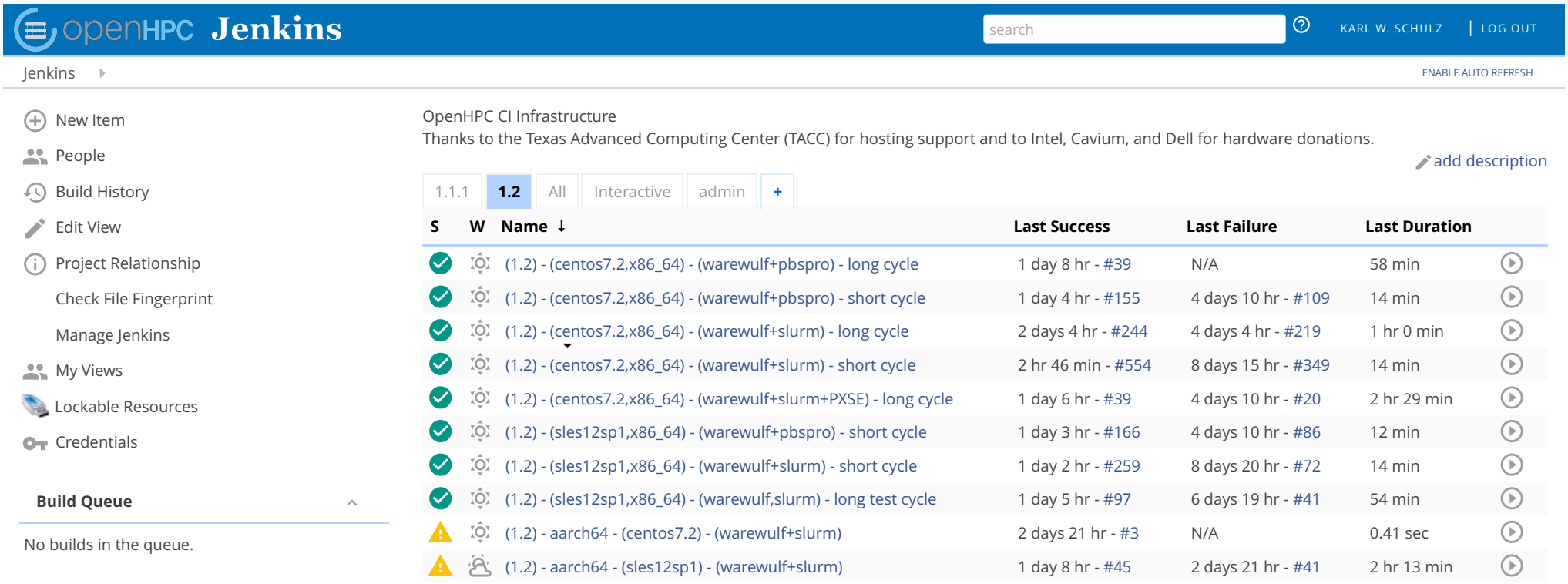
- major components have configuration options to enable/disable
  - end user tests need to touch all of the supported compiler and MPI families
  - we abstract this to repeat the tests with different compiler/MPI environments:
    - gcc/Intel compiler toolchains
    - MPICH, OpenMPI, MVAPICH2, Intel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

Community Test System (CI) - Jenkins

http://test.openhpc.community:8080



These tests periodically installing bare-metal clusters from scratch using OpenHPC recipes and then run a variety of integration tests.

Component Additions?

- A common question posed to the project has been how to request new software components? In response, the TSC has endeavored to formalize a simple submission/review process
- Submission site went live last month:

  https://github.com/openhpc/submissions

- Expecting to do reviews every quarter (or more frequent if possible)
  - just completed first iteration of the process now
  - next submission deadline: December 4th , 2016

Subset of information requested during submission process

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

Summary

- Community formalized as Linux Foundation collaborative project in May, 2016
- Technical Steering Committee (TSC) has been working together since the beginning of the summer
  - established a starting component selection process
  - latest release (Nov. 2016) incorporated additions based on this process
    - e.g. MPICH, PBS Pro, Scalasca/ScoreP
  - future addition to include xCAT based recipe
- OpenHPC BoF at SC'16
  - Wednesday, Nov. 16th (1:30-3pm)
- *We welcome participation from other interested researchers and end-user HPC sites*

Information/Places to Interact

http://openhpc.community (general info)

https://github.com/openhpc/ohpc (GitHub site)

https://github.com/openhpc/submissions (new submissions)

https://build.openhpc.community (build system/repos)

http://www.openhpc.community/support/mail-lists/ (email lists)

- openhpc-announce
- openhpc-users
- openhpc-devel

openHPC
http://openhpc.community

Thanks for your Time - Questions?

karl.w.schulz@intel.com