OpenHPC: Community Building Blocks for HPC

Karl W. Schulz, Ph.D.
Research Associate Professor
Oden Institute for Computational Engineering and Sciences and Dell Medical School

HPC Knowledge Meeting
hpcpk'19 • June 13, 2019 • Barcelona, Spain

https://openhpc.community
Outline

• What is OpenHPC: project overview
  – mission/vision
  – members, governance, adoption

• More technical details
  – key takeaways
  – packaging conventions
  – build/test environment
  – highlights from latest release and what's coming
What is the project’s 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: Current Project Members

Member participation interest? Contact Neil Caidin ncaidin@linuxfoundation.org
How is the community project governed?

Governance is dual-pronged with a Governing Board + Technical Steering Committee

OpenHPC Governing Board

- Responsible for budgetary oversight, marketing, long-term roadmap guidance

OpenHPC Technical Steering Committee (TSC)

- Project Leader
- Testing Coordinators
- Component Development Reps.
- End-User / Site Reps.
- Maintainers

Technical Project Leadership

HPC Community

TSC terms are for 1 year: call for nominations put out each summer in May - deadline is June 14!
Next: High-level Takeaways for OpenHPC...
OpenHPC: a building block repository

[ Key takeaway ]

- 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.
  - Public package repositories also make it easy to include desired elements in customized containers (e.g. Docker, Singularity, Charliecloud).
[ Key takeaway ]

- Designation of compiler/MPI variants is fundamental to OpenHPC’s build, test, and end-user environment.
- OpenHPC packaging is architected to allow multiple variants to be added over time.
- Example: two gcc variants are installed.

OpenHPC: A Hierarchical HPC Software Environment

$ module list
Currently Loaded Modules:
1) autotools 2) prun/1.2 3) gnu7/7.3.0 4) mvapich2/2.2 5) ohpc

$ module avail

```bash
adios/1.13.1 mpiP/3.4.1 phdf5/1.10.2 py3-scipy/1.1.0 superlu_dist/5.3.0
boost/1.67.0 mumps/5.1.2 pnetcdf/1.9.0 scalapack/2.0.2 tau/2.27.1
fftw/3.3.7 netcdf-cxx4/4.3.0 ptscotch/6.0.4 scalasca/2.3.1 trilinos/12.12.1
hypre/2.14.0 netcdf-fortran/4.4.4 py2-mpi4py/3.0.0 scorep/4.0
imb/2018.1 netcdf/4.6.1 py2-scipy/1.1.0 sionlib/1.7.1
mfem/3.3.2 petsc/3.9.1 py3-mpi4py/3.0.0 slepc/3.9.1
```

```
R/3.5.0 likwid/4.3.2 mvapich2/2.2 (L) openmpi3/3.1.0 py2-numpy/1.14.3 superlu/5.2.1
gsl/2.4 metis/5.1.0 ocr/1.0.1 pdtoolkit/3.25 py3-numpy/1.14.3
hdf5/1.10.2 mpich/3.2.1 openblas/0.2.20 plasma/2.8.0
```

```
EasyBuild/3.6.1 clustershell/1.8 gnu7/7.3.0 (L) ohpc (L) prun/1.2 (L)
autotools (L) cmake/3.11.1
charliecloud/0.2.4 hwloc/1.11.10
```

[ compiler tier ]

[ MPI tier ]
Quick (hierarchical) 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

```bash
# yum install petsc-gnu7-mvapich2-ohpc
...
```

<table>
<thead>
<tr>
<th>Package</th>
<th>Arch</th>
<th>Version</th>
<th>Repository</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>petsc-gnu7-mvapich2-ohpc</td>
<td>x86_64</td>
<td>3.7.6-104.1</td>
<td>OpenHPC-updates</td>
<td>13 M</td>
</tr>
<tr>
<td>gnu7-compilers-ohpc</td>
<td>x86_64</td>
<td>7.1.0-29.3</td>
<td>OpenHPC-updates</td>
<td>51 M</td>
</tr>
<tr>
<td>mvapich2-gnu7-ohpc</td>
<td>x86_64</td>
<td>2.2-26.2</td>
<td>OpenHPC-updates</td>
<td>8.6 M</td>
</tr>
<tr>
<td>openblas-gnu7-ohpc</td>
<td>x86_64</td>
<td>0.2.19-25.1</td>
<td>OpenHPC-updates</td>
<td>3.4 M</td>
</tr>
<tr>
<td>phdf5-gnu7-mvapich2-ohpc</td>
<td>x86_64</td>
<td>1.10.0-83.1</td>
<td>OpenHPC-updates</td>
<td>2.9 M</td>
</tr>
<tr>
<td>prun-ohpc</td>
<td>noarch</td>
<td>1.1-21.1</td>
<td>OpenHPC</td>
<td>9.5 k</td>
</tr>
<tr>
<td>scalapack-gnu7-mvapich2-ohpc</td>
<td>x86_64</td>
<td>2.0.2-39.1</td>
<td>OpenHPC-updates</td>
<td>3.4 M</td>
</tr>
</tbody>
</table>
OpenHPC: validated recipes for system install

[ Key takeaway ]

- 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
- 10 different recipes available with latest release
- the docs-ohpc RPM installs these recipes (along with shell scripts encapsulating all commands)

#### x86_64:
- Install_guide-CentOS7-Warewulf-PBSPro-1.3.8-x86_64.pdf
- Install_guide-CentOS7-Warewulf-SLURM-1.3.8-x86_64.pdf
- Install_guide-CentOS7-xCAT-Stateful-SLURM-1.3.8-x86_64.pdf
- Install_guide-CentOS7-xCAT-Stateless-SLURM-1.3.8-x86_64.pdf
- Install_guide-SLE_12-Warewulf-PBSPro-1.3.8-x86_64.pdf
- Install_guide-SLE_12-Warewulf-SLURM-1.3.8-x86_64.pdf

#### aarch64:
- Install_guide-CentOS7-Warewulf-PBSPro-1.3.8-aarch64.pdf
- Install_guide-CentOS7-Warewulf-SLURM-1.3.8-aarch64.pdf
- Install_guide-SLE_12-Warewulf-PBSPro-1.3.8-aarch64.pdf
- Install_guide-SLE_12-Warewulf-SLURM-1.3.8-aarch64.pdf

excellent place to start if you are new to the project
OpenHPC: Multiple Architecture Support

[Key takeaway]

- aarch64 and x86_64 package repositories are available
- benefit is that OpenHPC provides consistent development environment to the end user across multiple architectures
Is anybody using this...?
Project Adoption Growth

- Continued access/download growth since initial release at SC’15
- Plots highlight number of unique visitors/month to the OpenHPC build server/repo(s)
- Over 21TB downloaded in 2018
Additional Technical Details…

More comments on:
- packaging conventions
- infrastructure
- build and test environment
- current release
Packaging Conventions

- OpenHPC tries to play nicely with underlying OS’es and other repositories
- We endeavor to avoid namespace conflict in a variety of ways:
  - **install paths**: end-user oriented components housed under `/opt/ohpc/pub` (also allow for multiple versions to coexist)
  - **package names**: RPMs provided via OpenHPC repos include “-ohpc” as the suffix for the package name, for example:
    
    `Imod-ohpc-7.4.8-11.1.aarch64.rpm`
  - **dependencies**: compiler/MPI variant dependencies are explicitly managed
    
    ```
    %if "%{compiler_family}" == "gnu7"
    BuildRequires: gnu7-compilers%{PROJ_DELIM} >= 7.2.0
    Requires: gnu7-compilers%{PROJ_DELIM} >= 7.2.0
    %endif
    %if "%{mpi_family}" == "mvapich2"
    BuildRequires: mvapich2-%{compiler_family}%{PROJ_DELIM}
    Requires: mvapich2-%{compiler_family}%{PROJ_DELIM}
    %endif
    ```

[relevant logic from OHPC_macros]
Packaging Conventions

- Packaging adopts a consistent environment variable schema in modulefiles (we rely on Lmod implementation of modules):
  - `$PACKAGING_DIR` - top level install path of package
  - `$PACKAGING_BIN` - path to binaries supplied by package
  - `$PACKAGING_INC` - path to include headers for package
  - `$PACKAGING_LIB` - path to dynamic libraries for package

- Recommend using these vars 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

```bash
$ module load gnu8 mvapich2
$ module load petsc
$ mpicc -I$PETSC_INC petsc_hello.c -L$PETSC_LIB -lpetsc
```
Packaging conventions: custom ohpc RPM plugin

To avoid namespace collision for resolving dynamic libraries (.so's), we apply an "ohpc" color delimiter to libraries installed in /opt/ohpc/ path

# rpm -q --requires gnu7-compilers-ohpc | egrep "libc.so|libgcc"

```
libgcc_s.so.1()(64bit)(ohpc)
libgcc_s.so.1(GCC_3.0)(64bit)(ohpc)
libgcc_s.so.1(GCC_3.3)(64bit)(ohpc)
libgcc_s.so.1(GCC_4.2.0)(64bit)(ohpc)
libgcc_s.so.1(GCC_4.3.0)(64bit)(ohpc)
```

- .so's self contained within ohpc gcc build
- .so's required from base OS supplied packages

Custom RPM dependency analysis using ohpc plugin

**Convention introduced in v1.3.4 release**
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)
- Bats (unit testing for Bash)

Capable build/packaging system:
- 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
Reproducible builds using OBS

https://build.openhpc.community

- We use the **Open Build Service (OBS)** to manage build processes
- OBS can drive builds for multiple OS's and architectures
- Repeatable builds carried out in chroot environment
- **Generates binary and src rpms**
- Client/server architecture supports distributed build slaves and multiple architectures
- OpenHPC adopts a single .spec file input for builds
- Dependencies are self-managed and designed to avoid namespace conflict with distro provided variants

```xml
<link project='OpenHPC:1.3:Update6:Factory'
package='boost-gnu-openmpi'>
<patches>
 <topadd>%define compiler_family gnu8</topadd>
 <topadd>%define mpi_family mpich</topadd>
</patches>
</link>
```
Build system - distributed build servers

- We leverage ability in OBS to have distributed build servers:
  - x86_64: hosted at TACC (thanks to kind hardware donations) and on EC2
  - aarch64: hosted at Packet (thanks to Works on Arm project)
Integration/Test/Validation

Post-build testing is another **key element** for us and the intent is to build upon existing 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
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, MPICH, MVAPICH2 MPI families

`installable as test-suite-ohpc RPM`
Community Jenkins Instance for CI Testing

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

OpenHPC CI Infrastructure

- 1.1.1
- 1.2
- 1.3
- 1.4

<table>
<thead>
<tr>
<th>Last Success</th>
<th>Last Failure</th>
<th>Last Duration</th>
<th>Test Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 hr 19 min - #93</td>
<td>4 days 3 hr - #81</td>
<td>2 hr 50 min</td>
<td>N/A</td>
</tr>
<tr>
<td>3 hr 19 min - #93</td>
<td>4 days 11 hr - #29</td>
<td>1 hr 27 min</td>
<td>0 of 1,430 failed (0)</td>
</tr>
<tr>
<td>2 hr 29 min - #287</td>
<td>1 hr 19 min</td>
<td>0 of 1,430 failed (-2)</td>
<td></td>
</tr>
<tr>
<td>2 hr 13 min</td>
<td>1 hr 27 min</td>
<td>0 of 1,133 failed (0)</td>
<td></td>
</tr>
<tr>
<td>3 hr 15 min - #64</td>
<td>4 days 11 hr - #29</td>
<td>1 hr 27 min</td>
<td>0 of 1,133 failed (0)</td>
</tr>
<tr>
<td>2 hr 29 min - #287</td>
<td>1 hr 19 min</td>
<td>0 of 1,133 failed (0)</td>
<td></td>
</tr>
<tr>
<td>2 hr 29 min - #738</td>
<td>N/A</td>
<td>1 hr 8 min</td>
<td>0 of 1,133 failed (0)</td>
</tr>
<tr>
<td>2 hr 29 min - #287</td>
<td>5 days 0 hr - #238</td>
<td>50 min</td>
<td>0 of 1,133 failed (0)</td>
</tr>
<tr>
<td>3 hr 43 min - #285</td>
<td>1 day 22 hr - #271</td>
<td>2 hr 29 min</td>
<td>0 of 1,133 failed (0)</td>
</tr>
<tr>
<td>3 hr 24 min - #22</td>
<td>8 hr 23 min - #21</td>
<td>1 hr 27 min</td>
<td>0 of 1,133 failed (0)</td>
</tr>
<tr>
<td>3 hr 24 min - #22</td>
<td>8 hr 23 min - #21</td>
<td>1 hr 27 min</td>
<td>0 of 1,133 failed (0)</td>
</tr>
<tr>
<td>5 hr 6 min - #322</td>
<td>5 days 16 hr - #293</td>
<td>2 hr 29 min</td>
<td>0 of 1,133 failed (0)</td>
</tr>
<tr>
<td>4 days 16 hr - #2</td>
<td>N/A</td>
<td>0.73 sec</td>
<td>0 of 1,133 failed (0)</td>
</tr>
<tr>
<td>4 days 9 hr - #2</td>
<td>N/A</td>
<td>0.6 sec</td>
<td>0 of 1,133 failed (-3)</td>
</tr>
<tr>
<td>2 hr 53 min - #288</td>
<td>4 days 8 hr - #254</td>
<td>1 hr 25 min</td>
<td>0 of 1,133 failed (0)</td>
</tr>
<tr>
<td>1 hr 26 min - #455</td>
<td>17 hr - #19</td>
<td>1 hr 11 min</td>
<td>N/A</td>
</tr>
<tr>
<td>1 hr 44 min - #23</td>
<td>17 hr - #19</td>
<td>1 hr 0 min</td>
<td>0 of 1,142 failed (0)</td>
</tr>
<tr>
<td>3 hr 4 min - #456</td>
<td>2 days 9 hr - #438</td>
<td>1 hr 2 min</td>
<td>0 of 1,142 failed (0)</td>
</tr>
<tr>
<td>3 hr 26 min - #492</td>
<td>5 days 17 hr - #447</td>
<td>59 min</td>
<td>0 of 1,142 failed (0)</td>
</tr>
<tr>
<td>1 hr 26 min - #455</td>
<td>4 days 13 hr - #419</td>
<td>1 hr 11 min</td>
<td>0 of 1,142 failed (0)</td>
</tr>
<tr>
<td>2 hr 36 min - #192</td>
<td>3 days 8 hr - #175</td>
<td>2 hr 3 min</td>
<td>0 of 2,326 failed (0)</td>
</tr>
</tbody>
</table>

All recipes exercised in CI system (start w/ bare-metal installs + integration test suite)

our CI "build" is really a cluster
Latest Release and Stats
Additions and Upstream Version Changes

- Part of the motivation for community effort like OpenHPC is the rapidity of S/W updates that occurs in our space.
- We have been doing releases on a roughly quarterly cadence:
  - convention is to go with latest stable release of upstream components
  - additional components added over time
## OpenHPC v1.3.8 - S/W components

**Released June 11, 2019**

- 3rd Party libraries are built for each compiler/MPI family
- Resulting repositories currently comprised of ~700 RPMs

<table>
<thead>
<tr>
<th>Functional Areas</th>
<th>Components</th>
<th>new in v1.3.8 release</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base OS</td>
<td>CentOS 7.6, SLES12 SP4</td>
<td></td>
</tr>
<tr>
<td>Architecture</td>
<td>aarch64, x86_64</td>
<td></td>
</tr>
<tr>
<td>Administrative Tools</td>
<td>Conman, Ganglia, Lmod, LosF, Nagios, NHC, pdsh, pdsh-mod-slurm, prun, EasyBuild, ClusterShell, mrsh, Genders, Shine, Spack, test-suite</td>
<td></td>
</tr>
<tr>
<td>Provisioning</td>
<td>Warewulf, xCAT</td>
<td></td>
</tr>
<tr>
<td>Resource Mgmt.</td>
<td>SLURM, Munge, PBS Professional, PMIx</td>
<td></td>
</tr>
<tr>
<td>Runtimes</td>
<td>Charliecloud, OpenMP, OCR, Singularity</td>
<td></td>
</tr>
<tr>
<td>I/O Services</td>
<td>Lustre client, BeeGFS client*</td>
<td></td>
</tr>
<tr>
<td>Numerical/Scientific Libraries</td>
<td>Boost, GSL, FFTW, Hypre, Metis, MFEM, Mumps, OpenBLAS, OpenCoarrays, PETSc, PLASMA, Scalapack, Scotch, SLEPc, SuperLU, SuperLU_Dist, Trilinos</td>
<td></td>
</tr>
<tr>
<td>I/O Libraries</td>
<td>HDF5 (pHDF5), NetCDF/pNetCDF (including C++ and Fortran interfaces), Adios</td>
<td></td>
</tr>
<tr>
<td>Compiler Families</td>
<td>GNU (gcc, g++, gfortran), Clang/LLVM, Intel Parallel Studio*</td>
<td></td>
</tr>
<tr>
<td>MPI Families</td>
<td>MVAPICH2, OpenMPI, MPICH, Intel MPI*</td>
<td></td>
</tr>
<tr>
<td>Development Tools</td>
<td>Autotools, cmake, hwloc, mpi4py, R, SciPy/NumPy, Valgrind</td>
<td></td>
</tr>
<tr>
<td>Performance Tools</td>
<td>PAPI, IMB, Likwid, mpiP, ptoolkit TAU, Scalasca, ScoreP, SIONLib, GeoPM, msr-safe, Dimemas, Extrae, Paraver, OSU Microbenchmarks</td>
<td></td>
</tr>
</tbody>
</table>
How to Request Additional Software?

- We have a simple submission site for new requests:
  - https://github.com/openhpc/submissions

- Example components added via this mechanism since the v1.2. release (Nov’ 16)
  - BeeGFS client
  - xCAT recipe
  - hwloc
  - Singularity
  - LLVM/clang
  - PLASMA
  - pNetCDF
  - SCOTCH
  - SLEPc
  - PMIx
  - MPI4py
  - Likwid
  - MFEM
  - NHC
  - Charliecloud
  - GeoPM
  - Dimemas/Extrae, Paraver
  - OpenCoarrays
  - OSU Benchmarks
What's coming for next big release?

● Will be migrating from 1.3.x series to 2.0 release (targeting SC'19)
● Target major new distro versions:
  ○ CentOS 8
  ○ SUSE Leap 15.1 (switching to Leap from SLES)

● Once we release a new major branch (e.g. v2.x) which supports a newer major distro version at time $t$, we will restrict updates to previous branch (e.g. v1.3.x) to include:
  ○ security patches (e.g. address known CVEs)
  ○ significant bugs affecting functionality
  ○ older branch stays in maintenance mode for $\Delta t = 1$ year
Summary

• Provided an overview of the Linux Foundation project along with key takeaways and packaging details

• Community project efforts provides package repository for use with native OS package managers
  – for use on bare-metal or cloud/containerized environments
  – significant companion test effort with documentation recipes for deployment and execution in HPC- runtime environment
  – RHEL8/CentOS8 and SUSE Leap 15.1 coming later this year
Thanks for your time....

- OpenHPC Home: http://www.openhpc.community/
- Primary GitHub Site: https://github.com/openhpc/ohpc
- Package Repositories: http://build.openhpc.community/OpenHPC:
- 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

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