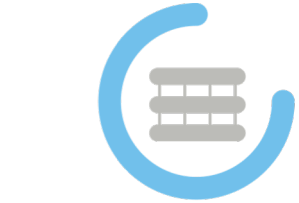
OpenHPC: Community Project Updates



https://openhpc.community

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

6th Annual MVAPICH User Group (MUG) Meeting
August 8, 2018 • Columbus, Ohio

Outline

- Brief project overview
- New items/updates from last year
- Fun with packaging

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.

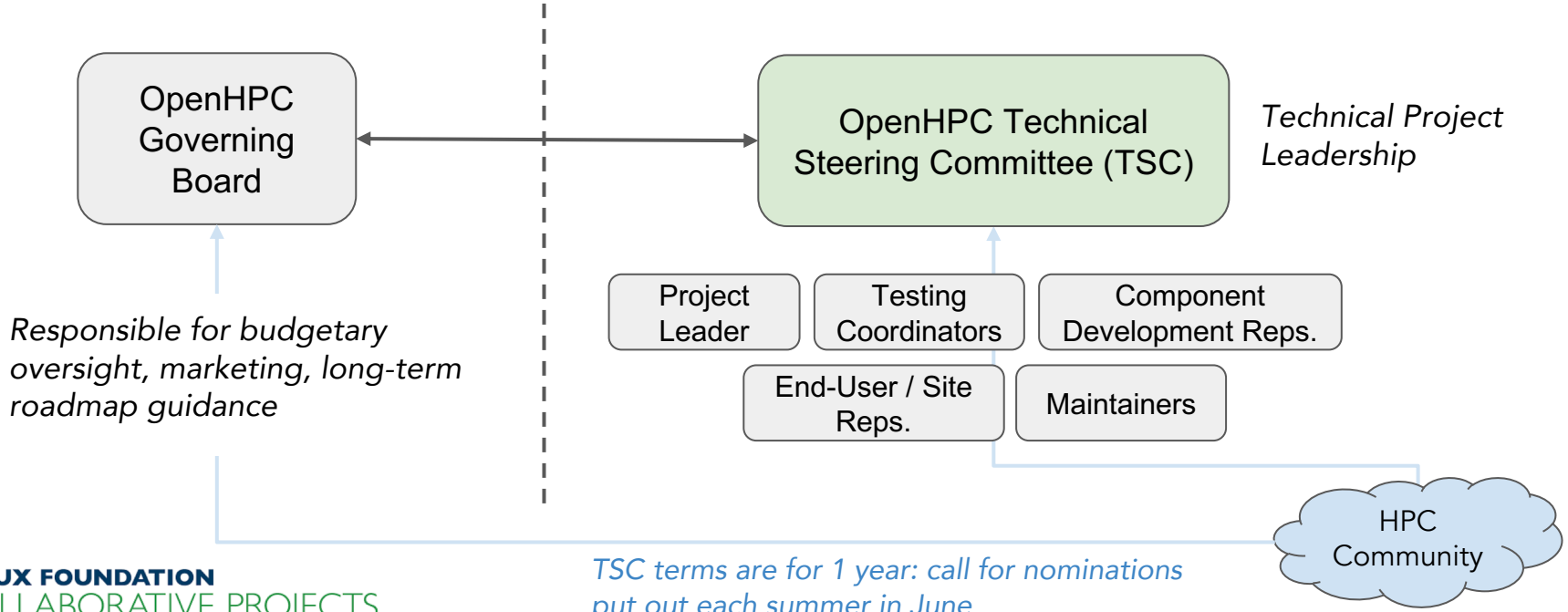
Vision: OpenHPC components and best practices will enable and accelerate innovation and discoveries by broadening access to state-of-the-art, open-source HPC methods and tools in a consistent environment, supported by a collaborative, worldwide community of HPC users, developers, researchers, administrators,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

OpenHPC: Growing Project Membership

Member participation interest? Contact Jeff ErnstFriedman, jernstfriedman@linuxfoundation.org

How is the community project governed?

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



TSC terms are for 1 year: call for nominations put out each summer in June

OpenHPC TSC – Individual Members

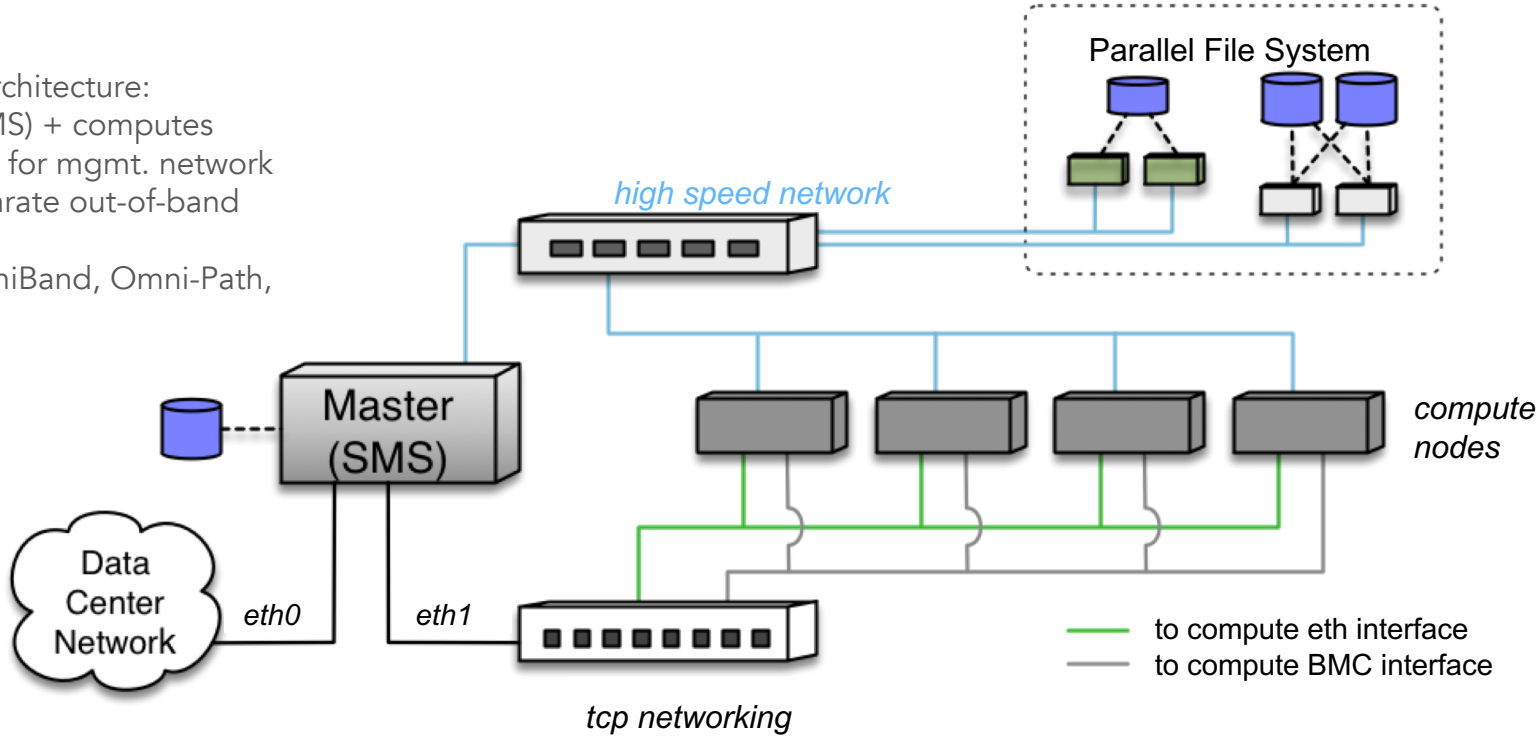
- Reese Baird, Intel (Maintainer)
- David Brayford, LRZ (Maintainer)
- Eric Coulter, Indiana University (End-User/Site Representative)
- Chris Downing, Red Oak Consulting (Maintainer)
- Craig Gardner, SUSE (Maintainer)
- Renato Golin, Linaro (Testing Coordinator)
- Michael Karo, Altair (Maintainer)
- Janet Lebens, Cray (Maintainer)
- Thomas Moschny, ParTec (Maintainer)
- Takayuki Okamoto, Fujitsu (Maintainer)
- Kevin Pedretti, Sandia National Laboratory (Maintainer)
- Paul Peltz, Los Alamos National Laboratory (Maintainer)
- Nam Pho, Harvard Medical School (Maintainer)
- Cyrus Proctor, Texas Advanced Computing Center (Maintainer)
- Adrian Reber, Red Hat (Maintainer)
- Karl W. Schulz, UT Austin (Project Lead, Testing Coordinator)
- Jeff Schutkoske, Cray (Component Development Representative)
- Derek Simmel, Pittsburgh Supercomputing Center (End-User/Site Representative)
- Chris Simmons, UT Dallas (Maintainer)
- Nirmala Sundararajan, Dell (Maintainer)

New for 2018-2019

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

Target System Architecture

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



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)

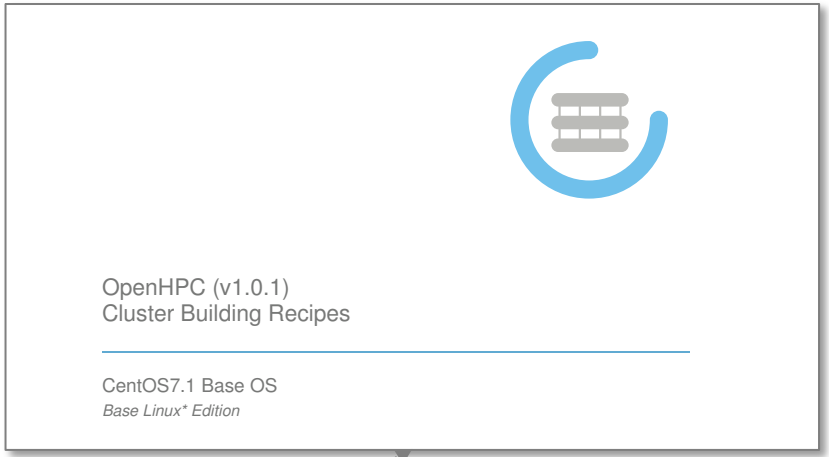
- You can pick relevant bits of interest for your site
  - if you prefer a resource manager that is not included, you can build that locally and still leverage the scientific libraries and development environment
  - similarly, you might prefer to utilize a different provisioning system

Newish Items/Updates

changes and new items since we were last together at MUG'17

Variety of recipes now available

Choose your own adventure…



Initially, we started off with a single recipe with the intent to expand

10 recipes now available with v1.3.5 release

We continue to expand recipe option(s) with multiple resource managers, OSes, provisioners, and architectures:

x86_64:

- Install_guide-CentOS7-Warewulf-PBSPro-1.3.5-x86_64.pdf
- Install_guide-CentOS7-Warewulf-SLURM-1.3.5-x86_64.pdf
- Install_guide-CentOS7-xCAT-Stateful-SLURM-1.3.5-x86_64.pdf
- Install_guide-CentOS7-xCAT-Stateless-SLURM-1.3.5-x86_64.pdf
- Install_guide-SLE_12-Warewulf-PBSPro-1.3.5-x86_64.pdf
- Install_guide-SLE_12-Warewulf-SLURM-1.3.5-x86_64.pdf

aarch64:

- Install_guide-CentOS7-Warewulf-PBSPro-1.3.5-aarch64.pdf
- Install_guide-CentOS7-Warewulf-SLURM-1.3.5-aarch64.pdf
- Install_guide-SLE_12-Warewulf-PBSPro-1.3.5-aarch64.pdf
- Install_guide-SLE_12-Warewulf-SLURM-1.3.5-aarch64.pdf

rysnc now supported for OpenHPC repos

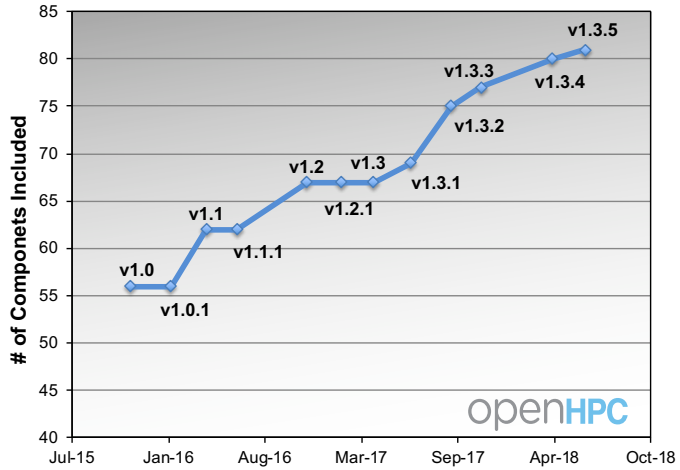
- At SC'17 BoF, we had a question asking if we could enable rsync for mirroring of OpenHPC repositories
- We have control over the back end, so rsync now supported for repos hosted at:

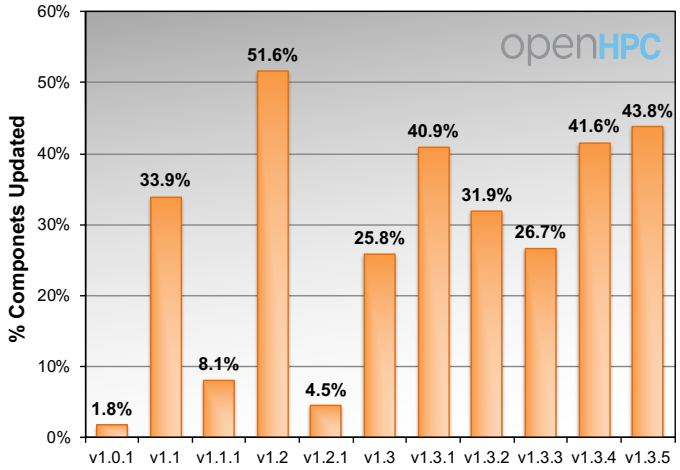
  http://build.openhpc.community/OpenHPC:/
- Three methods now available for access:
  - direct repo access via yum/zipper (requires external routing)
  - local repo mirroring (download self-contained tarball with binary/src RPMs)
  - local repo mirroring (via rsync)
- Typical rsync mirroring example for binaries in 1.3 release(s):

```
$ rsync -avzH --exclude src --exclude repocache --delay-updates
rsync://build.openhpc.community/OpenHPC/1.3/ 1.3
```

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





Released June 13, 2018

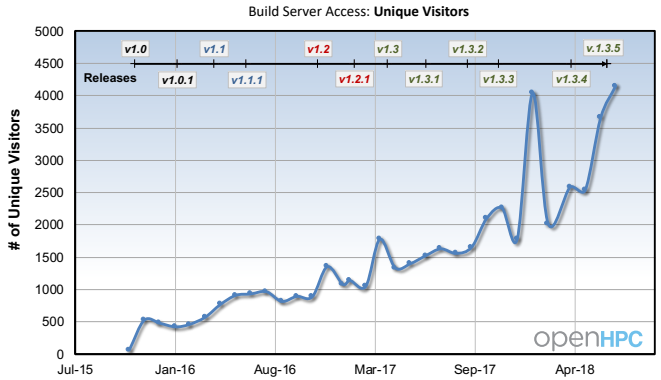
OpenHPC v1.3.5 - S/W components

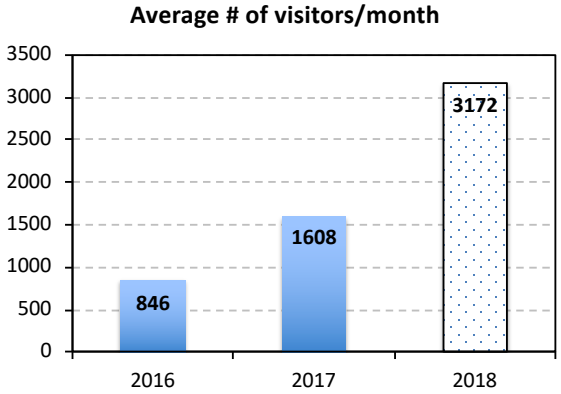
components available 81 | new additions 1 | updates 44%

Functional Areas	Components (new since v1.3.1)
Base OS	CentOS 7.5, SLES12 SP3
Architecture	aarch64, x86_64
Administrative Tools	Conman, Ganglia, Lmod, LosF, Nagios, NHC, pdsh, pdsh-mod-slurm, prun, EasyBuild, ClusterShell, mrsh, Genders, Shine, Spack, test-suite
Provisioning	Warewulf, xCAT
Resource Mgmt.	SLURM, Munge, PBS Professional, PMIx
Runtimes	Charliecloud, OpenMP, OCR, Singularity
I/O Services	Lustre client, BeeGFS client
Numerical/Scientific Libraries	Boost, GSL, FFTW, Hypre, Metis, MFEM, Mumps, OpenBLAS, PETSc, PLASMA, Scalapack, Scotch, SLEPc, SuperLU, SuperLU_Dist, Trilinos
I/O Libraries	HDF5 (pHDF5), NetCDF/pNetCDF (including C++ and Fortran interfaces), Adios
Compiler Families	GNU (gcc, g++, gfortran),), Clang/LLVM
MPI Families	MVAPICH2, OpenMPI, MPICH
Development Tools	Autotools, cmake, hwloc, mpi4py, R, SciPy/NumPy, Valgrind
Performance Tools	PAPI, IMB, Likwid, mpiP, pdtoolkit TAU, Scalasca, ScoreP, SIONLib

- 3rd Party libraries are built for each compiler/MPI family
- Resulting repositories currently comprised of ~600 RPMs

Project Access Growth



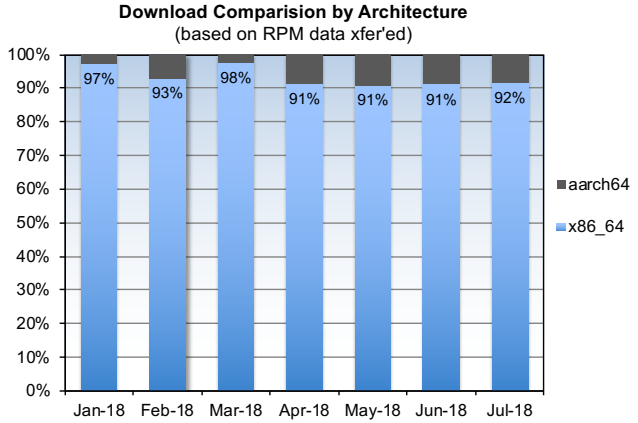


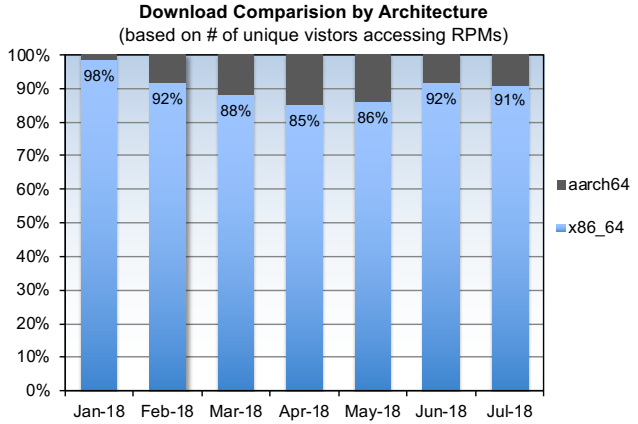
- We are no MVAPICH with more than 485,000 downloads from OSU project site
- But, are seeing continued growth since initial release at SC'15
- Example of end sites accessing multiple MVAPICH related builds:

Bandwidth	Data
11.1 GiB (3.71%)	/OpenHPC:/1.3/updates/CentOS_7/x86_64/trilinos-gnu7-mvapich2-ohpc-12.12.1-12.7.x86_64.rpm
8.94 GiB (2.98%)	/OpenHPC:/1.3/updates/CentOS_7/x86_64/tau-gnu7-mvapich2-ohpc-2.27-11.1.x86_64.rpm
7.41 GiB (2.47%)	/OpenHPC:/1.3/updates/CentOS_7/x86_64/python34-scipy-gnu7-mvapich2-ohpc-1.0.0-5.2.x86_64.rpm
7.22 GiB (2.41%)	/OpenHPC:/1.3/updates/CentOS_7/x86_64/python-scipy-gnu7-mvapich2-ohpc-1.0.0-5.2.x86_64.rpm
6.36 GiB (2.12%)	/OpenHPC:/1.3/updates/CentOS_7/x86_64/tau-gnu7-mvapich2-ohpc-2.27.1-24.1.x86_64.rpm
5.52 GiB (1.84%)	/OpenHPC:/1.3/updates/CentOS_7/x86_64/python34-scipy-gnu7-mvapich2-ohpc-1.1.0-4.1.x86_64.rpm
5.35 GiB (1.79%)	/OpenHPC:/1.3/updates/CentOS_7/x86_64/python-scipy-gnu7-mvapich2-ohpc-1.1.0-4.1.x86_64.rpm
4.80 GiB (1.60%)	/OpenHPC:/1.3/updates/CentOS_7/x86_64/scorep-gnu7-mvapich2-ohpc-3.1-156.2.x86_64.rpm
4.32 GiB (1.44%)	/OpenHPC:/1.3/updates/CentOS_7/x86_64/scorep-gnu7-mvapich2-ohpc-4.0-4.1.x86_64.rpm
4.03 GiB (1.35%)	/OpenHPC:/1.3/updates/CentOS_7/x86_64/**mvapich2-gnu7-ohpc-2.2-37.4.x86_64.rpm**
3.99 GiB (1.33%)	/OpenHPC:/1.3/updates/CentOS_7/x86_64/petsc-gnu7-mvapich2-ohpc-3.8.3-6.8.x86_64.rpm
2.79 GiB (0.93%)	/OpenHPC:/1.3/updates/CentOS_7/x86_64/boost-gnu7-mvapich2-ohpc-1.66.0-5.8.x86_64.rpm
2.64 GiB (0.88%)	/OpenHPC:/1.3/CentOS_7/x86_64/python-scipy-gnu-mvapich2-ohpc-0.19.0-61.1.x86_64.rpm
2.33 GiB (0.78%)	/OpenHPC:/1.3/CentOS_7/x86_64/trilinos-gnu-mvapich2-ohpc-12.10.1-124.1.x86_64.rpm
1.74 GiB (0.58%)	/OpenHPC:/1.3/updates/CentOS_7/x86_64/adios-gnu7-mvapich2-ohpc-1.13.0-6.1.x86_64.rpm
1.72 GiB (0.58%)	/OpenHPC:/1.3/updates/CentOS_7/x86_64/mvapich2-psm2-gnu7-ohpc-2.2-194.1.x86_64.rpm
1.39 GiB (0.47%)	/OpenHPC:/1.3/CentOS_7/x86_64/mvapich2-gnu-ohpc-2.2-19.1.x86_64.rpm
1.32 GiB (0.44%)	/OpenHPC:/1.3/updates/CentOS_7/x86_64/adios-gnu7-mvapich2-ohpc-1.13.1-5.1.x86_64.rpm
1.32 GiB (0.44%)	/OpenHPC:/1.3/CentOS_7/x86_64/boost-gnu-mvapich2-ohpc-1.63.0-57.1.x86_64.rpm
1.20 GiB (0.40%)	/OpenHPC:/1.3/CentOS_7/x86_64/tau-intel-mvapich2-ohpc-2.26-147.1.x86_64.rpm

Multiple Architecture Support

- ARM tech preview first introduced with SC'16 release
- Removed tech preview designation at SC'17 with introduction of bare-metal provisioning support (v1.3.3 release)
- Starting to see some uptake in aarch64 builds in monitoring 2018 logs
  - Plots compare percentages for the amount of data xfer'ed and the # of unique visitors accessing (aarch64|x86_64) files
- Work in progress for future release:
  - companion build(s) with ARM HPC compilers





RPM dependency fun/pain

"Alexa….what in the wide wide world is rpmbuild doing under the covers?"

Quick Comments on Packaging Conventions

- OpenHPC tries to play nicely with underlying OS'es and other repositories
- 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:

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

  - dependencies: compiler/MPI variant dependencies 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]

Quick Comments on Packaging Conventions

- This is all well and good: but important to remember that rpm itself has it's own automatic dependency analysis mechanism:
  - scans package contents for necessary binaries, e.g.
    /bin/bash
    /bin/perl
  - scans package contents for shared library usage: embeds these as 'Requirements', e.g.

    libmpi.so.12()(64bit) → /opt/mvapich2-x/gnu5.4.0/ifs10.6/basic/mpirun/lib64/libmpi.so

    e.g. from MVAPICH2-X rpm
- we have learned the hard way that some difficulties can arise with the automatic dependency analysis when you do not install into default paths seen by linker
  - will highlight the gotcha
  - talk about how we fixed using RPM plugin infrastructure

RPM dependencies

- The gotcha: although unintended, we realized that we had some namespace collision with base OS packages with respect to .so libraries (coming out of automatic RPM dependency analysis):
- Two examples:

```
# repoquery --whatprovides "libopenblas.so.0()(64bit)”
  openblas-0:0.2.20-3.el7.x86_64
  openblas-gnu7-ohpc-0:0.2.20-1.ohpc.x86_64


# repoquery --whatprovides "libltdl.so.7()(64bit)"
  libtool-ltdl-0:2.4.2-22.el7_3.x86_64
  libtool-ohpc-0:2.4.6-1.x86_64
```

- Note: any ohpc packages which might require the ohpc variant of openblas or libtool included additional dependency information that call out the need for ohpc version
- No issues if using all ohpc variants. But, problem *can* arise via order of operations if user installs the ohpc variant of openblas, but then installs other packages requiring libopenblas.so from EPEL (since we explicitly avoid installing into distro paths)

RPM dependencies

- We wanted to see if we could possibly avoid this conflict by adjusting the automatic dependency analysis as it applies to shared libraries
  - end-game: add a delimiter of our choosing to .so dependencies
- Required some quality time getting to know more about RPMs dependency analysis scheme
  - there are multiple dependency generators associated with RPM
  - you will sometimes see them referred to as "internal" vs "external" dependency generators
- The "**external**" tools are the old mechanism and are scripts that get called after the build completes
  - /usr/lib/rpm/find-requires
  - /usr/lib/rpm/find-provides
  - Pros: they are scripts so they are easy to hack/modify
    Cons: this is the old way of doing things, newer rpm versions don't use by default
- The "**internal**" tools are the new default. They are binaries, e.g.
  - /usr/lib/rpm/rpmdeps
  - /usr/lib/rpm/elfdeps
  - Cons: can't directly hack without recompilation
    Pros: plugin infrastructure allows for customization

RPM dependencies – custom ohpc plugin

- The new dependency tool has a plugin infrastructure which allows you to call out to your own dependency generator on a per path (and per file type basis)
- Triggered via the addition of attribute files in /usr/lib/rpm/fileattrs
- We now include in our `ohpc-buildroot` package which is needed to build ohpc packages

```
# rpm -ql ohpc-buildroot
/opt/ohpc/admin/ohpc
/opt/ohpc/admin/ohpc/OHPC_setup_compiler
/opt/ohpc/admin/ohpc/OHPC_setup_mpi
/usr/lib/rpm
/usr/lib/rpm/fileattrs
/usr/lib/rpm/fileattrs/ohpc.attr
/usr/lib/rpm/ohpc-find-provides
/usr/lib/rpm/ohpc-find-requires
```

- Let's look at the magic in ohpc.attr:

```
%__ohpc_provides       /usr/lib/rpm/ohpc-find-provides
%__ohpc_requires       /usr/lib/rpm/ohpc-find-requires %{buildroot} /opt/ohpc

%__ohpc_path           ^/opt/ohpc
%__elf_exclude_path    ^/opt/ohpc

%__ohpc_magic          ^ELF (32|64)-bit.*$
%__ohpc_flags          magic_and_path
```

- General approach here is to trigger these scripts for any build that has ELF binaries found in /opt/ohpc
  - Provides logic: any .so found in /opt/ohpc has the "(ohpc)" color string appended
  - Requires logic: any .so found in the build (or coming in as BuildRequires) that lives in /opt/ohpc has matching "(ohpc)" color string appended. Otherwise, dependencies belonging to packages outside of OpenHPC maintain default string.

RPM dependencies: custom ohpc plugin

Let's look at some of the resulting Provides: differences for gcc using this approach.
Essentially, every .so detected in /opt/ohpc has delimiter appended

```
# rpm -q --provides gnu7-compilers-ohpc | egrep libgomp
```

OHPC 1.3.1 RPM

```
libgomp.so.1()(64bit)
libgomp.so.1(GOACC_2.0)(64bit)
libgomp.so.1(GOACC_2.0.1)(64bit)
libgomp.so.1(GOMP_1.0)(64bit)
libgomp.so.1(GOMP_2.0)(64bit)
libgomp.so.1(GOMP_3.0)(64bit)
libgomp.so.1(GOMP_4.0)(64bit)
libgomp.so.1(GOMP_4.0.1)(64bit)
libgomp.so.1(GOMP_4.5)(64bit)
libgomp.so.1(GOMP_PLUGIN_1.0)(64bit)
libgomp.so.1(GOMP_PLUGIN_1.1)(64bit)
libgomp.so.1(OACC_2.0)(64bit)
libgomp.so.1(OMP_1.0)(64bit)
libgomp.so.1(OMP_2.0)(64bit)
libgomp.so.1(OMP_3.0)(64bit)
libgomp.so.1(OMP_3.1)(64bit)
libgomp.so.1(OMP_4.0)(64bit)
libgomp.so.1(OMP_4.5)(64bit)
```

Build with Updated Approach

```
libgomp.so.1()(64bit)(ohpc)
libgomp.so.1(GOACC_2.0)(64bit)(ohpc)
libgomp.so.1(GOACC_2.0.1)(64bit)(ohpc)
libgomp.so.1(GOMP_1.0)(64bit)(ohpc)
libgomp.so.1(GOMP_2.0)(64bit)(ohpc)
libgomp.so.1(GOMP_3.0)(64bit)(ohpc)
libgomp.so.1(GOMP_4.0)(64bit)(ohpc)
libgomp.so.1(GOMP_4.0.1)(64bit)(ohpc)
libgomp.so.1(GOMP_4.5)(64bit)(ohpc)
libgomp.so.1(GOMP_PLUGIN_1.0)(64bit)(ohpc)
libgomp.so.1(GOMP_PLUGIN_1.1)(64bit)(ohpc)
libgomp.so.1(OACC_2.0)(64bit)(ohpc)
libgomp.so.1(OMP_1.0)(64bit)(ohpc)
libgomp.so.1(OMP_2.0)(64bit)(ohpc)
libgomp.so.1(OMP_3.0)(64bit)(ohpc)
libgomp.so.1(OMP_3.1)(64bit)(ohpc)
libgomp.so.1(OMP_4.0)(64bit)(ohpc)
libgomp.so.1(OMP_4.5)(64bit)(ohpc)
```

RPM dependencies: custom ohpc plugin

Let's look at some of the resulting Requires: differences for gcc using this approach.

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

OHPC 1.3.1 RPM

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

.so's contained within ohpc gcc build

Build with Updated Approach

```
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)
```

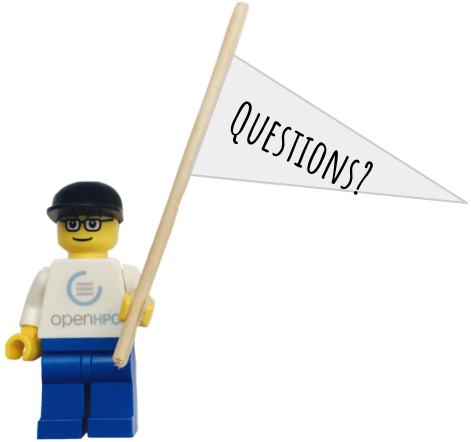
```
libc.so.6()(64bit)
libc.so.6(GLIBC_2.10)(64bit)
libc.so.6(GLIBC_2.11)(64bit)
libc.so.6(GLIBC_2.14)(64bit)
libc.so.6(GLIBC_2.16)(64bit)
libc.so.6(GLIBC_2.17)(64bit)
libc.so.6(GLIBC_2.2.5)(64bit)
libc.so.6(GLIBC_2.3)(64bit)
libc.so.6(GLIBC_2.3.2)(64bit)
libc.so.6(GLIBC_2.3.3)(64bit)
libc.so.6(GLIBC_2.6)(64bit)
libc.so.6(GLIBC_2.7)(64bit)
```

.so's required from base OS supplied packages

```
libc.so.6()(64bit)
libc.so.6(GLIBC_2.10)(64bit)
libc.so.6(GLIBC_2.11)(64bit)
libc.so.6(GLIBC_2.14)(64bit)
libc.so.6(GLIBC_2.16)(64bit)
libc.so.6(GLIBC_2.17)(64bit)
libc.so.6(GLIBC_2.2.5)(64bit)
libc.so.6(GLIBC_2.3)(64bit)
libc.so.6(GLIBC_2.3.2)(64bit)
libc.so.6(GLIBC_2.3.3)(64bit)
libc.so.6(GLIBC_2.6)(64bit)
libc.so.6(GLIBC_2.7)(64bit)
```

- We introduced this new convention in v1.3.4 release
- One could imagine doing something similar for MPI variant designation

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:/
- 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