Running an HPC app on the E4S container

https://e4s.io

Sameer Shende
Performance Research Laboratory, OACISSL, U. Oregon

sameer@cs.uoregon.edu

Tutorial, “Container Computing for HPC and Scientific Workloads”

Sunday, Nov. 17, 2019, 3:30 – 4:00 pm, Room # 207, Denver, CO.

https://sc19.supercomputing.org/presentation/?id=tut129&sess=sess206
Outline

• 13:30 – 13:45 Introduction to Containers in HPC (Younge)
• 13:45 – 14:15 How to build your first Docker container (Canon)
• 14:15 – 14:45 How to deploy a container on a supercomputer (Canon)
• 14:45 – 15:00 Best Practices (Canon)
• 15:00 – 15:30 -- Break –
• 15:30 – 16:00 Running an HPC app on the E4S container (Shende)
• 16:00 - 16:30 How to build a Singularity container image (Arango)
• 16:30 - 16:50 Running Singularity on a supercomputer & adv features (Arango)
• 16:50 - 17:00 Success Stories & Summary (Canon)

Link: https://tinyurl.com/sc19tut
Extreme-scale Scientific Software Stack (E4S)
https://e4s.io

- E4S is a community effort to provide open source software packages for developing, deploying, and running scientific applications on HPC platforms.
- E4S provides both source builds and containers of a broad collection of HPC software packages.
- E4S exists to accelerate the development, deployment and use of HPC software, lowering the barriers for HPC users.
- E4S provides containers and turn-key, from-source builds of 80+ popular HPC software packages:
  - MPI: MPICH and OpenMPI
  - Development tools: TAU, HPCToolkit, and PAPI
  - Math libraries: PETSc and Trilinos
  - Data and Viz tools: Adios, HDF5, and Paraview
Extreme-scale Scientific Software Stack (E4S)  
https://e4s.io

- Spack [http://spack.io] is the primary means for software delivery

- SDKs: collection of related ECP ST products where coordination across package teams will improve usability and practices, and foster community growth among teams that develop similar and complimentary capabilities. An SDK involves several products.

- Containers of pre-built binaries of ECP ST products.

- Container runtimes supported
  - Docker: Dockerhub: exascaleproject/sdk:AHM19
  - Charliecloud
  - Shifter
  - Singularity
  - Inception at NCAR

- VirtualBox Open Virtualization Appliance (OVA) image that contains these runtimes

- MPI replacement strategies to use native network interconnect
Spack

- E4S uses the Spack package manager for software delivery
- Spack provides the ability to specify versions of software packages that are and are not interoperable.
- Spack is a build layer for not only E4S software, but also a large collection of software tools and libraries outside of ECP ST.
- Spack supports achieving and maintaining interoperability between ST software packages.
- Acknowledgement: The remaining Spack slides in this presentation are from a talk given by the Spack PI, Todd Gamblin, CASC, LLNL.

- Next: Motivation for Spack!
Scientific software is becoming extremely complex
Even proprietary codes are based on many open source libraries

- Half of this DAG is external (blue); *more* than half of it is open source
- Nearly *all* of it needs to be built specially for HPC to get the best performance
The Exascale Computing Project is building an entire ecosystem

- 15+ applications
- 80+ software packages
- 5+ target architectures/platforms
  - Xeon
  - Power
  - KNL
  - NVIDIA
  - ARM
  - Laptops?
- Up to 7 compilers
  - Intel
  - GCC
  - Clang
  - XL
  - PGI
  - Cray
  - NAG
- 10+ Programming Models
  - OpenMPI
  - MPICH
  - MVAPICH
  - OpenMP
  - CUDA
  - OpenACC
  - Dharma
  - Legion
  - RAJA
  - Kokkos
- 2-3 versions of each package +
  - external dependencies

= up to 1,260,000 combinations!

- Every application has its own stack of dependencies.
- Developers, users, and facilities dedicate (many) FTEs to building & porting.
- Often trade reuse and usability for performance.

We must make it easier to rely on others’ software!
How to install software on a Mac laptop, circa 2013

```
(gluon):~$ port install libelf
```

```bash
libelf-0.7.9 2013-06-06 02:40:48
```

```
libelf-0.7.9 2013-06-06 02:40:48
```

```
libelf-0.7.9 2013-06-06 02:40:48
```

```
libelf-0.7.9 2013-06-06 02:40:48
```

```
libelf-0.7.9 2013-06-06 02:40:48
```

```
libelf-0.7.9 2013-06-06 02:40:48
```

```
libelf-0.7.9 2013-06-06 02:40:48
```

```
libelf-0.7.9 2013-06-06 02:40:48
```

```
libelf-0.7.9 2013-06-06 02:40:48
```

```
libelf-0.7.9 2013-06-06 02:40:48
```

```
libelf-0.7.9 2013-06-06 02:40:48
```

```
libelf-0.7.9 2013-06-06 02:40:48
```

```
libelf-0.7.9 2013-06-06 02:40:48
```

```
libelf-0.7.9 2013-06-06 02:40:48
```

How to install software on a supercomputer

1. Download all 16 tarballs you need
2. Start building!
3. Run code
4. Segfault!?
5. Start over…
What about modules?

• Most supercomputers deploy some form of environment modules
  – TCL modules (dates back to 1995) and Lmod (from TACC) are the most popular

```
$ gcc
-bash: gcc: command not found

$ module load gcc/7.0.1
$ gcc --dumpversion
7.0.1
```

• Modules don’t handle installation!
  – They only modify your environment (things like PATH, LD_LIBRARY_PATH, etc.)

• Someone (likely a team of people) has already installed gcc for you!
  – Also, you can only `module load` the things they’ve installed
What about containers?

• Containers provide a great way to reproduce and distribute an already-built software stack

• Someone needs to build the container!
  – This isn’t trivial
  – Containerized applications still have hundreds of dependencies

• Using the OS package manager inside a container may not be enough
  – Most binaries are built unoptimized
  – Generic binaries, not optimized for specific architectures

• Developing with an OS software stack can be painful
  – Little freedom to choose versions
  – Little freedom to choose compiler options, build options, etc. for packages

We need something more flexible to build the containers
Spack is a flexible package manager for HPC

• How to install Spack (works out of the box):

  $ \text{git clone } \text{https://github.com/spack/spack}$
  $\text{. spack/share/spack/setup-env.sh}$

• How to install a package:

  $\text{spack install hdf5}$

• HDF5 and its dependencies are installed within the Spack directory.

• Unlike typical package managers, Spack can also install many variants of the same build.
  – Different compilers
  – Different MPI implementations
  – Different build options
Spack provides the **spec** syntax to describe custom configurations

- Each expression is a **spec** for a particular configuration
  - Each clause adds a constraint to the spec
  - Constraints are optional – specify only what you need.
  - Customize install on the command line!

- Spec syntax is recursive
  - Full control over the combinatorial build space

```
$ spack install mpileaks
$ spack install mpileaks@3.3
$ spack install mpileaks@3.3 %gcc@4.7.3
$ spack install mpileaks@3.3 %gcc@4.7.3 +threads
$ spack install mpileaks@3.3 cxxflags="-O3 -g3"
$ spack install mpileaks@3.3 os=cnl10 target=haswell
$ spack install mpileaks@3.3 ^mpich@3.2 %gcc@4.9.3
```

- unconstrained
- @ custom version
- % custom compiler
- +/- build option
- setting compiler flags
- setting target for X-compile
- ^ dependency information
Spack has over 3,000 builtin package recipes.
$ spack find

`spack find` shows what is installed

<table>
<thead>
<tr>
<th>Package</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageMagick</td>
<td>6.8.9-10</td>
</tr>
<tr>
<td>SAMRAI</td>
<td>3.9.1</td>
</tr>
<tr>
<td>adept-utils</td>
<td>1.0</td>
</tr>
<tr>
<td>atk</td>
<td>2.14.0</td>
</tr>
<tr>
<td>boost</td>
<td>1.55.0</td>
</tr>
<tr>
<td>cairo</td>
<td>1.14.0</td>
</tr>
<tr>
<td>callpath</td>
<td>1.0.2</td>
</tr>
<tr>
<td>dyninst</td>
<td>8.1.2</td>
</tr>
<tr>
<td>dyninst</td>
<td>8.1.2</td>
</tr>
<tr>
<td>freetype</td>
<td>2.5.3</td>
</tr>
<tr>
<td>gdk-pixbuf</td>
<td>2.31.2</td>
</tr>
<tr>
<td>adept-utils</td>
<td>1.0.1.1</td>
</tr>
<tr>
<td>adept-utils</td>
<td>1.0.1.1</td>
</tr>
<tr>
<td>adept-utils</td>
<td>1.0.1.1</td>
</tr>
<tr>
<td>adept-utils</td>
<td>1.0.1.1</td>
</tr>
<tr>
<td>adept-utils</td>
<td>1.0.1.1</td>
</tr>
<tr>
<td>adept-utils</td>
<td>1.0.1.1</td>
</tr>
<tr>
<td>adept-utils</td>
<td>1.0.1.1</td>
</tr>
<tr>
<td>hwloc</td>
<td>1.9</td>
</tr>
<tr>
<td>hwloc</td>
<td>1.9</td>
</tr>
<tr>
<td>hwloc</td>
<td>1.9</td>
</tr>
<tr>
<td>hwloc</td>
<td>1.9</td>
</tr>
</tbody>
</table>

- All the versions coexist!
  - Multiple versions of same package are ok.
- Packages are installed to automatically find correct dependencies.
- Binaries work *regardless* of user’s environment.
- Spack also generates module files.
  - Don’t have to use them.
The Spack community is growing rapidly

• Spack simplifies HPC software for:
  – Users
  – Developers
  – Cluster installations
  – The largest HPC facilities

• Spack is central to ECP’s software strategy
  – Enable software reuse for developers and users
  – Allow the facilities to consume the entire ECP stack

• The roadmap is packed with new features:
  – Building the ECP software distribution
  – Better workflows for building containers
  – Stacks for facilities
  – Chains for rapid dev workflow
  – Optimized binaries
  – Better dependency resolution

Visit spack.io

github.com/spack/spack

@spackpm
Exascale Platform Preparation

- SDK Exascale platform preparation is focused on interoperable delivery.
- ST products from SDKs are released in the Extreme-scale Scientific Software Stack (E4S) [https://e4s.io].
  - E4S: a community effort to provide open source software packages for developing, deploying, and running scientific applications on HPC platforms
- E4S containers and Spack based builds currently support the following pre-exascale systems:
  - Theta at ALCF (Cray XC).
  - Cori at NERSC (Cray XC).
  - Summit, Sierra, Butte, RZAnsel (IBM Power 9 AC922).
  - Linux x86_64 systems at LANL (Grizzly), Sandia (Voltrino), LLNL (Quartz).
  - Other NSF platforms including Frontera (TACC).
- E4S preparation for future Exascale systems includes testing on AMD and Intel systems.
Integration and Interoperability: E4S

- E4S is released twice a year. Two versions have been released to date and we are planning for a release at SC19. The E4S 0.2 release supports:
  - Containers and turn-key, from-source builds of 80+ popular HPC software packages
  - 37 full release ECP ST products including:
    - MPI: MPICH and OpenMPI
    - Development tools: TAU, HPCToolkit, and PAPI
    - Math libraries: PETSc and Trilinos
    - Data and Viztools: Adios, HDF5, and Paraview
  - Limited access to 10 additional ECP ST products
    - Docker
    - Singularity
    - Shifter
    - Charliecloud
    - Inception
  - Open Virtualization Appliance (OVA) for VirtualBox features Spack, E4S containers, and support for container environments
Integration and Interoperability: E4S on AWS

- E4S AWS public image ami-063e830287b86155c (US-West-2 Oregon) has following container runtimes:
  - Docker
  - Shifter
  - Singularity
  - Charliecloud
- Spack with base PMR components
- E4S full featured Singularity image
  - (exascaleproject/sdk:AHM19)
- Used in ISC-HPC 2019 tutorials
- Used as base image for NASA GEOS-Chem E4S public image
- Resources provided by AWS AI/ML team
Reproducible, Customizable Container Builds & Spack Mirrors

• E4S provides base images and recipes for building Docker containers based on SDKs
  - Git: https://github.com/UO-OACISS/e4s
  - Base images released (September 2019):
    • UBI 7.6 (RHEL Universal Binary Image for container builds) for x86_64
    • Centos 7.6 for x86_64
    • Ubuntu 18.04 for x86_64
    • UBI 7.6 (RHEL) for ppc64le

• E4S provides build caches for Spack for native bare-metal as well as container builds based installation of ST products
  - Build caches: https://oaciss.uoregon.edu/e4s
    • The build cache model can be extended to target platforms, and can be managed by facilities staff when appropriate.
# E4S Build Cache Binaries

## Spack E4S Build Cache

Last updated: 13-Oct-2019 07:43 PST

Click on one of the packages below to see a list of all available variants.

<table>
<thead>
<tr>
<th>1363 Spack binaries in the build cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search</td>
</tr>
</tbody>
</table>

```
adios2@2.4.0    adlx@0.9.2    adol-c@develop    alquimia@xdlk-0.4.0    aml@0.1.0    amrex@18.10.1    amt@1.10.0    argbots@1.0rc1    arpack-ng@3.7.0    autoconf@2.69
automake@1.16.1    axl@0.1.1    bldtopcf@1.0.5    bintutils@2.32    bison@3.0.5    bmi@develop    bolt@1.0rc1    boost@1.70.0    butterflypack@1.0.1    butterflypack@master
bzip2@1.0.8    c-blosc@1.17.0    cairo@1.16.0    caliper@2.0.1    cgns@develop    cinch@develop    cmake@3.15.1    cmake@3.15.3    cuda@10.0.130    cuda@10.1.243
curl@7.63.0    darshan-runtime@3.17    darshan-uuid@3.1.7    dealii@9.0.1    diffutils@3.7    double-conversion@2.0.1    doxygen@1.8.15    dtc@1.1.0
```

**dyninst@10.1.0**

Click on the full spec link to find out more.

<table>
<thead>
<tr>
<th>Link</th>
<th>Arch</th>
<th>OS</th>
<th>Compiler</th>
<th>Created</th>
<th>Full Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Spec</td>
<td>x86_64</td>
<td>centos7</td>
<td>gcc@7.3.0</td>
<td>18-Sep-2019 19:22</td>
<td>cf277611b06f547f5a603f79c25778b7</td>
</tr>
<tr>
<td>Full Spec</td>
<td>x86_64</td>
<td>centos7</td>
<td>gcc@7.3.0</td>
<td>18-Sep-2019 19:22</td>
<td>cf277611b06f547f5a603f79c25778b7</td>
</tr>
<tr>
<td>Full Spec</td>
<td>x86_64</td>
<td>rhel7</td>
<td>gcc@7.3.0</td>
<td>18-Sep-2019 19:22</td>
<td>cf277611b06f547f5a603f79c25778b7</td>
</tr>
<tr>
<td>Full Spec</td>
<td>x86_64</td>
<td>ubuntu18.04</td>
<td>gcc@7.3.0</td>
<td>18-Sep-2019 19:22</td>
<td>cf277611b06f547f5a603f79c25778b7</td>
</tr>
<tr>
<td>Full Spec</td>
<td>x86_64</td>
<td>ubuntu18.04</td>
<td>gcc@7.3.0</td>
<td>18-Sep-2019 19:22</td>
<td>cf277611b06f547f5a603f79c25778b7</td>
</tr>
</tbody>
</table>

elfutils@0.176    emacs@26.2    environment-modules@4.3.0    environment-modules@4.3.1    er@0.0.3    exmcutils@0.5.7    expat@2.2.5    faodel@1.1906.1    findutils@4.6.0
flatcc@0.5.3    fclesi@develop    flex@2.6.4    font-util@1.3.2    fontconfig@2.12.3    fontsproto@2.1.3    freetype@2.9.1    gasnet@2019.3.0    gawk@4.1.4    gcc@7.3.0
glibm@1.18.1    geopm@1.0.0-rc2    gettext@0.19.8.1    git@2.21.0    glib@2.56.3    glm@0.9.7.1    globalarrays@5.7    glproto@1.4.17    gmp@6.1.2    googletest@1.8.1

[https://oaciss.uoregon.edu/e4s](https://oaciss.uoregon.edu/e4s)
Reproducible Container Builds using E4S Base Images

- PMR SDK base image (UBI 7.6) has Spack build cache mirror and GPG key installed.
- Base image has GCC and MPICH configured for MPICH ABI level replacement (with system MPI).
- Customized container build using binaries from E4S Spack build cache for fast deployment.
- No need to rebuild packages from the source code.
- Same recipe for container and native bare-metal builds with Spack!
Reproducible Base Images on Dockerhub

- ecpe4s
- x86_64
- ppc64le
- aarch64
- Centos 7.6
- Ubuntu 18.04
- RHEL/UBI 7.6
Docker Recipes on GitHub

- Base images
- SDKs
- E4S

https://github.com/UO-OACISS/e4s
## Spack Build Caches from E4S Base Images

### x86_64 build cache
- **40 GB on disk**

### IBM Power 9 (ppc64le) build cache
- **2.6 GB on disk**
- **early stages of effort**
- **Initial ARM 64 build cache is underway**

### Table Examples

**Index of /e4s/x86_64/build_cache/linux-rhel7-x86_64/gcc-7.3.0**

<table>
<thead>
<tr>
<th>Name</th>
<th>Last Modified</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent Directory</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e4s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x86_64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>build_cache</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>linux-rhel7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x86_64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gcc-7.3.0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Index of /e4s/ppc64le/build_cache/linux-centos7/ppc64le/gcc-7.3.0**

<table>
<thead>
<tr>
<th>Name</th>
<th>Last Modified</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent Directory</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e4s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ppc64le</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>build_cache</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>linux-centos7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ppc64le</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gcc-7.3.0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Docker container of E4S

% docker pull exascaleproject/e4s_x86_64:1.0

• Using USB stick or images from https://e4s.io:
  • % unxz -c e4s_x86_64.xz | docker load
    % docker images

• Mount home directory:
  % docker -i -v $HOME:$HOME -t exascaleproject/e4s_x86_64:1.0 /bin/bash
  % which spack
  % cp -r /usr/local/packages/ecp/demo . ; cd demo; cat README
Using Shifter at NCSA BlueWaters

Load shifter module and E4S image on the compute node

- Allocate a node
  - `% qsub -l -l nodes=1:ppn=32 -l walltime=01:15:00 -l gres=shifter16`

- Load the shifter module
  - `% module load shifter`

- Pull the image (once)
  - `% shifterimg pull exascaleproject/sdk:AHM19`

- Launch the image
  - `% shifter --image=exascaleproject/sdk:AHM19 -- /bin/bash`
  - `% unset CRAYPE_VERSION; . /etc/bashrc`
  - `% spack find`
E4S v1.0 Release (50 Products):

on Dockerhub
Extreme-scale Scientific Software Stack (E4S)
https://e4s.io

Singularity> cd `spack location -i trilinos`/lib
Singularity> ls *.so*1
libamesos2.so.12.12.1
libamesos.so.12.12.1
libanaszietepetra.so.12.12.1
libanasazi.so.12.12.1
libanasziptpetra.so.12.12.1
libaprepro_lib.so.12.12.1
libaztecoo.so.12.12.1
libbelosepetra.so.12.12.1
libbelos.so.12.12.1
libbelostpetra.so.12.12.1
libchaco.so.12.12.1
libepetraext.so.12.12.1
libepetra.so.12.12.1
libexodus_for.so.12.12.1
libexodus.so.12.12.1
libexolviv2for32.so.12.12.1
libgaleri-epetra.so.12.12.1
libgaleri-xpetra.so.12.12.1
libgtest.so.12.12.1
libifpack2-adapters.so.12.12.1
libifpack2.so.12.12.1
libifpack.so.12.12.1
libIeexo_fac.so.12.12.1
libIeox.so.12.12.1
libifx.so.12.12.1
libIogn.so.12.12.1
libIohb.so.12.12.1
libinfo_info_lib.so.12.12.1

libIInit.so.12.12.1
libIopg.so.12.12.1
libIopx.so.12.12.1
liblooss.so.12.12.1
libIotr.so.12.12.1
libIovs.so.12.12.1
libIorroppia.so.12.12.1
libIkkkflags_algorithms.so.12.12.1
libIkkkflags_containers.so.12.12.1
libIkkkflags_core.so.12.12.1
libIkkkflags_kernels.so.12.12.1
libIkkkflags_kernelsqr.so.12.12.1
liblocacpetra.so.12.12.1
liblocalapack.so.12.12.1
liblocasc.so.12.12.1
liblocathyra.so.12.12.1
libmapvarlib.so.12.12.1
libml.so.12.12.1
libcModelLaplace.so.12.12.1
libmuelu-adapters.so.12.12.1
libmuelu-interface.so.12.12.1
libmuelu.so.12.12.1
libmesis.so.12.12.1
libmoxepetra.so.12.12.1
libmoxlapack.so.12.12.1
libmox.so.12.12.1
libmugen_extras.so.12.12.1
libmugen.so.12.12.1
librtop.so.12.12.1
libsacado.so.12.12.1
libshyhu.so.12.12.1
libshyexpreval.so.12.12.1
libshykeysearch.so.12.12.1
libshykeytopology.so.12.12.1
libshykey_transferimpl.so.12.12.1
libshykey_util_diag.so.12.12.1
libshykey_util_env.so.12.12.1
libshykey_util_parallel.so.12.12.1
libshykey_util_registry.so.12.12.1
libshykey_util_usecases.so.12.12.1
libshykey_util_util.so.12.12.1
libshykeysubkhoisosmesos.so.12.12.1
libshykeysubkhoisipack.so.12.12.1
libshykeysubkhoisimsl.so.12.12.1
libshykeysubkhoisimzos.so.12.12.1
libshykeysubkhoisaztec.so.12.12.1
libshykeysubkhoisbelos.so.12.12.1
libshykeysubkhoisifpack.so.12.12.1
libshykeysubkhoisimsl.so.12.12.1
libsiceres.so.12.12.1
libsicereslib.so.12.12.1
libsicereslib_c.so.12.12.1
libsicereslib_cpp.so.12.12.1
libsicereslib_c.so.12.12.1
libsicereslibso.so.12.12.1
libsicereslibso2.so.12.12.1
libsicereslibso3.so.12.12.1
libsicereslibso4.so.12.12.1
libsicereslibso5.so.12.12.1
libsupermatrix.so.12.12.1
libsupermatrix-c.so.12.12.1
libsupermatrix-cpp.so.12.12.1
libsupermatrix.so.12.12.1
libsupermatrix-c.so.12.12.1
libsupermatrix-cpp.so.12.12.1
libsupermatrix.so.12.12.1
libsupermatrix-c.so.12.12.1
libsupermatrix-cpp.so.12.12.1
libsupermatrix.so.12.12.1
libsupermatrix-c.so.12.12.1
libsupermatrix-cpp.so.12.12.1
libsupermatrix.so.12.12.1
libsupermatrix-c.so.12.12.1
libsupermatrix-cpp.so.12.12.1
libsupermatrix.so.12.12.1
libsupermatrix-c.so.12.12.1
libsupermatrix-cpp.so.12.12.1
libsupermatrix.so.12.12.1
libsupermatrix-c.so.12.12.1
libsupermatrix-cpp.so.12.12.1
libsupermatrix.so.12.12.1
libsupermatrix-c.so.12.12.1
libsupermatrix-cpp.so.12.12.1
Running MPI applications on other systems

- Applications built with MPI in the E4S container can be replaced by the system MPI!
- This allows fast inter-node communication using the native interconnect.
- Application and data are external to the E4S container.
- Programming models, compilers, runtime libraries, and tools are inside the container.
- We can replace MPI using the MPICH ABI compatibility layer.
- Goal: Build an MPI binary once and run it un-modified on all HPC Linux x86_64 clusters!
Using E4S on NCSA BlueWaters and replacing MPI

**Step 1: Allocate a node with the E4S image**

- `qsub -l -l nodes=2:ppn=32 -l walltime=01:15:00 -l gres=shifter16 -v UDI=exascaleproject/sdk:AHM19`
- This allocates a single node for 1:15h
- Specifies the use of Shifter as the container environment
- The image is `exascaleproject/sdk:AHM19`
- This image was pulled on a compute node previously using:
  - `%module load shifter; shifterimg pull exascaleproject/sdk:AHM19`
- After this `qsub` step, we can now launch the job using `aprun`
Using E4S on NCSA BlueWaters Replacing MPI

**Step 2: Create a file called ~/shifter_mpi.sh**

```bash
% cat ~/shifter_mpi.sh
#!/bin/bash
# set up LD_LIBRARY_PATH
for dir in $(echo $CRAY_LD_LIBRARY_PATH:/opt/cray/wlm_detect/default/lib64 | tr ':' ' ')
do
    realpath=$(readlink -f "$dir")
    if [[ -z $LD_LIBRARY_PATH ]]
        then
            eval 'export LD_LIBRARY_PATH=/dsl'$realpath
    else
            eval 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/dsl'$realpath
    fi
done
```
Replacing MPI using cray-mpich-abi package

Step 3: Source this ~/shifter_mpi and setup LD_LIBRARY_PATH

% cat run.sh
#!/bin/bash
export CRAY_ROOTFS=SHIFTER
module load shifter
module unload PrgEnv-cray # or any other PrgEnv module currently loaded
module load PrgEnv-gnu # or PrgEnv-intel
module unload cce
module unload cray-mpich
module load cray-mpich-abi
export LD_LIBRARY_PATH=$CRAY_MPICH_DIR/lib:$LD_LIBRARY_PATH
source ~/shifter_mpi.sh
export LD_LIBRARY_PATH=/usr/local/packages/ecp/spack/opt/spack/linux-centos7-x86_64/gcc-7.3.0/hwloc-1.11.9-7xxgxbg65an7zmrztfcuu3hs73puj6v3/lib:$LD_LIBRARY_PATH
export OMP_NUM_THREADS=2
aprun -b -n 64 -- ./lulesh.host -i 100
Replacing MPI using cray-mpich-abi package

Step 4: run the example

% ./run.sh
Running problem size 30^3 per domain until completion
Num processors: 64
Num threads: 2
Total number of elements: 1728000

... Run completed:

  Problem size = 30
  MPI tasks = 64
  Iteration count = 100
  Final Origin Energy = 8.465100e+07
  Testing Plane 0 of Energy Array on rank 0:
    MaxAbsDiff = 7.916242e-09
    TotalAbsDiff = 3.030168e-08
    MaxRelDiff = 1.224484e-13

Elapsed time = 16.58 (s)
Grind time (us/z/c) = 6.1409471 (per dom) (0.095952298 overall)
FOM = 10421.845 (z/s)
Elapsed time = 16.58 (s)
Grind time (us/z/c) = 6.0131382 (per dom) (0.22270882 overall)
FOM = 4490.1679 (z/s)

Application 81575093 resources: utime ~442s, stime ~20s, Rss ~45404, inblocks ~9110
Singularity on Theta at ALCF

% qsub -A ECP_SDK -t 30 -n 2 -q debug-cache-quad -l
% /projects/ECP_SDK/tutorial/run_job.sh

module swap PrgEnv-intel PrgEnv-gnu
module swap cray-mpich cray-mpich-abi
export SINGULARITYENV_LIBWLM_DETECT=/opt/cray/wlm_detect/
1.3.2-6.0.6.0_3.8__g388ccd5.ari/lib64
aprun -n 16 -N 8 singularity exec -H $HOME -B /projects/ECP_SDK:/projects/ECP_SDK:ro
-B /opt:/opt:ro -B /var:/var:ro /projects/ECP_SDK/containers/singularity/ecp.simg bash
-c 'unset CRAYPE_VERSION; source /usr/local/packages/ecp/misc/bashrc; spack load
trilinos hypre parmetis hdf5 metis openblas superlu zlib netcdf matio boost@1.66.0 scalapack
suite-sparse tau ;spack unload openmpi mpich ; export
Singularity on Quartz at LLNL

MVAPICH2 needs /lib. Mount it as /hostlib64 and add it to LD_LIBRARY_PATH

% sallocate -N 2
% srun -n 4 -c 2 singularity exec -B /lib64:/hostlib64 -B $SLURM_SUBMIT_DIR:$SLURM_SUBMIT_DIR -B /usr/tce:/usr/tce ./ecp.simg /bin/bash -c '. /etc/bashrc ; spack load trilinos hypre parmetis hdf5 metis openblas superlu zlib netcdf matio boost@1.66.0 scalapack suite-sparse tau; spack unload openmpi mpich; export LD_LIBRARY_PATH=/usr/tce/packages/mvapich2/mvapich2-2.2-intel-18.0.1/lib:$LD_LIBRARY_PATH:/hostlib64; ./Zoltan'
Replacing MPI with Shifter on Cori.nersc.gov

% shifterimg images
exascaleproject/sdk:AHM19 ...

% To replace MPI with system MPI:
# salloc -N 2 -q interactive -t 00:30:00 --image=exascaleproject/sdk:AHM19 -C haswell -L SCRATCH
# ~sameer/run_shifter.sh
# cat ~/run_shifter.sh

srun -n 32 shifter -- /bin/bash -c 'unset CRAYPE_VERSION; . /etc/bashrc ; spack load trilinos hypre parmetis hdf5metis openblas superlu zlib netcdf matio boost@1.66.0 scalapack suite-sparse tau; spack unload openmpi mpich; ./Zoltan'
E4S VirtualBox OVA image

Contains all four container runtimes and the E4S Singularity image!

- Docker
- Singularity
- Shifter
- Charliecloud
E4S image on Amazon AWS

Contains all four container runtimes and the E4S Singularity image!

- AWS AMI ID (Oregon, us-west-2 region):
  - ami-063e830287b86155c

- Royalty free, public image with HPC, AI, and 4 container runtimes

- Launch EC2 instance with this AMI
  - Login: tutorial
  - Password: ****
Future work, issues...

- Increasing the number of ST packages in E4S
- Porting to IBM and ARM platforms
- Support for GPUs and visualization tools
- Addition of CI testing
- Facility deployment
- Scalable startup with full-featured “Supercontainers”
- Improving the launch of MPI applications
E4S: How to get involved

● E4S BoF at SC19
  ● Tuesday, Nov. 19, 12:15pm – 1:15pm, Room 405-406-407

● CANOPIE-HPC Workshop at SC19
  ● 1st Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC
  ● Monday, Nov. 18, 2019, 2pm – 5:30pm, Room 704-706
  ● https://canopie-hpc.nersc.gov/
Acknowledgment

“This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering, and early testbed platforms, in support of the nation’s exascale computing imperative.”