Building Software Environments for Research Computing Clusters

Mark Howison, Aaron Shen, Andrew Loomis

Brown University Center for Computation & Visualization
Challenges [HPC] sysadmins face

Maintain multiple software installations on a shared filesystem supporting hundreds of users

-Typical system:

```
./configure --disable-threads --with-petsc=/gpfs/runtime/opt/petsc/3.0.0-p12 --with-petsc-arch=linux-gnu-
cxx-opt --with-umfpack --with-trilinos=/gpfs/runtime/opt/trilinos/10.2.2 --with-metis=/gpfs/runtime/opt/
metis/4.0.1 --with-blas=goto2 --with-lapack=goto2 --with-p4est=/gpfs/runtime/opt/dealii/7.0.0/p4est --with-
mumps=/gpfs/runtime/opt/mumps/4.9.2 --with-scalapack=/gpfs/runtime/opt/gotoblas2/1.13/lib --with-blacs=/
gpfs/runtime/opt/gotoblas2/1.13/src/BLACS --enable-mpi CC=mpicc CXX=mpiCC LDFLAGS=-L/gpfs/runtime/opt/
gotoblas2/1.13/lib
```

-HPC:

Multiple hardware setups
Failed to load the JNI shared library "C:\Program Files (x86)\Java\jre6\bin\client\jvm.dll".
Challenges [HPC] sysadmins face

Maintain multiple software installations on a shared filesystem supporting hundreds of users
- Typical system:

```
./configure --disable-threads --with-petsc=/gpfs/runtime/opt/petsc/3.0.0-p12 --with-petsc-arch=linux-gnu-
cxx-opt --with-umfpack --with-trilinos=/gpfs/runtime/opt/trilinos/10.2.2 --with-metis=/gpfs/runtime/opt/
metis/4.0.1 --with-blas=goto2 --with-lapack=goto2 --with-p4est=/gpfs/runtime/opt/dealii/7.0.0/p4est --with-
mumps=/gpfs/runtime/opt/mumps/4.9.2 --with-scalapack=/gpfs/runtime/opt/gotoblas2/1.13/lib --with-blacs=/
gpfs/runtime/opt/gotoblas2/1.13/src/BLACS --enable-mpi CC=mpicc CXX=mpiCC LDFLAGS=-L/gpfs/runtime/opt/
gotoblas2/1.13/lib
```

- HPC:

Building/optimizing software
- Many different dependencies

Multiple hardware setups
Pymodules: Our enhanced Environment Modules* system designed to mitigate the above issues.

Pymodules is...

- User-friendly
- Flexible
- Fast

Without modules

```
PYTHON_ROOT=/gpfs/runtime/opt/python/2.7.3
export PATH=$PYTHON_ROOT/bin:$PATH
export LD_LIBRARY_PATH=$PYTHON_ROOT/lib:$LD_LIBRARY_PATH
export MANPATH=$PYTHON_ROOT/share/man:$PYTHON_ROOT/man:$MANPATH
export PKG_CONFIG_PATH=$PYTHON_ROOT/lib/pkgconfig
```

With modules

```
module load python/2.7.3
```
Pymodules relies on modulefiles

**Modulefiles**: small, INI-style text files for each application specifying changes to be made to the environment, for each version (example: Boost C++ modulefile)

```
[DEFAULT]

brief = Boost - The Boost C++ Libraries
url = http://www.boost.org/
category = libraries

prepend CPATH = %\(rootdir\)s/include
prepend LIBRARY_PATH = %\(rootdir\)s/lib
prepend LD_RUN_PATH = %\(rootdir\)s/lib

set BOOST = -L%\(rootdir\)s/lib
set BOOST_DIR = %\(rootdir\)s
set BOOST_ROOT = %\(rootdir\)s

[1.49.0]
default = true

[1.52.0]
prepend CPATH= %\(rootdir\)s/src/boost_1_52_0

[1.40.0]
```

metadata

common environment settings

all versions present on system
Using modules in the software installation process

Build time issues
Software can depend on (many) other softwares in the build process, leading to very complicated builds.

Solution:
Dependency modules set LIBRARY_PATH and CPATH.

```bash
```
Using modules in the software installation process, cont.

**Runtime issues**
Modules can depend on other modules’ libraries @ runtime

**Goal:** Make things simple as possible for the user

<table>
<thead>
<tr>
<th>Make user load dependency modules (LD_LIBRARY_PATH)</th>
<th>User (@ Runtime)</th>
<th>Admin (@ build time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>module load deal.ii</td>
<td>module load deal.ii</td>
<td></td>
</tr>
<tr>
<td>module load openmpi</td>
<td>module load openmpi</td>
<td></td>
</tr>
<tr>
<td>module load petsc</td>
<td>module load petsc</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dependencies to module X set LD_RUN_PATH</th>
<th>User (@ Runtime)</th>
<th>Admin (@ build time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>module load deal.ii</td>
<td>module load deal.ii</td>
<td></td>
</tr>
<tr>
<td>module load openmpi</td>
<td>module load openmpi</td>
<td></td>
</tr>
<tr>
<td>module load petsc</td>
<td>module load petsc</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Using modules to organize software inventory

Usability issues
Finding the application you want out of 500+ options is a challenge.

Use a database, not files, to keep track of all installed software.

Runtime of “module avail” command:

<table>
<thead>
<tr>
<th>Files</th>
<th>Scales up w/# of software packages installed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Constant</td>
</tr>
</tbody>
</table>
Using modules to organize software inventory

Modules offer the following features:
- Search by package name/version (can also search by regex)
- Tab completion
- Store metadata: category of software...date last loaded, etc

[user@ccv ~]$ module avail
~~~~~~~~~~~~category: Math ~~~~~~~~~~
atlas/3.10.1  sage/5.11
~~~~~~~~~~~~category: bio ~~~~~~~~~~~
abyss/1.3.4                   macs/1.4.2
agalma/0.3.0                  macs/2.0.10
...
Using modules for performance optimization

Performance issues
- Heterogeneous hardware
  - AMD vs. Intel
  - SIMD instruction set
  - Processor Model

Solution:
- Optimized modules for different architectures

Example:
LLCBench benchmark linked against Intel and AMD linear algebra routines…
Comparison of BLAS performance under ACML and MKL. AMD machine, 4 GB memory, 1 CPU core.

- Red line: MKL
- Blue line: ACML

Graph shows the performance in MFlops/sec against problem size.
Pymodules vs other tools

Environment Modules (original):
- Tcl syntax
- Uses files to store module information

CDE (Guo):
- Build once, run everywhere
- Not optimized for specific hardware or libraries

Traditional package managers
- Not good at managing multiple versions of same software.
Conclusion

- Software modules can maintain multiple software installations on shared system.
- Can improve ease of software installation and use, performance gains

-Pymodules

Further work: Integrating the install process

-Pymodules is available at
http://www.bitbucket.org/mhowison/pymodules