HPC Application Deployment through Environment Modules and NFS

Stephen R. Wheat

Professor of Computer Science

Director of the Oral Roberts University Research Computing and Analytics Facility

Motivation

- **Applications for one or more users … typically more than one.**
- **Desire to install applications without rebooting the compute nodes**
- **Install updated versions of the application without interfering with the use of prior versions.**
- **Install complex applications with “minimal” effort.**

Assumptions

- **OS Dependencies**
  - **Some applications may need additional libraries or services to be available, some of which must be installed in standard locations**
    - **For example: OpenSSL.**
  - **For this talk, it is assumed that all such dependencies are already installed in the OS images for the compute nodes.**
- **CPU Architectures**
  - **It is assumed that all compute nodes are of the same CPU architecture and generation.**
  - **It is assumed that the “build” node is of the same architecture/generation as the compute node**

Introduction to Environment Modules

- **The Environment Modules (or simply “modules”) environment provides for the deployment and management of applications**
- **Users access an application by “loading” the appropriate module**
  - **The loading action transparently modifies the user’s PATH and LD_LIBRARY_PATH (and more) environment variables such that the application’s executable(s) and libraries are made accessible.**
- **Users run the application through direct invocation of the application name, without providing a full path to the application.**
- **This talk utilizes the Lmod modules management system**

Live Example of Usage of Modules

- **Finding which modules are available**
- **Loading one or more modules**
- **Running the application**
- **Using modules in sbatch scripts**
- **Unloading modules**
- **Purging all modules**

Modules architecture

- **Modules are typically installed in an NFS shared file system. At ORU, we deploy them in a shared /opt.**
- **Modules directory: There can be more than one, we'll focus on /opt/modules**
  - **There can be more than one collection of modules, each a directory within /opt/modules. At ORU, we have these and more:**
    - **For those installed via Easybuild (more on Easybuild later)**
      - **all – the applications**
      - **tools – tools such as compilers**
    - **For the intel OneAPI modules, we have:**
      - **/opt/intel/oneapi**
- **Applications directory: /opt/software**
  - **This is where the applications executables and libraries are installed**
  - **For example: /opt/software/BLAST+**

Installing the Applications Environment - Lmod

- There are several ways to install Lmod. For consistency and simplicity, we use the lmod package provided in the OpenHPC repo.
- Use the following commands

```
dnf -y install http://repos.openhpc.community/OpenHPC/2/EL_8/x86_64/ohpc-release-2-1.el8.x86_64.rpm

dnf -y install dnf-plugins-core

dnf config-manager --set-enabled powertools

dnf -y install lmod-ohpc
```

- The Lmod package has been installed, but to “see” it, log out and back in again. Then issue “which modules” to verify it is there.

Prepare for EasyBuild

- **We will configure EasyBuild to install applications in /opt/modules/all and /opt/modules/tools**
- **Create an easybuild user that belongs to group root, choose an ID as appropriate**

```
useradd -u 1003 -d /home/easybuild easybuild -g root
```

- **Create the directories for easybuild’s apps**

```
mkdir –p /opt/modules/all /opt/modules/tools /opt/software
chown –R easybuild /opt/modules /opt/software
```

- **Easybuild will need python3**

```
dnf -y install python3
```

Further prep for Easybuild

- **These may be needed for both building and running certain applications. If needed for running, they will need to be also installed in the OS image of the compute nodes.**

```
dnf –y install openssl
dnf –y groupinstall "Development Tools"
dnf -y install openssl-devel
dnf –y install rdma-core-devel
```

- **Now edit /etc/profile.d/lmod.sh to know about the EasyBuild directories. Add :/opt/modules/all:/opt/modules/tools to the colon-separated MODULEPATH variable exports.**

Installing EasyBuild, part 1

- **Somewhat following the "install EasyBuild with EasyBuild" section of https://docs.easybuild.io/installation/#more_pip_env_EB_VERBOSE**
- **Log in as easybuild**

```
su - easybuild
```

- **Install a temporary copy of EasyBuild**

```
export EB_TMPDIR=/tmp/$USER/eb_tmp
python3 -m pip install --ignore-installed --prefix $EB_TMPDIR easybuild
```

Installing Easybuild, part 2

- **As user easybuild , create a config file.**

```
mkdir -p ~/.config/easybuild
```

- **Put the following into .config/easybuild/config.cfg**

```
[basic]
# always enable logging to stdout
#logtostdout = true
[config]
# use Lmod as modules tool
modules-tool: Lmod
# use different default installation path
prefix = .local/easybuild/
installpath = /opt
installpath-modules = /opt/modules
```

Installing Easybuild, part 3

- **Set up to use the temporary version of EB**

```
export PATH=$EB_TMPDIR/bin:$PATH
export PYTHONPATH=$(/bin/ls -rtd -1 $EB_TMPDIR/lib*/python*/site-
packages | tail -1):$PYTHONPATH
export EB_PYTHON=python3
```

- **Install the production version of EasyBuild**

```
eb --install-latest-eb-release
```

- **Verify that it was installed**

```
ls /opt/modules/EasyBuild
```

Installing Easybuild, part 4

- **Now log out and log back in as easybuild**
- **Then issue the command: module avail**
- **It should show EasyBuild/4.7.2 (D) in /opt/modules/all**

Useful module commands

- **module list – shows the currently “loaded” modules in your environment**
- **module purge – Removes all modules from your environment**
- **module avail – Displays all the modules available**
- **module load xyz – Causes the “xyz” module to be loaded**

Using EasyBuild to build applications

- **su – easybuild – get into the easybuild persona**
- **module purge – unload all default loaded modules**
- **module load EasyBuild – load the EasyBuild module**
- **Let’s build and install the latest version of GCC; first find a listing of the EB configurations. With the following command, we will find GCC-13.1.0 is the latest**

```
eb -S GCC | grep \/GCC\/
```

- **Issue the following command to build and install that version**

```
eb GCC-13.1.0.eb --robot
```

- **The --robot tells it to build any/all the packages upon which it depends.**
- **This will take a while. And it will populate several additional modules.**

- **To build the latest version of OpenFoam:**

```
eb OpenFOAM-v2206-foss-2022a.eb --robot
```