Spack

Course: Software Development in Science

Teffy Sam

18.05.2020

Agenda

1. Introduction to package managers
2. Comparisson between different approaches
3. What is unique to Spack
4. Spack demo

What is a package manager?

- Packages are held in a central repository. Downloaded and installed onto a user system from there
- Packages were previously extracted in the 90s from .tgz files and installed from source
- Automation was not available previously
- Software packages have multple dependencies
- Conflicts arise between dependency versions

- On a supercomputer, software is normally built from source
- .tgz tarballs are downloaded and then installed with *make install* or *cmake*
- The software then needs to be linked with libraries and dependencies should be acquired too

Popular package managers

Package managers are maintained by a group of people who ensure everything works together

- Debian – Apt -> dpkg installs .deb binaries
- Arch Linux – Pacman/AUR
- FreeBSD – pkg/Ports fetches source and installs with make
- MacOS – MacPorts
- Flatpak installs into sandboxed environments
- Chocolatey on windows -> NuGet is the back end

Problems with OS package managers

- Poor support for versions and scientific applications
- Not flexible with installs thereby limiting users

HPC package manager

- Why do we need a HPC package manager?
- Problem of multiple platforms
- Hardware upgrades leading to incompatiblity
- Spack is not just limited to HPC

HPC package manager

- How is spack different from Environment Modules?
- Packages still have to built from source for Modules
- They only *manage* your environment
- How is spack different from Containers?

Comparissons

- Docker and Virtual Machines enclose the environment of an application from others
- Containerization is essential to ensure software works even with portability
- But it still requires to be built and configured from source
- Docker for users on HPC machines are limited by root permissions for the Docker Daemon
- Spack combines docker support with its own environement feature enabling good portability and ease of use

Spack

- Spack is non-destructive: new installs do no break current installs, allowing for multiple versions
- Directories of installed packages are hashed with its own prefix
- The prefix ensures there is no mix up when loading libraries
- Each configuration installed can be distinguised from other versions with respective hashes

Spack

- Spack is useful when setting up a new HPC system or upgrading older hardware.
- Installation of applications is a streamlined process
- Spack packages are templated giving user more freedom to configure

Spack users

Spack is used by and is contributed to by many supercomputing centres

1. Riken – Japan
2. LLNL(developed) - California
3. Max Planck – Hamburg
4. Fermilab - CERN

Easybuild

- This is spack's closest competitor
- But the templating is absent here
- The Easybuild framework consists of python modules and packages
- After installing with easybuild, packages can be loaded with Module

Spack demo: My story

- Binary built is not compatible with other packages if these packages were not built using the same compiler.
- FLASH3D <***must use*** gcc-4.8; specifically libgfortran.so.3>
    1. Netcdf-fortran
        - Netcdf-c
            1. HD5
            2. Zlib
    2. Lapack
    3. Blas

    One problematic package = segmentation fault :(

- Arch Linux by default installs latest gcc with Pacman
- gcc48 is available in the AUR
- But other dependencies of FLASH3D are limited to already compiled binaries (compiled with *latest* gcc)
- This breaks everything, cdo, ncview etc
- Final option: compile each package from source

- Using spack
  - spack install netcdf-fortran %gcc@4.8.5
  - spack install openblas %gcc@4.8.5
  - spack find --paths

FLASH3D binary works!

Setup spack

- Install spack from github
- Source setup-env.sh to enable spack/module support
- Simple Install
- Complex Install
- Runs minor tests to ensure binary works

```
spack install <package>
spack install <package> ^<dependency>
spack install <package> %<compiler>@<version>
```

- Spack environments
- Portable with spack.yaml

References

- T. Gamblin et al. The Spack package manager: Bringing order to HPC software chaos [PDF slides] https://tgamblin.github.io/files/Gamblin-Spack-SC15-Talk.pdf
- Spack documentation. http://spack.readthedocs.io/en/latest