SOFTWARE PACKAGES IN HPC WITH SPACK AND EASYBUILD

William Lucas

17 June 2020

Overview

- Building and managing software
- Spack
  - Description
  - Usage
- EasyBuild
  - Description
  - Usage
- Brief comparison between the two
- Disclaimer – this isn’t a tutorial, but rather an introduction and basic primer.

Issues with software on HPC

- Aside from user-built software, often a lot of centrally available software provided through environment modules
- Performance is key so this will be built from source
- Any given version will have its own set of dependencies
- And may prove difficult to build
- And you will very possibly want to provide multiple versions

Issues with software on HPC

- Consequences:
  - Potentially a lot of time scratching your head
  - And if you need to build it again but didn't make notes, more time lost scratching your head
- Keep notes!
  - https://github.com/hpc-uk/build-instructions
- Or you could use a build system:
  - Spack
  - EasyBuild
- Automate the build process, resolve dependencies

Spack overview

- Created by Todd Gamblin at LLNL, + community support
- Written in Python (requires 2.6 or 2.7 or 3.5 to 3.8)
- Install simply by cloning from GitHub
  - https://github.com/spack/spack
- Can be installed and run site-wide or locally
- Just source the setup script `$SPACK_ROOT/share/spack/setup-env.sh`
- Then it's available on command line via `spack`

Installing HDF5 with Spack

- Once Spack is available on the command line, run:

```
      > spack install hdf5
```

- What happens?
- On my OpenSUSE VM – a lot!

```
Input spec
--------------------------------
hdf5

Concretized
--------------------------------
hdf5@1.10.6%gcc@7.5.0~cxx~debug~fortran~hl+mpi+pic+shared~szip~threadsafe api=none arch=linux-
opensuse_leap15-sandybridge
^openmpi@3.1.6%gcc@7.5.0~atomics~cuda~cxx~cxx_exceptions+gpfs~java~legacylaunchers~memchecker~pmi+runpath
~sqlite3+static~thread_multiple+vt fabrics=none schedulers=none arch=linux-opensuse_leap15-sandybridge
        ^hwloc@1.11.11%gcc@7.5.0~cairo~cuda~gl~libudev+libxml2~netloc~nvml+pci+shared arch=linux-
opensuse_leap15-sandybridge
            ^libpciaccess@0.13.5%gcc@7.5.0 arch=linux-opensuse_leap15-sandybridge
                 ^m4@1.4.18%gcc@7.5.0+sigsegv
patches=3877ab548f88597ab2327a2230ee048d2d07ace1062efe81fc92e91b7f39cd00 arch=linux-opensuse_leap15-
sandybridge
```

Installing HDF5 with Spack

- What we're seeing is the Spack concretizer at work.
- We provided a barebones request of 'install HDF5'.
- The Spack concretizer takes that and fills it out:
  - Which version of HDF5?
  - Built with which version of which compiler?
  - Which HDF5 build options should be used?
  - What are the dependencies for HDF5?
    - And which versions and options should be used for those?
- Once Spack knows exactly what it should be building, it goes ahead and does so.
  - Of course this might take some time.

Installing HDF5 with Spack

- We can make our own choices rather than just let the concretizer choose ‘best’ options for us by expanding on the install spec.
- Add as postfixes to the package:
    - Package version with ‘@’
    - Compiler with ‘%’
    - Enable options with ‘+’, disable with ‘~’
    - Do the same for dependencies by including these after a ‘^’.
- `hdf5@1.10.6%gcc@6.3.0+fortran^openmpi@4.0.3`
- This will install HDF5 1.10.6 with optional Fortran support, building with GCC 6.3.0, and using OpenMPI 4.0.3.

Spack package installation tips

- When you settle on what you would like to install, you can check 1) that there aren't conflicts in the spec, and 2) what the full set of concretized packages will look like by running `spack spec <install spec>`
- Search the available packages with `spack list <search term>`
- Display all packages currently installed by Spack with `spack find` or search installed packages with `spack find <search term>`
- Uninstall with `spack uninstall <package name>`

Spack package installation tips

- It can be quite confusing if a lot of similar packages are installed (and this can easily end up being the case)
- Every installed package receives a unique hash – you can see these with `spack find -L` (or a truncated hash with lowercase `-l`)
- You can use that hash in any command where you want to refer to that package by prepending it with '/'
  - e.g. on my VM 'wgh7dtv' is the first part of an OpenMPI installation
  - `spack install hdf5^/wgh7dtv`
  - `spack uninstall /wgh7dtv`

What else can Spack do?

- Generation of modulefiles for installed packages
- Full package management capabilities
- Organise packages into Spack environments
  - Allows for software to be loaded and unloaded together
  - Automate build and installation of software matrices
    - E.g. different versions of different software with different versions of different compilers or dependencies
- Of course it's very configurable
  - Preferred versions, preferred providers
  - Where to install
  - Which modulefiles to generate, where, and how to name them

EasyBuild overview

- Maintainers and package support from many across the world, lead developer Kenneth Hoste at Ghent University.
- Written in Python (requires Python 2.6 or 2.7 or >= 3.5) and requires modules to be available
- Can be installed with `pip` or `easy_install` – a bootstrapping script is recommended and makes the process very easy.
- Install locally or site-wide
- The script provides a handy module which can be loaded.
- Once on the `PATH`, run EasyBuild via the `eb` command.

EasyBuild overview

- Separated into three parts - bootstrapping installs all three:
  - The EasyBuild framework itself
  - The easyblocks which provide instructions on how to perform each install technique such as `./configure`, `make`, `make install`
  - The easyconfigs (`.eb` extension) which provide the information necessary to install a given piece of software under a given toolchain: code URLs, checksums, dependencies, build options…
- Toolchains (themselves versioned) include the major tools needed to build software
  - `foss` includes GCC, OpenMPI, OpenBLAS, ScaLAPACK, FFTW
  - `intel` includes Intel compilers, MPI, MKL

Installing HDF5 with EasyBuild

- To install something we must choose an easyconfig to use
- Run `eb -S '^HDF5'` to search for available easyconfigs
  - Search for regex 'begins with HDF5' rather than straight 'HDF5' to avoid other easyconfigs with it in the name such as h5py.
  - Select the HDF5 version we want to be built by the toolchain version we want.
  - Let's go with HDF5 1.8.20, using foss 2018a (GCC 6.4.0, OpenMPI 2.1.2, OpenBLAS 0.2.20, ScaLAPACK 2.0.2 and FFTW 3.3.7)
- To install, just run EasyBuild followed by the name of the easyconfig: `eb HDF5-1.8.20-foss-2018a.eb`
- This will use the information in the easyconfig to build using the EB_HDF5 easyblock.

Installing HDF5 with EasyBuild

- Each easyconfig details the dependencies required.
- Pass EasyBuild the `--robot` (or `-r`) option for it to create a tree of dependencies and build and install as needed
- Can also use `--dry-run` (or `-D`) to see what would be done without actually performing the installation.
- The output of `eb HDF5-1.8.20-foss-2018a.eb -Dr`

```
== temporary log file in case of crash /tmp/eb-8kuq4gyl/easybuild-t0rmc0n9.log
== found valid index for /home/vagrant/EasyBuild/software/EasyBuild/4.2.1/easybuild/easyconfigs, so using
it...
Dry run: printing build status of easyconfigs and dependencies
CFGS=/home/vagrant/EasyBuild/software/EasyBuild/4.2.1/easybuild/easyconfigs
 * [x] $CFGS/m/M4/M4-1.4.18.eb (module: M4/1.4.18)
 * [x] $CFGS/z/zlib/zlib-1.2.11.eb (module: zlib/1.2.11)
 * [x] $CFGS/h/help2man/help2man-1.47.4.eb (module: help2man/1.47.4)
 * [x] $CFGS/m/M4/M4-1.4.17.eb (module: M4/1.4.17)
 * [x] $CFGS/b/Bison/Bison-3.0.4.eb (module: Bison/3.0.4)
 * [ ] $CFGS/f/flex/flex-2.6.3.eb (module: flex/2.6.3)
 * [ ] $CFGS/b/binutils/binutils-2.28.eb (module: binutils/2.28)
```

Toolchains and options

- It may be that you would prefer not to use the defaults given in a provided easyconfig.
- Make some changes using the command line:
  - Use an easyconfig with a different toolchain version:
    ```
    eb HDF5-1.8.20-foss-2018a.eb --try-toolchain-version 2019a
    ```
  - Use a completely different toolchain:
    ```
    eb HDF5-1.8.20-foss-2018a.eb --try-toolchain intel,2018a
    ```
  - Or change options (view these with `eb -a -e <easyblock>`):
    ```
    eb HDF5-1.8.20-foss-2018a.eb --try-amend variable=value
    ```

Easyconfigs

Contents of the HDF5-1.8.20-foss-2018a.eb easyconfig:

```
name = 'HDF5'
version = '1.8.20'

homepage = 'https://portal.hdfgroup.org/display/support'
description = """HDF5 is a data model, library, and file format for storing and managing
data.
 It supports an unlimited variety of datatypes, and is designed for flexible
 and efficient I/O and for high volume and complex data."""

toolchain = {'name': 'foss', 'version': '2018a'}
toolchainopts = {'pic': True, 'usempi': True}

source_urls = ['https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-%(version_major_minor
)s/hdf5-%(version)s/src']
sources = [SOURCELOWER_TAR_GZ]
checksums = ['6ed660ccd2bc45aa808ea72e08f33cc64009e9dd4e3a372b53438b210312e8d9']

dependencies = [
    ('zlib', '1.2.11'),
    ('Szip', '2.1.1'),
]

moduleclass = 'data'
```

Toolchains and options

- If you would prefer things to be more permanent you can easily write a new easyconfig (or, even easier, copy and modify an already existing one)
  - Change toolchain
    ```
    toolchain = {'name': 'foss', 'version': ‘2019a’}
    toolchain = {'name’: ‘intel', 'version': ‘2018a’}
    ```
  - Change toolchain options
    ```
    toolchainopts = {'pic’: False, 'usempi': True}
    toolchainopts = {'pic': True, 'usempi’: False}
    ```
  - Some easyblocks have very useful extra options, e.g. CP2K has: `type = ‘popt’` or `type = ‘psmp’` to build with MPI or MPI+OpenMP hybrid respectively.

EasyBuild

- Once you’ve selected or created the easyconfig you want to use to build and install your software, EasyBuild will build it, test it and install it in the configured location.
- Modulefiles for the correct environment will also be automatically generated ready for use.
- Load them and start using the software.

Quick plug:

- The EasyBuild developers will be running a free and more comprehensive online training session next week on Tuesday 23 June.
- https://github.com/easybuilders/easybuild/wiki/EasyBuild-Tutorial

Spack and EasyBuild

- To avoid confusion while managing a cluster, you would typically only want to use one of the two.
- Both are powerful tools:
  your circumstances should suit one (or possibly neither) better.

Spack and EasyBuild

- Spack:
  - Powerful concretizer generates new build configurations easily but can lead to many near duplicate installations if you aren’t careful.
  - Only as good as the packages – there’s potential for a new combination of dependencies or options to be incompatible.
  - Strong package management capabilities, and getting better.
- EasyBuild
  - Centrally provided easyblocks and easyconfigs should work, but there’s not as much immediacy in changing things up.
  - Some unnecessary packages may be built and installed, but probably not as many as Spack would.
  - Stable and resulting software environments easily reproducible.

Spack and EasyBuild

- Of course, you may still want complete control over your installation.
- In this case manually building yourself will remain the only way to maintain this level of control.
- If you want to highly customize the environment, this will be necessary.
- But there is real potential to reduce time and effort when using a build system such as Spack or EasyBuild.

Thanks for listening!