Uni.lu HPC School 2021

PS5: HPC Software Building: optimizing and complementing the ULHPC software set

Uni.lu High Performance Computing (HPC) Team

Dr. S. Varrette

University of Luxembourg (UL), Luxembourg

http://hpc.uni.lu

Latest versions available on Github:

UL HPC tutorials: https://github.com/ULHPC/tutorials

UL HPC School: hpc.uni.lu/education/hpcschool

PS5 tutorial sources: ulhpc-tutorials.rtfd.io/en/latest/tools/easybuild/

Summary

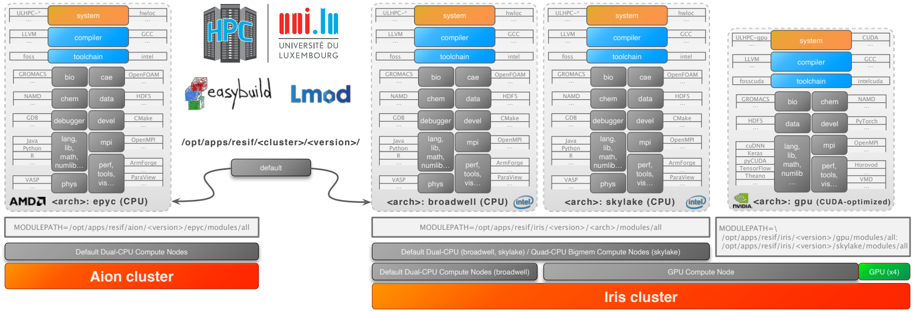
Main Objectives of this Session

- **Discover** Environment Modules and Lmod — Part 1
- Building Autotools-based software with GNU stow — Part 2
- Building software with Easybuild — Part 2
  - ↪ Understanding **local vs. global** installation
  - ↪ **Build your own software** on top of the provided software set
    - ✓ local installation of a select software
    - ✓ using **existing** easyconfigs
  - ↪ **Write your own easyconfig** file

Summary

UL HPC User Software Sets

- **Over 270 software packages** available for researchers
  - ↪ software environment generated using RESIF 3.0 framework over Easybuild
  - ↪ **optimized builds** organized by architecture, exposed through Environment Modules/Lmod
    - ✓ convenient way to dynamically change the users environment with `module` command
    - ✓ Categorized Naming Scheme <category>/<name>/<version>-<toolchain><versionsuffix>


ULHPC-*
system
hwloc
LLVM
compiler
GCC
foss
toolchain
intel
GROMACS
bio
cae
OpenFOAM
NAMD
chem
data
HDF5
GDB
debugger
devel
CMake
Java
Python
R
lang, lib, math, numlib...
mpi
OpenMPI
perf, tools, vis...
ArmForge
ParaView
VASP
phys
AMD <arch>: epyc (CPU)
UNIVERSITÉ DU LUXEMBOURG
easybuild
Lmod
/opt/apps/resif/<cluster>/<version>/
default
<arch>: broadwell (CPU)
<arch>: skylake (CPU)
ULHPC-gpu
CUDA
fosscuda
intelcuda
PyTorch
cuDNN
Keras
pyCUDA
TensorFlow
Theano
Horovod
VMD
<arch>: gpu (CUDA-optimized)
MODULEPATH=/opt/apps/resif/aion/<version>/epyc/modules/all
MODULEPATH=/opt/apps/resif/iris/<version>/<arch>/modules/all
MODULEPATH=\
/opt/apps/resif/iris/<version>/gpu/modules/all:
/opt/apps/resif/iris/<version>/skylake/modules/all
Default Dual-CPU Compute Nodes
Aion cluster
Default Dual-CPU (broadwell, skylake) / Quad-CPU Bigmem Compute Nodes (skylake)
Default Dual-CPU Compute Nodes (broadwell)
GPU Compute Node
GPU (x4)
Iris cluster

Software/Modules Management

- Key `module` variable: `$MODULEPATH` / where to look for modules.
  - ↪ **default** `iris`: `/opt/apps/resif/iris/<version>/{broadwell,skylake,gpu}/modules/all`
  - ↪ **default** `aion`: `/opt/apps/resif/aion/<version>/{epyc}/modules/all`

Software/Modules Management

- Key `module` variable: `$MODULEPATH` / where to look for modules.
  - ↪ **default** `iris`: `/opt/apps/resif/iris/<version>/{broadwell,skylake,gpu}/modules/all`
  - ↪ **default** `aion`: `/opt/apps/resif/aion/<version>/{epyc}/modules/all`

Command	Description
`module avail`	Lists all the modules which are available to be loaded
`module spider <pattern>`	Search for among available modules **(Lmod only)**
`module load <mod1> [mod2...]`	Load a module
`module unload <module>`	Unload a module
`module list`	List loaded modules
`module purge`	Unload all modules (purge)
`module use <path>`	Prepend the directory to the MODULEPATH environment variable
`module unuse <path>`	Remove the directory from the MODULEPATH environment variable

ULHPC Toolchains and Software Set Versioning

- **Yearly** release based on Easybuild release of toolchains
  - ↪ see Component versions (**fixed per release**) in the foss and intel toolchains
    - ✓ count 6 months of validation/import *after* EB release before ULHPC release

Name	Type	**2019b** (legacy)	**2020a**	**2020b** (prod)	**2021a** (devel)
GCCCore	compiler	8.3.0	9.3.0	10.2.0	10.3.0
foss	toolchain	2019b	2020a	2020b	2021a
intel	toolchain	2019b	2020a	2020b	2021a
binutils		2.32	2.34	2.35	2.36
Python		3.7.4 (and 2.7.16)	3.8.2 (and 2.7.18)	3.8.6	3.9.2
LLVM	compiler	9.0.1	10.0.1	11.0.0	11.1.0
OpenMPI	MPI	3.1.4	4.0.3	4.0.5	4.1.1

```
# test (new) development software set
resif-load-swset-devel
# Restore production settings
resif-load-swset-prod
```

ULHPC Software Sets in RESIF 3

- **User Software Sets** now defined as native Easybuild Module `Bundle` easyblock
  ↪ **ULHPC** bundles, associated to toolchain version – see `easyconfigs/u/ULHPC*`

Bundle Name	Description	Featured applications
ULHPC-<version>	Default global bundle for 'regular' nodes	ULHPC-*-<version> (root bundle)
ULHPC-toolchains-<version>	Toolchains, compilers, debuggers, programming languages, MPI suits, Development tools and libraries	GCCcore, foss, intel, LLVM, OpenMPI, CMake, Go, Java, Julia, Python, Spack...
ULHPC-bd-<version>	Big Data	Apache Spark, Flink, Hadoop...
ULHPC-bio-<version>	Bioinformatics, biology and biomedical	GROMACS, Bowtie2, TopHat, Trinity...
ULHPC-cs-<version>	Computational science, incl. CAE, CFD, Chemistry, Earth Sciences, Physics and Materials Science	ANSYS, OpenFOAM, ABAQUS, NAMD, GDAL, QuantumExpresso, VASP...
ULHPC-dl-<version>	AI / Deep Learning / Machine Learning	TensorFlow, PyTorch, Horovod...
ULHPC-math-<version>	High-level mathematical software and Optimizers	R, MATLAB, CPLEX, GEOS, GMP, Gurobi...
ULHPC-perf-<version>	Performance evaluation / Benchmarks	ArmForge, PAPI, HPL, IOR, Graph500...
ULHPC-tools-<version>	General purpose tools	DMTC, Singularity, gocryptfs...
ULHPC-visu-<version>	Visualization, plotting, documentation & typesetting	OpenCV, ParaView...
ULHPC-gpu-<version>	Specific GPU/CUDA-accelerated software	{foss,intel}cuda, cuDNN, TensorFlow, PyTorch, GROMACS...

Practical Session Time

Your Turn!

Hands-on Interact with the existing Software Set ▶ url ◀ | github | src

- **Discover** Environment Modules and Lmod
  - ↪ play with `module` command
  - ↪ understand `$MODULEPATH`
  - ↪ check module files `module show toolchain/foss`
  - ↪ search for a given software `module spider <pattern>`

Quickly swap ULHPC Software Set version

- If you find the software you're looking for YET in an old version:
  ↪ a **newer version may exists in the `devel` software set release**!

```
resif-load-swset-devel # Eq. of export MODULEPATH=$MODULEPATH_DEVEL
module spider <pattern>
```

- if on the contrary, you **want to use an deprecated software** from the `legacy` release
  ↪ common when we promote to production a new version

```
resif-load-swset-legacy # Eq. of export MODULEPATH=$MODULEPATH_LEGACY
module load <category>/<name>[/<oldversion>]
```

- use `resif-reset-swset` to restore the default MODULEPATH
  ↪ equivalent of `resif-load-swset-prod`

Useful RESIF 3.0 Variables

- **Recommended** to use to customize building path:

Environment Variable	Description	Example
`$ULHPC_CLUSTER`	Current ULHPC supercomputer you're connected to	`aion`
`$RESIF_VERSION_{PROD,DEVEL,LEGACY}`	Production / development / legacy ULHPC software set version	`2020b`
`$RESIF_ARCH`	RESIF Architecture	`epyc`
`$MODULEPATH_{PROD,DEVEL,LEGACY}`	Production / development / legacy MODULEPATH	

- Let's now assume you want to **build** a new version of an existing software
  ↪ or simply a missing one. . .

GNU Stow Concepts

- **stow directory** (`~/stow`):
  - ↪ root directory which contains all the stow packages, each with their own private subtree.
    - ✓ each subdirectory represents a stow package (`<name>-<version>-<cluster>` typically)
- **stow package** (`<name>-<version>-${ULHPC_CLUSTER}`):
  - ↪ nothing more than a list of files and directories related to a specific software
  - ↪ managed as an entity
  - ↪ you are free to use any naming convention: make is useful and self-explanaitory for you!
- **stow target directory**:
  - ↪ the directory in which the package files must appear to be installed.
  - ↪ By default the directory **above** the directory in which stow is invoked from.
    - ✓ This behaviour can be easily changed by using the `-t` option (short for `--target`), which allows us to specify an alternative directory.

GNU Stow Concepts

- Built Autotools-based software **prefixed to install in the stow dir**
  ↪ `./configure --prefix $HOME/stow/[...]; make && make install`
- Quickly install / uninstall a stow package as follows:

```
cd ~/stow
# install / enable <name> package
stow <name>-<version>-<cluster>
# [...]
# uninstall / disable <name> package
stow -D <name>-<version>-<cluster>
```

Practical Session Time

Your Turn!

Hands-on GNU Stow (Part 2)

▶ url ◀ | github | src

- Setup your homedir for GNU stow installation
  - ↪ `mkdir -p bin include lib share/{doc,man} src stow`
- Build and install 2 concurrent versions of GNU parallel.
  - ↪ build the lastest (up-to-date) version as in the "GNU Parallel" Tutorial
  - ↪ build intermediate version 20201222 version
- Quickly switch between the 3 co-existing co-existing:

↪ system version `/usr/bin/parallel`	`stow -D [...]`
↪ latest built	`stow parallel-20211222-${ULHPC_CLUSTER}`
↪ intermediate built	`stow parallel-20201222-${ULHPC_CLUSTER}`

Summary

Software/Modules Management

- Easybuild: open-source framework to (automatically) build scientific SW
- **Why?**: *"Could you please install this software on the cluster?"*
  - ↪ Scientific software is often **difficult** to build
    - ✓ non-standard build tools / incomplete build procedures
    - ✓ hardcoded parameters and/or poor/outdated documentation
  - ↪ EasyBuild helps to facilitate this task
    - ✓ **consistent** software **build and installation** framework,
    - ✓ **automatically generates LMod modulefiles**

```
(node)$ module spider BCFtools   # Complementaty tool to SAMTools
Lmod has detected the following error:  Unable to find: "BCFtools".
(node)$ module load tools/EasyBuild
# Search for recipes for the missing software
(node)$ eb -S BCFtools
(node)$ eb BCFtools-1.12-GCC-10.2.0.eb -Dr   # Dry-run
(node)$ eb BCFtools-1.12-GCC-10.2.0.eb -r
```

Recommended Settings for local Easybuild installs

- Easybuild is provided to you as a software module.

```
module load tools/EasyBuild
```

- Important configuration variables:
  - ↪ `EASYBUILD_PREFIX`: where to install **local** modules and software
    - ✓ **set globally for you** to `$HOME/.local/easybuild` by default
    - ✓ **YET better to make it match the cluster/software set version**
      see ULHPC technical documentation on Easybuild
  - ↪ `EASYBUILD_MODULES_TOOL`: the type of modules tool you are using, **i.e.** `LMod`
    - ✓ **set globally for you**
  - ↪ `EASYBUILD_MODULE_NAMING_SCHEME`:
    - ✓ the way the software and modules should be organized: flat view or hierarchical
    - ✓ **set globally for you to** `CategorizedModuleNamingScheme`

Practical Session Time

Your Turn!

Hands-on Easybuild (Part 3)

▶ url ◀ | github | src

- Local Easybuild configuration
  - ↪ see ULHPC technical documentation
  - ↪ `echo $EASYBUILD_PREFIX` should return `$HOME/.local/easybuild/aion/2020b`
- Search for a missing software `eb -S <pattern>`
  - ↪ understand how to select a promising Easyconfig
- Install the missing software
  - ↪ check missing dependencies (dry-run) `eb <name>.eb -Dr`
  - ↪ build and install `eb <name>.eb -r`
- Load the installed modules in your `$LOCAL_MODULES`
  - ↪ `mu && module load bio/BCFtools`

Thank you for your attention...

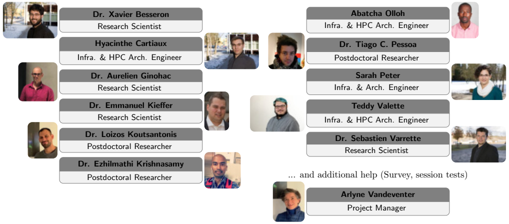
Questions?

ulhpc-tutorials.rtfd.io/en/latest/tools/easybuild/

High Performance Computing @ Uni.lu

University of Luxembourg, Belval Campus
Maison du Nombre, 4th floor
2, avenue de l'Université
L-4365 Esch-sur-Alzette
mail: hpc@uni.lu

Uni.lu HPC School 2021 Contributors



Dr. Xavier Besseron
Research Scientist

Hyacinthe Cartiaux
Infra. & HPC Arch. Engineer

Dr. Aurelien Ginohac
Research Scientist

Dr. Emmanuel Kieffer
Research Scientist

Dr. Loizos Koutsantonis
Postdoctoral Researcher

Dr. Ezhilmathi Krishnasamy
Postdoctoral Researcher

Abatcha Olloh
Infra. & HPC Arch. Engineer

Dr. Tiago C. Pessoa
Postdoctoral Researcher

Sarah Peter
Infra. & Arch. Engineer

Teddy Valette
Infra. & HPC Arch. Engineer

Dr. Sebastien Varrette
Research Scientist

... and additional help (Survey, session tests)

Arlyne Vandeventer
Project Manager

hpc.uni.lu