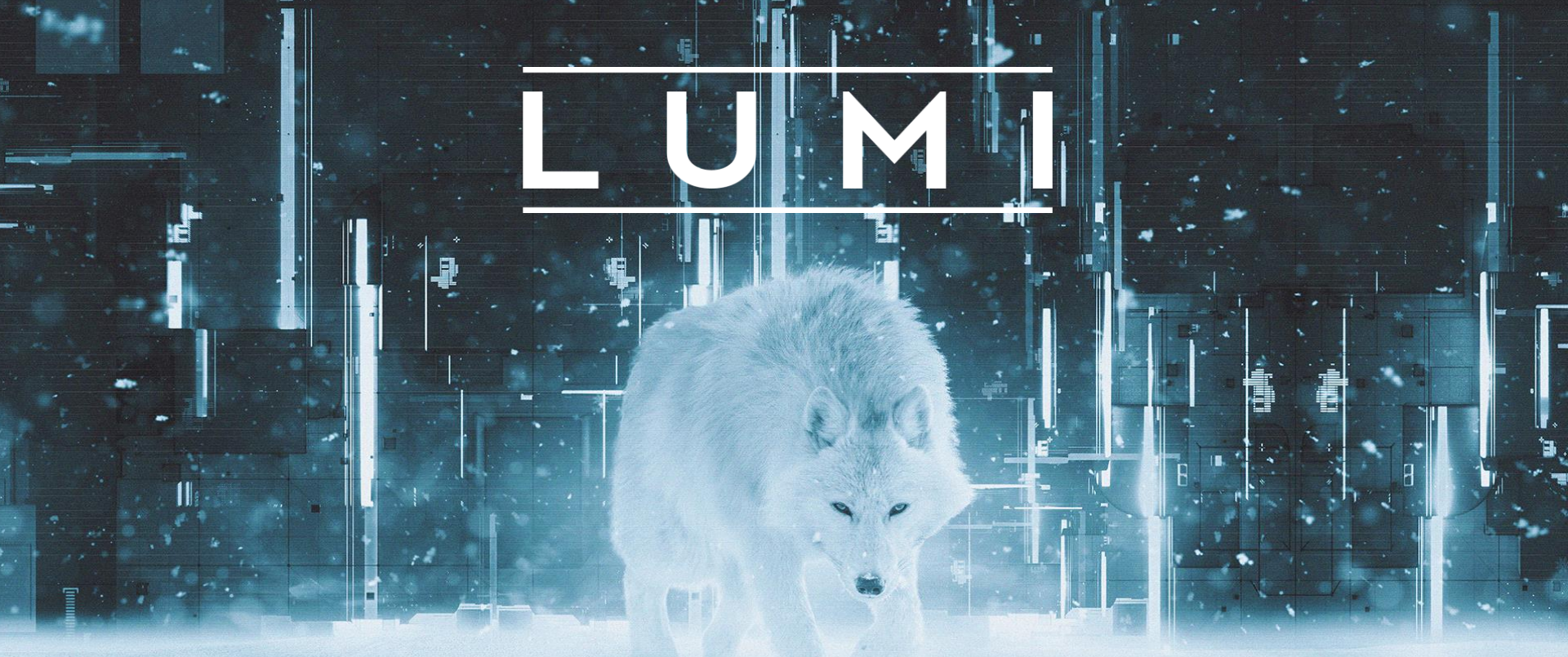


LUMI Software Stacks

Kurt Lust
LUMI User Support Team (LUST)
University of Antwerp

LUMI

What this talk is about…

- Software stacks on LUMI
- Some remarks about Lmod
- Creating your customised environment with EasyBuild

- Containers (separate presenter)

Design considerations

- Very leading edge and inhomogeneous machine (new interconnect, new GPU architecture with an immature software ecosystem, some NVIDIA GPUs for visualisation, a mix of zen2 and zen3)
  - Need to remain agile
- Users that come to LUMI from 11 different channels (not counting subchannels), with different expectations
- Small central support team considering the expected number of projects and users and the tasks the support team has
  - But contributions from local support teams
- Cray Programming Environment is a key part of our system
- Need for customised setups
  - Everybody wants a central stack as long as their software is in there but not much more
  - Look at the success of conda, Python virtual environments, containers, …

The LUMI solution

- Software organised in extensible software stacks based on a particular release of the PE
  - Many base libraries and some packages already pre-installed
  - Easy way to install additional packages in project space
- Modules managed by Lmod
  - More powerful than the (old) Modules Environment which is also supported by HPE Cray
  - Powerful features to search for modules
- EasyBuild is our primary tool for software installations
  - But uses HPE Cray specific toolchains
  - Offer a library of installation recipes
  - User installations integrate seamlessly with the central stack
  - We can help you with setting up Spack also, but this is not yet automated

Policies

- Bring-your-own-license except for a selection of tools that are useful to a larger community
  - One downside of the distributed user management is that we do not even have the information needed to determine if a particular userid can use a particular software license
- LUST tries to help with installations of recent software but porting or bug fixing is not our work
  - Not all Linux or even supercomputer software will work on LUMI
  - We're too small a team to do all software installations, so don't count on us to do all the work
- Conda, Python installations need to go in containers
  - We offer a container-based wrapper (lumi-container-wrapper) to do that

LUMI

Organisation: Software stacks

- CrayEnv: Cray environment with some additional tools pushed in through EasyBuild
- LUMI stacks, each one corresponding to a particular release of the PE
  - Work with the Cray PE modules , but accessed through a replacement for the PrgEnv-* modules
  - Tuned versions for the 4 types of hardware: zen2 (login, large memory nodes), zen3 (LUMI-C compute nodes), zen2 + NVIDIA GPU (visualisation partition), zen3 + MI250X (LUMI-G GPU partition)
  - Some software may be installed outside those stacks
- Far future: Stack based on common EB toolchains as-is
  - MPI may be the problem

Accessing the Cray PE on LUMI
3 different ways

LUMI

- Very bare environment available directly after login
  - What you can expect on a typical Cray system
  - Few tools as only the base OS image is available
  - User fully responsible for managing the target modules
- CrayEnv
  - "Enriched" Cray PE environment
  - Takes care of managing the target modules: (re)loading CrayEnv will reload an optimal set for the node you're on
  - Some additional tools, e.g., newer build tools (offered here and not in the bare environment as we need to avoid conflicts with other software stacks)
  - Otherwise used in the way discussed in this course

Accessing the Cray PE on LUMI
3 different ways

LUMI

- LUMI software stack
  - Each stack based on a particular release of the HPE Cray PE
    - Other modules are accessible but hidden from the default view
  - Better not to use the PrgEnv modules but the LUMI toolchains

HPE Cray PE	LUMI toolchain	
PrgEnv-cray	cpeCray	Cray Compiler Environment
PrgEnv-gnu	cpeGNU	GNU C/C++ and Fortran
PrgEnv-aocc	cpeAOCC	AMD CPU compilers
PrgEnv-amd	cpeAMD	AMD ROCm GPU compilers (LUMI-G only)

  - cpeXXX modules also load the MPI libraries and LibSci just as the PrgEnv-* modules
  - Environment in which we install most software

Accessing the Cray PE on LUMI
The LUMI software stack

LUMI

- The LUMI software stack uses two levels of modules
  - LUMI/21.08, LUMI/21.12: Versions of the LUMI stack
  - partition/L, partition/C, partition/EAP (and future partition/D, partition/G): To select software optimised for the respective LUMI partition
    - partition/L is for both the login nodes and the large memory nodes (4TB)
    - partition/EAP doesn't really have any software preinstalled (except for tools that we have everywhere)
  - Hidden partition/common for software that is available everywhere, but be careful using it for your own installs
  - When (re)loaded, the LUMI module will load the best matching partition module.
  - Hence be careful in job scripts: When your job starts, the environment will be that of the login nodes, but if you trigger a reload of the LUMI module it will be that of the compute node!

Exploring modules with Lmod

LUMI
Lmod

- Contrary to some other module systems, not all modules are immediately available for loading
  - Installed modules: All modules on the system that can be loaded one way or another
  - Available modules: Can be loaded without first loading another module
- Examples in the HPE Cray PE:
  - cray-mpich can only be loaded if a compiler module and network target module are loaded
  - Many of the performance monitoring tools only become available after loading perftools-base
  - cray-fftw only becomes avialable when a processor target module is loaded
- Tools
  - `module avail` is for searching in the available modules
  - `module spider` and `module keyword` are for searching in the installed modules

module spider

- `module spider` : Long list of all installed software with short description
  - Will also look into modules for "extensions" and show those also, marked with an "E"
- `module spider gnuplot` : Shows all versions of gnuplot on the system
  `module spider CMake`
- `module spider gnuplot/5.4.3-cpeGNU-21.12` : Shows help information for the specific module, including what should be done to make the module available
  - But this does not completely work with the Cray PE modules
- `module spider CMake/3.22.2` : Will tell you which module contains CMake and how to load it

module keyword

- Currently not yet very useful due to a bug in Cray Lmod
- It searches in the module short description and help for the keyword.
  - E.g., try
    ```
    module keyword https
    ```
- We do try to put enough information in the modules to make this a suitable additional way to discover software that is already installed on the system

Sticky modules and module purge

LUMI Lmod

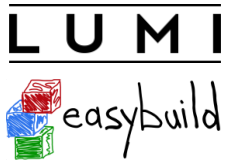
- On some systems, you will be taught to avoid `module purge` (which unloads all modules)
- Sticky modules are modules that are not unloaded by `module purge`, but reloaded.
  - They can be force-unloaded with `module --force purge` and `module --force unload`
- Used on LUMI for the software stacks and modules that set the display style of the modules
  - But keep in mind that the modules are reloaded, which implies that the target modules and partition module will be switched (back) to those for the current node.

Changing how the module list is displayed

LUMI
Lmod

- You may have noticed that you don't see directories in the module view but descriptive texts
- This can be changed by loading a module
  - ModuleLabel/label : The default view
  - ModuleLabel/PEhierarchy : Descriptive texts, but the PE hierarchy is unfolded
  - ModuleLabel/system : Module directories
- Turn colour on or off using ModuleColour/on or ModuleColour/off
- Show some hidden modules with ModulePowerUser/LUMI
  - This will also show undocumented/unsupported modules!

Installing software on HPC systems



- Software on an HPC system is rarely installed from RPM
  - Generic RPMs not optimised for the specific CPU
  - Generic RPMs may not work with the specific LUMI environment (SlingShot interconnect, kernel modules, resource manager)
  - Multi-user system so usually no "one version fits all"
  - Need a small system image as nodes are diskless
- Spack and EasyBuild are the two most popular HPC-specific software build and installation frameworks
  - Usually install from sources to adapt the software to the underlying hardware and OS
  - Installation instructions in a way that can be communicated and executed easily
  - Make software available via modules
  - Dependency handling compatible with modules

Extending the LUMI stack with EasyBuild

- Fully integrated in the software stack
  - Load the LUMI module and modules should appear in your module view
  - EasyBuild-user module to install packages in your user space
  - Will use existing modules for dependencies if those are already on the system or in your personal/project stack
- EasyBuild build-in easyconfigs do not work on LUMI
  - GNU-based toolchains: Would give problems with MPI
  - Intel-based toolchains: Intel compilers and AMD CPUs are a problematic cocktail
- Library of recipes that we made in the LUMI-EasyBuild-contrib GitHub repository
  - EasyBuild-user will find a copy on the system or in your install
  - Work on presenting a list in the documentation

Step 1: Where to install

- Default location is `$HOME/EasyBuild`
- But better is to install in your project directory for the whole project
  - `export EBU_USER_PREFIX=/project/project_465000000/EasyBuild`
  - Set this *before* loading the LUMI module
  - All users of the software tree have to set this environment variable to use the software tree
- Then load the LUMI module, partition module and EasyBuild-user module
  - In many cases, cross-compilation is possible by loading a different partition module than the one auto-loaded by LUMI

Step 2: Install the software

- Let's, e.g., install GROMACS
  - Search if GROMACS build recipes are available
    ```
    eb --search GROMACS
    eb -S GROMACS
    ```
    This will be improved in the future.
  - Let's take GROMACS-2021.4-cpeCray-21.12-PLUMED-2.8.0-CPU.eb:
    ```
    eb -r GROMACS-2021.4-cpeCray-21.12-PLUMED-2.8.0-CPU.eb -D
    eb -r GROMACS-2021.4-cpeCray-21.12-PLUMED-2.8.0-CPU.eb
    ```
- Now the module should be available
  ```
  module avail GROMACS
  ```

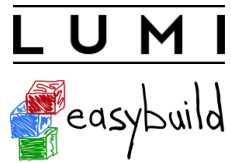
Step 2: Install the software - Note

- Note: Sometimes the module does not show up immediately. This is because Lmod keeps a cache and fails to detect that the cache is outdated.
  - Remove `$HOME/.lmod.d/.cache`
    ```
    rm -rf $HOME/.lmod.d/.cache
    ```
  - We've seen rare cases where internal Lmod data structures where corrupt and logging out and in again was needed
- Installing this way is 100% equivalent to an installation in the central software tree. The application is compiled in exactly the same way as we would do and served from the same file systems.

More advanced work

- You can also install some EasyBuild recipes that you got from support and are in the current directory (preferably one without subdirectories):
  `eb -r . my_recipe.eb`
  - Note the dot after the `-r` to tell EasyBuild to also look for dependencies in the current directory (and its subdirectories)
- In some cases you will have to download the sources by hand, e.g., for VASP, which is then at the same time a way for us to ensure that you have a license for VASP. E.g.,
  - `eb --search VASP`
  - Then from the directory with the VASP sources:
    `eb -r . VASP-6.3.0-cpeGNU-21.12.eb`

More advanced work (2): Repositories

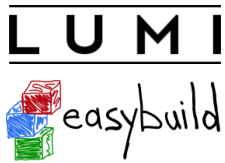


- It is possible to have your own clone of the LUMI-EasyBuild-contrib repo in your $EBU_USER_PREFIX subdirectory if you want the latest and greatest before it is in the centrally maintained repository
  - `cd $EBU_USER_PREFIX`
    `git clone https://github.com/Lumi-supercomputer/LUMI-EasyBuild-contrib.git`
- It is also possible to maintain your own repo
  - The directory should be `$EBU_USER_PREFIX/UserRepo` (but of course on GitHub the repository can have a different name)
  - Structure should be compatible with EasyBuild: easyconfig files go in `$EBU_USER_PREFIX/easybuild/easyconfigs`

More advanced work (3): Reproducibility

- EasyBuild will keep a copy of the sources in `$EBU_USER_PREFIX/sources`
- EasyBuild also keeps copies of all installed easyconfig files in two locations:
  - In $EBU_USER_PREFIX/ebrepo_files
    - And note that EasyBuild will use this version if you try to reinstall and did not delete this version first!
    - This ensures that the information that EasyBuild has about the installed application is compatible with what's in the module files
  - With the installed software (in $EBU_USER_PREFIX/SW) in a subdirectory called easybuild
    This is meant to have all information about how EasyBuild installed the application and to help in reproducing

EasyBuild training for support team members



- EasyBuild web site: easybuild.io
- Generic EasyBuild training materials on easybuilders.github.io/easybuild-tutorial.
- Training for CSC and local support organisations on May 9 and May 11, 12:30-15:30 CEST.
  - Contact LUMI support if you want to join.