EasyBuild on LUMI one year later: successes and frustrations

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

LUMI: one of the fastest supercomputers in the world

LUMI

- LUMI is a **HPE Cray EX** supercomputer manufactured by **Hewlett Packard Enterprise**
- Two main compute sections:
  - LUMI-C with 1536 2-socket AMD Milan nodes
  - LUMI-G with 2560 nodes with 4 MI250x GPUs each
- Peak vector FP64 performance close to **500 petaflop/s** makes the system one of the world's fastest
  - Roughly one third of Frontier, and same architecture
  - Peak matrix FP64 performance close to 1 exaflop

1 system

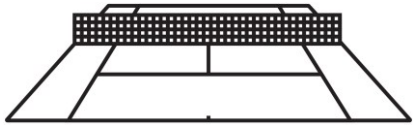
550 Pflop/s

Peak Performance

Computing power equivalent to

1 500 000

Modern laptop computers



Size of a tennis court

Modern platform for

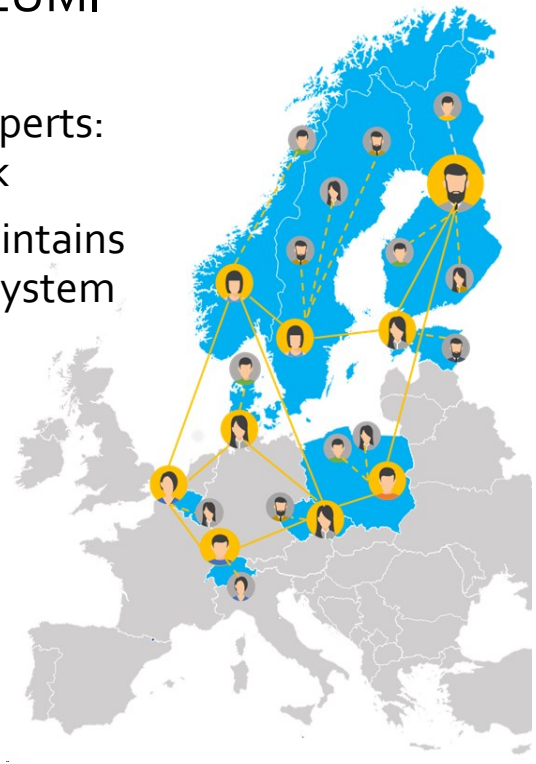
High-performance computing, Artificial intelligence, Data analytics

Based on GPU technology

LUMI

LUMI user support

- Centralized virtual help-desk run by the distributed LUMI User Support Team
  - The model is based on a network of dedicated LUMI experts: each partner (except one) provides one FTE for the task
  - User Support Team also provides end-user training, maintains the software portfolio and user documentation of the system
- ”Level 3” support (e.g. application enabling, methodology support) via local centres, the EuroHPC Competence Centres, a team at HPE and AMD, and new EuroHPC projects
- National support for issues with accounts and allocations



Maintain a software stack for…

LUMI

- A rather experimental and inhomogeneous machine (new interconnect, new GPU architecture with an immature software ecosystem, some NVIDIA GPUs, a mix of zen2 and zen3)
- With users that come to LUMI from 11 different channels (not counting subchannels)
- And this has to be done by a (too) small central support team considering the expected number of projects and users and the tasks the support team has
  - But the consortium should contribute
- Cray Programming Environment is a key part of our system
  - And Clang/LLVM is an important compiler for us
- Operational: 4 copies of the software stack due to the file system setup

Lean and customisable

LUMI

- Small central software stack of high-priority libraries
  - Quick updating after installation of a new programming environment
- Other easyconfigs evolve as needed
  - Installed in user's project directory
  - Development often driven by request
  - Sometimes even very customised setups that we may keep out of the repositories to avoid confusion
- Managing such evolution in a central installation is hard
  - E.g., cannot install over another package while users might be running
  - And even harder for us as we have to keep 4 copies in sync

Personal environments

LUMI

- Every user wants **their** packages preferably preinstalled in a central stack
  - But doesn't want to be confronted with the packages that someone else needs
  - "What modules should I load? I really can't find my way in this mess…"
- Less problems with version conflicts than in a big central stack
  - It is only a matter of time before things become as bad as it is today in Python
  - Having everything in one big stack slows down development tremendously, especially if you start from the principle that you try to avoid two different versions of a package (dependency) in a single stack.
- Look at the success of Conda, containers, Python virtual environments, …
- Setup with an Lmod EasyBuild configuration module and software stack modules that ensure that user installed software blends seamlessly with centrally installed software.

EasyBuild installation on LUMI

LUMI

- Software stacks based on the releases of the HPE Cray PE
  - Compilers not installed with EasyBuild
  - 4 different compilers, 3 hardware platforms
  - Toolchains derived from the work done at CSCS
  - No hierarchy: Only full toolchains
    - Though we are thinking how we could implement something that takes the role of GCCcore.
- Fix the version of EasyBuild for a given version of the software stack
  - Bootstrapped for each version of the LUMI software stack to make those stacks fully independent of each other

EasyBuild installation on LUMI (2)

LUMI

- 2 central repositories with various service levels:
  - LUMI-SoftwareStack: Repository for the central software stack and some other packages that we fully support and install centrally
  - LUMI-EasyBuild-contrib: Repository for software that we do not want to install centrally, e.g., because we cannot fully support the package or are not convinced that the configuration is already OK for a large enough group of users, or license conditions make it impossible to offer the package to everybody.
    - Would also include "annoying" packages such as OpenFOAM or Yambo that will probably never make it to the central stack

EasyBuild installation on LUMI (3)

LUMI

- Configuration modules for EasyBuild to configure for specific tasks
  - Single module for the LUMI stacks linked with different names, e.g.,
    - EasyBuild-production when installing software in the central stack
    - EasyBuild-user to let a user install software
  - Single piece of code is more complex but it is easier to ensure consistency of the settings for central and user/project install of software
  - Picks up where to install software from its name and its location in the module tree
- Installing GROMACS:

```
$ ml LUMI/22.08 partition/C EasyBuild-user
$ eb GROMACS-2021.4-cpeGNU-22.08-PLUMED-2.7.4-CPU.eb -r
```

SYSTEM toolchain

LUMI

- We (ab)use the SYSTEM toolchain and therefore are annoyed that it doesn't always work as other toolchains
- Our use case:
  - Install software that is available without loading any toolchain or HPE Cray compiler module
  - And we often use static linking for those packages to minimize interactions with other software on the system
  - E.g., a set of build tools available to (almost) all users.
- Developing a GCCcore equivalent may be a partial but not a complete solution

GPU toolchains

LUMI

- In the HPE Cray PE ecosystem this is meant to be fairly transparent.
  - Same compiler wrappers, but loading a different set of so-called target modules to reconfigure the wrappers and some GPU-specific modules (for CUDA or ROCm)
  - Needed very little work, though may need to think about more toolchain options
- ROCm on LUMI
  - Comes pre-packaged with the HPE Cray PE and installs in its default location
  - As developers often want a newer version, we build our own ROCm modules also that resemble the HPE Cray PE modules as much as possible
    - ROCm versions limited by the driver version
    - So far, things often just work
    - Problem with ROCm: Hard-coded paths to the default locations (/opt/rocm-5.2.5) and sometimes even not version-specific (/opt/rocm).
    - Can have performance implications for some software (MIOpen seems to be the main victim)

But…

LUMI

- May do a better job with more settings via toolchainopts but would need to gather more insight in how packages are built.
- EasyBuild does not distinguish enough between languages from the C family
  - Just cstd and no cxxstd, but the same is added to both CFLAGS and CXXFLAGS…
  - COMPILER_C_UNIQUE_FLAGS but could not find a COMPILER_CXX_UNIQUE_FLAGS
  - HIP, SYCL, OpenCL all may require different settings.
- No way to add flags to LDFLAGS (apart from additional –L arguments)

Documentation is everything!

LUMI

- Make full use of EasyBuild features to include help information in the Lmod modules.
- On a HPE Cray system users really need to learn to read man pages again.
  - For those used to reading them we do include them in the software installations also.
- Developed the LUMI Software Library
  - Generated from markdown files stored with the easyconfigs
  - Idea: Documentation is more likely to be adapted when the software changes if it is not stored separately

[package list]

lumi-vnc

pre-installed

License information

The lumi-vnc module provides scripts developed by CSC for LUMI.

Thse scripts are licensed under the MIT License, see the LICENSE file in the utily-tools repository for LUMI.

User documentation

To know how to start the VNC server, check the help information included in the most recent version of the module returned by the above `module spider` command. E.g., assuming that version is 20230110:

```
module spider lumi-vnc/20230110
```

Known issues

Pre-installed modules (and EasyConfigs)

To access module help and find out for which stacks and partitions the module is installed, use `module spider lumi-vnc/<version>`.

EasyConfig:

- lumi-vnc/20230110 (EasyConfig: lumi-vnc-20230110.eb)

Technical documentation

Installs a singularity container with the TurboVNC server and some scripts and shell functions to start and stop the server.

- See the repository Lumi-supercomputer/utility-tools for the scripts and container definition.

Instructions for system staff

- It is not yet possible to build the container on LUMI. Hence it has to be build offline. The EasyConfig file then expects to find the container in a tar file in the root directory of that archive, with the tar file named `turbovnc-container-<version>.tar` with `<version>` the version used for the module. The container file itself is called `vnc.sif`. The tar file is used to be able to store different versions of the container in the sources directory to be able to reproduce older versions in case of problems.

  That tar-file should then be put in a location where EasyBuild can find it, e.g., in `/appl/lumi/sources/easybuild/l/lumi-vnc` on uan04.

- The EasyConfig file can then be used to install the TurboVNC container. Note though that downloading the sources if they are not yet in a location where EasyBuild can find them, e.g., in the directory mentioned above which is the master copy for the software

Table of contents

But…

LUMI

- Cannot exploit the Lmod whatis lines enough
  - Even though these are just strings in Lmod, Lmod expects "key: value" strings.
  - The "Description:" line has a special status in Lmod. Using the same easyconfig `description` parameter for whatis and the help block is not a good idea
    - The description should probably be the same for all modules with the same name as sometimes only one is shown, e.g., when using module whatis without arguments.
    - The description in whatis should be short to not screw up the output too much.
    - But the description in the help block of a module should be more descriptive and can give more details about the specific module.
    - Suggestion: `short_description` and use this for whatis if present and if not, use the regular `description` (to not break compatibility).

But…

LUMI

- And the current `whatis` keyword isn't optimal either
  - Doesn't enforce the key:value idea.
  - As soon as you use it, EasyBuild doesn't add the data anymore that it would otherwise add automatically.
- Just an idea: I noticed that at HLRS modules for libraries build with autotools or CMake contained a line with the arguments that were passed to those tools so that I could see how the libraries were configured.

EasyBlocks

LUMI

- EasyBlocks often fail on LUMI in ways that can be avoided
- Some EasyBlocks are too strict when testing for compilers and fail if it is not one they recognize
  - Understandable if the argument is that we could not test that configuration
  - But isn't a user better served with something that might work than with something that doesn't work at all?
- Testing for modules should be done through metadata and not through module names
  - So they would work with external modules also
  - It would be great to have a way to recognize moduleless OS dependencies also through metadata generated on the fly
  - A system may use another module name or bundle to provide the software

Module and directory hell

LUMI

- Arguments in favour of splitting up in many modules
  - More manageable chunks for installation
- Arguments against splitting up in too many modules
  - Disk capacity is cheap but IOPs are expensive
  - Long PATH variables and long link lines are terrible for developers
  - Some software expects some components to be together (various netCDF interfaces)
  - Too many modules that few users need explicitly complicate the search for suitable modules for others
  - Easy to overlook dependencies, having software pick up libraries from the system instead
  - What is the point of having packages in separate modules if we want only one version of each in a software stack?
    - You don't really save space as most basic packages are installed everywhere anyway

Module and directory hell (2)

LUMI

- Non-arguments in favour of splitting up
  - Better visibility of what is installed
    - With appropriate information in module files and/or the use of extensions in Lmod, module spider is a very powerful tool to find software in bundles
  - Don't install more than what is really needed
    - See above: disk space is cheap, IOPs are expensive so it may be better to reduce the number of directories that have to be searched than the amount of disk space consumed
  - But Linux distributions do it too so why would we not mimic that?
    - One big difference: Linux distributions do install various packages in the same directory
    - Some distributions target very small systems also but that is a non-issue for HPC systems
    - EESSI is probably targeting workstations also but caching is on a file basis anyway so files that are not needed will not be pulled in?
- If ease of installation in more manageable chunks is the argument then we need a better way of bundling installations, not more modules

Python and conda with Tykky

LUMI

- CSC-developed tool to package Conda and pip-installed Python installations.
- Aim: Reduce load on the Lustre metadata servers
- Approach
  - Minimal singularity container (OS dependent)
  - Packs the installation in a SquashFS file during creation, mounted in the container when running
  - Creates wrapper scripts for commands in the bin subdirectory to automatically run them in the container

Python and conda with Tykky (2)

LUMI

- No need to be able to build containers on the system
  - Good for LUMI as fakeroot is currently not enabled
- On LUMI, uses the Cray-provided Python and packages and installs on top of those
- I have been thinking about ways to interface with EasyBuild
  - Generate from an EasyConfig and create a module to put the bin directory in the search path (but without using EasyBuild to build the software)
  - Or even using EasyBuild to build software that is then packaged
    - But obviously this would require a good Bundle approach to install all required software at once.

  But basically lack the time to do it.
- docs.csc.fi/computing/containers/tykky/

Where is LLVM? Where is MPICH?

LUMI

- Support for LLVM very limited at the moment
  - Understandable due to the confusing state of Fortran support
  - But it is the number one compiler base for HPC and even more so outside HPC
    - More and more vendor compilers build on Clang/LLVM
    - Better platform for compiler research so it attracts more developers
    - LLVM ecosystem is the basis for most GPU development (including SYCL)
- Where is MPICH?
  - Vendor MPIs often derived from MPICH, particularly from network vendors (with the exception of Mellanox/NVIDIA InfiniBand)
  - OpenMPI is behind in GPU support for networks that do not support UCX
    - Looks like things have not yet changed in 5.0

Stop bashing spack!

LUMI

- Our experience is that with Spack we often have a solution much quicker!
  - Comparing someone who is experienced with Spack doing the Spack work and someone who is experienced with EasyBuild doing the EasyBuild work
- Even though the API changes from version to version, the API for features that have been longer in Spack seems pretty stable
  - And the API is also much more readable than the API for EasyBlocks
- The criticism that when you're building something with Spack, it is basically an untested configuration is only partly true.
  - Spack has quality control also, though of course they cannot test every combination, but…
  - … very often it just works, and when it does there is no way EasyBuild will beat it.
  - Given the variations in Linux and underlying hardware requiring different compiler optimisation options, the EasyBuild recipe may also not be tested.

Stop bashing spack! (2)

LUMI

- "If you want to add another package to an environment, the concretizer may come up with a very different solution, forcing you into reinstalling a lot".
  - May be true, but…
  - … as far as I know it tries to take into account what is installed already.
  - And in most cases EasyBuild would be unable to come up with a solution at all, or at least also require installations in different toolchains that cannot be loaded together.
  - So from a user's perspective Spack is then the better tool to create an environment as they at least have something…
- A flexible tool whose developers don't feel the need to test everything may be better prepared for the "Cambrian explosion" phase we're in
- Bashing the competitor can make you blind for your own shortcomings

Meet in the middle?

LUMI

- Spack and EasyBuild on opposite ends of the spectrum?
  - Spack the very flexible tool for individual (but untested, at least that is what the EasyBuild community sometimes claims) configurations
  - EasyBuild with fully fixed but well tested configurations
- And maybe we really need something in the middle?
  - I agree and I often think about how to make EasyBuild a bit more flexible
    - And have some awkward ways to do that on LUMI at the moment
  - But actually Spack is taking that middle already with its environments features,
  - while sometimes I have the feeling that EasyBuild is moving even more to the extreme

EESSI: An opportunity and a threat

LUMI

- Opportunity: Extra personpower for EasyBuild development also
- Threats:
  - Even more a focus on a single big software stack and features to enable that
  - Even less emphasis on making EasyBuild a flexible tool
  - Support hell for systems that use EESSI is a threat to the EESSI project but may turn out to be a threat to the EasyBuild project also if EasyBuild becomes too dependent on EESSI for success
    - If a user at a site that uses EESSI notes a problem with an installation, who will they turn to?
    - If a solution requires interaction between the people who manage the hardware and the people who manage the software installation, how will this work?

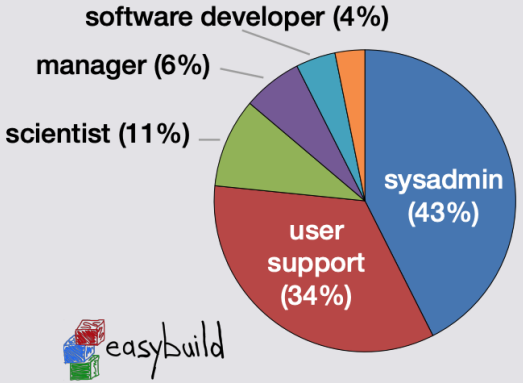
EESSI: An opportunity and a threat

LUMI

- Some thoughts
  - Finding a user-friendly way to build on top of EESSI will be crucial!
  - Other distribution models are also crucial
    - Not every site ready to set up a cache (though the CSCS setup for LHC/Atlas could be inspiring)
    - Extra daemons are not always an option
      - But could be solved by running in a container, at the cost of scalability?
    - Putting stuff in /opt is also not always an option
      - Could also be build in the container instead
  - A native build with or without the compatibility layer may be a better solution for a system as LUMI.

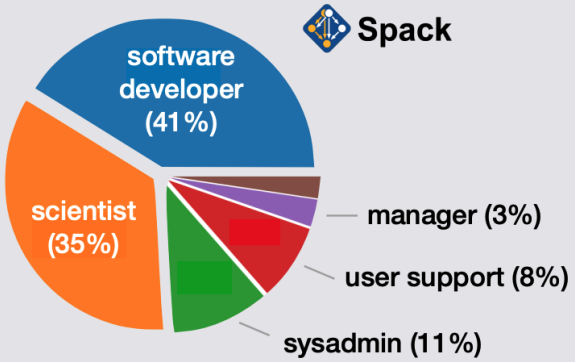
EuroHPC: An opportunity and a threat

LUMI



"What kind of user are you?"

- EasyBuild: sysadmins + user support (77%)
- Spack: software developers + scientists (76%)
- User communities are clearly very different!
- **Impact of different philopsophies?**



- Observations:
  - Plenty of money for machines but very little for support close to the machine
  - Seems to favour strong communities that do development and support near the user rather than near the supercomputer
- Isn't this a better fit for the Spack model?

Conclusions

LUMI

- EasyBuild on LUMI is an a-typical EasyBuild installation
  - EasyBuild does not control the whole environment from the compilers onwards
  - Not the standard EasyBuild compilers
  - Works, but probably with more pain than needed
- Continued investments in EasyBuild on LUMI
  - But we are not blind for other options, as …
  - … particularly Spack does a good job on LUMI for some users…
- Users needed some time to adapt to the personal environment idea but I think many now see the benefits

Conclusions (2)

LUMI

- Things for improvements
  - Toolchain issues: SYSTEM, C/CXX issues, but others are very Cray-specific
    - Could argue about AMD GPU support also of course
  - Documentation through modules issues
  - Easyblocks
  - Module hell and metadata server load
  - Where is LLVM? Where is MPICH?
    - EasyBuild support is currently best on Mellanox/NVIDIA hardware…
  - EasyBuild community may be a bit guilty of navel-gazing or taking an extreme position
- Keep your eyes open for upcoming threats!