Leibniz Supercomputing Centre

of the Bavarian Academy of Sciences and Humanities

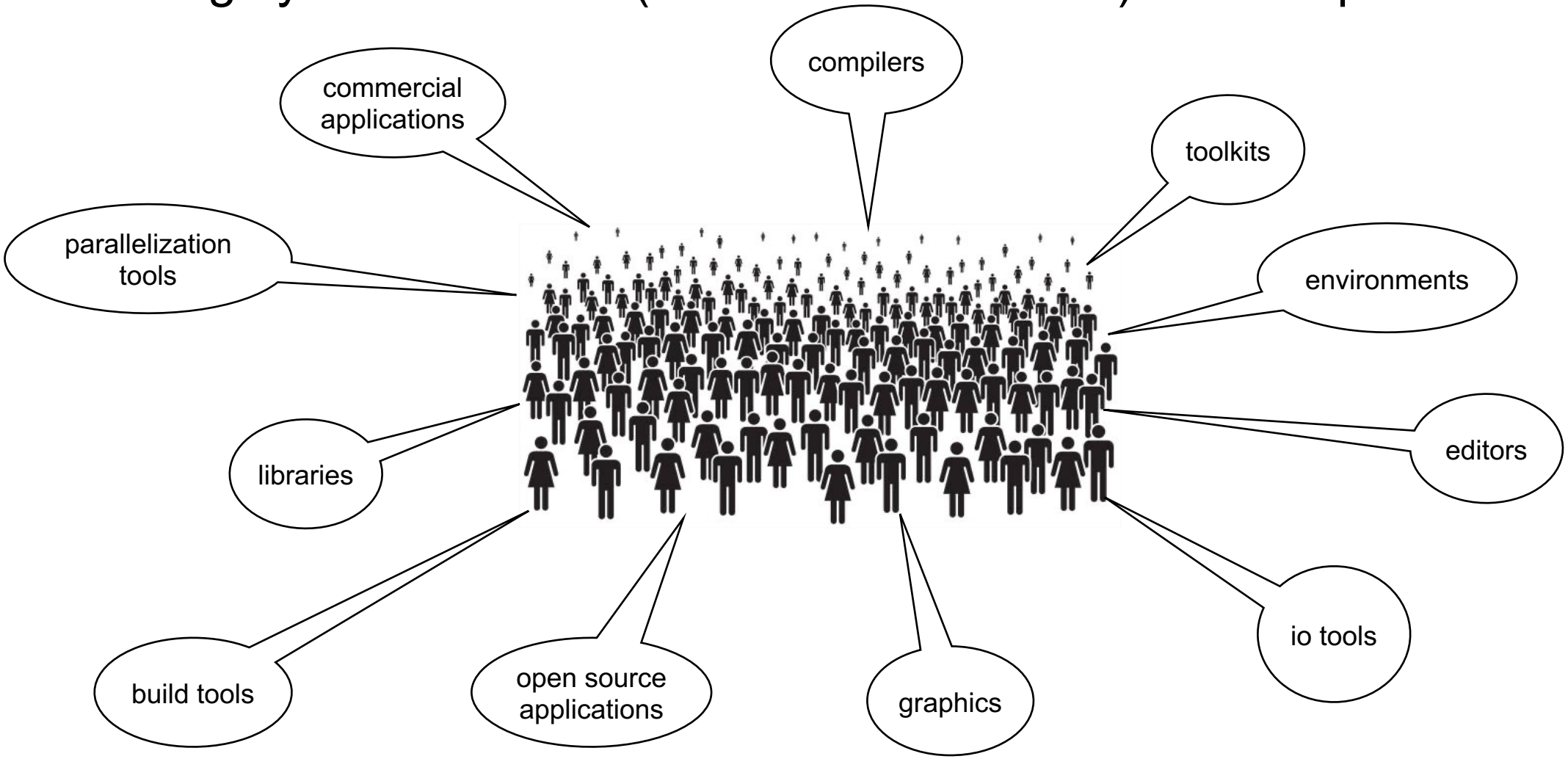
Leibniz Supercomputing Centre

Software Provisioning at LRZ | Nisarg Patel

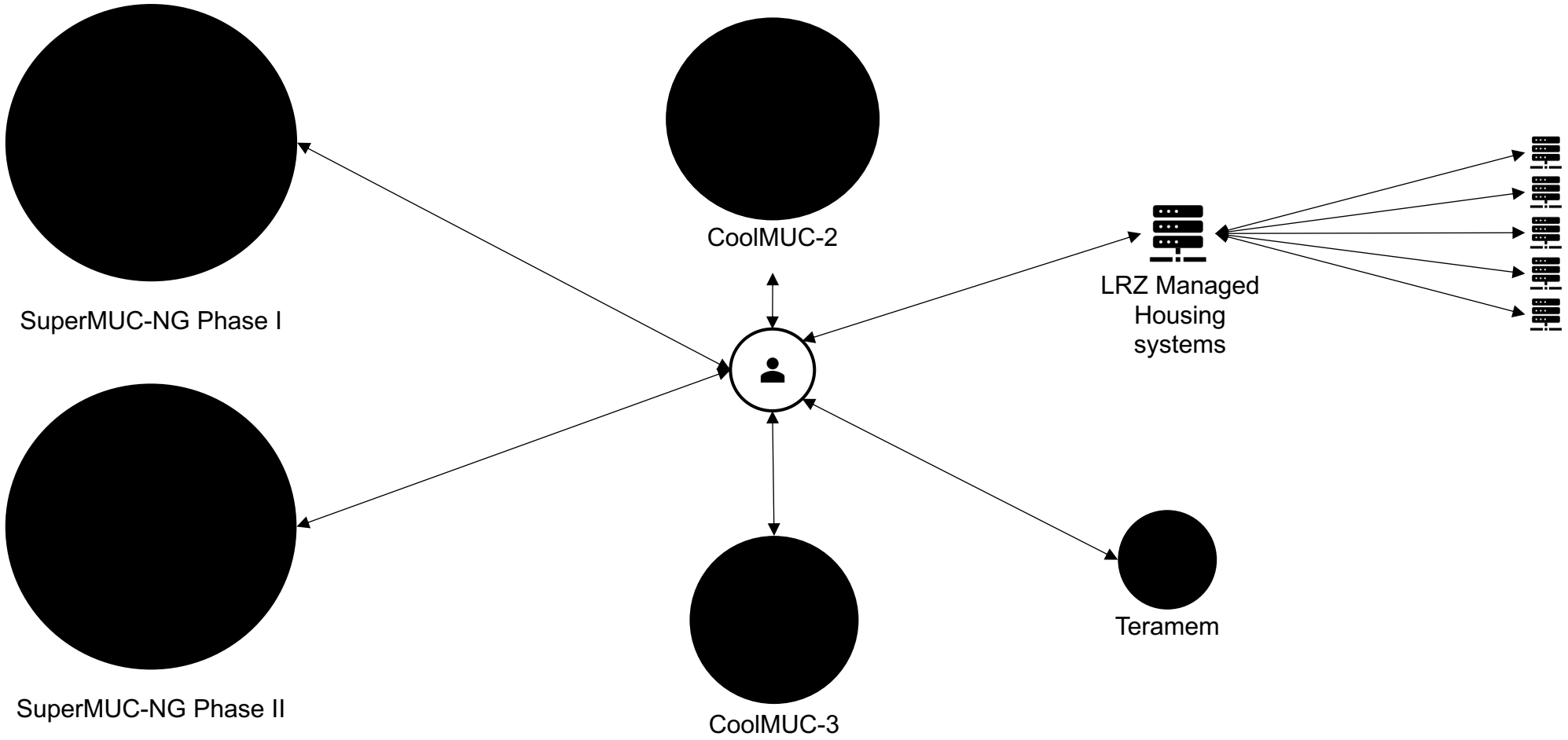
Motivation

- Software stack on compute resources
- How do we manage HPC Software at LRZ
- General info about Software Stack
- Environmental Modules
- Flow chart to get my application ready for HPC
- Spack Introduction and user spack.
- Spack and Conda envs

HPC users
We have highly diverse users (< **4500 active users**) with unique needs



Software in form of “modules” on compute resources at LRZ

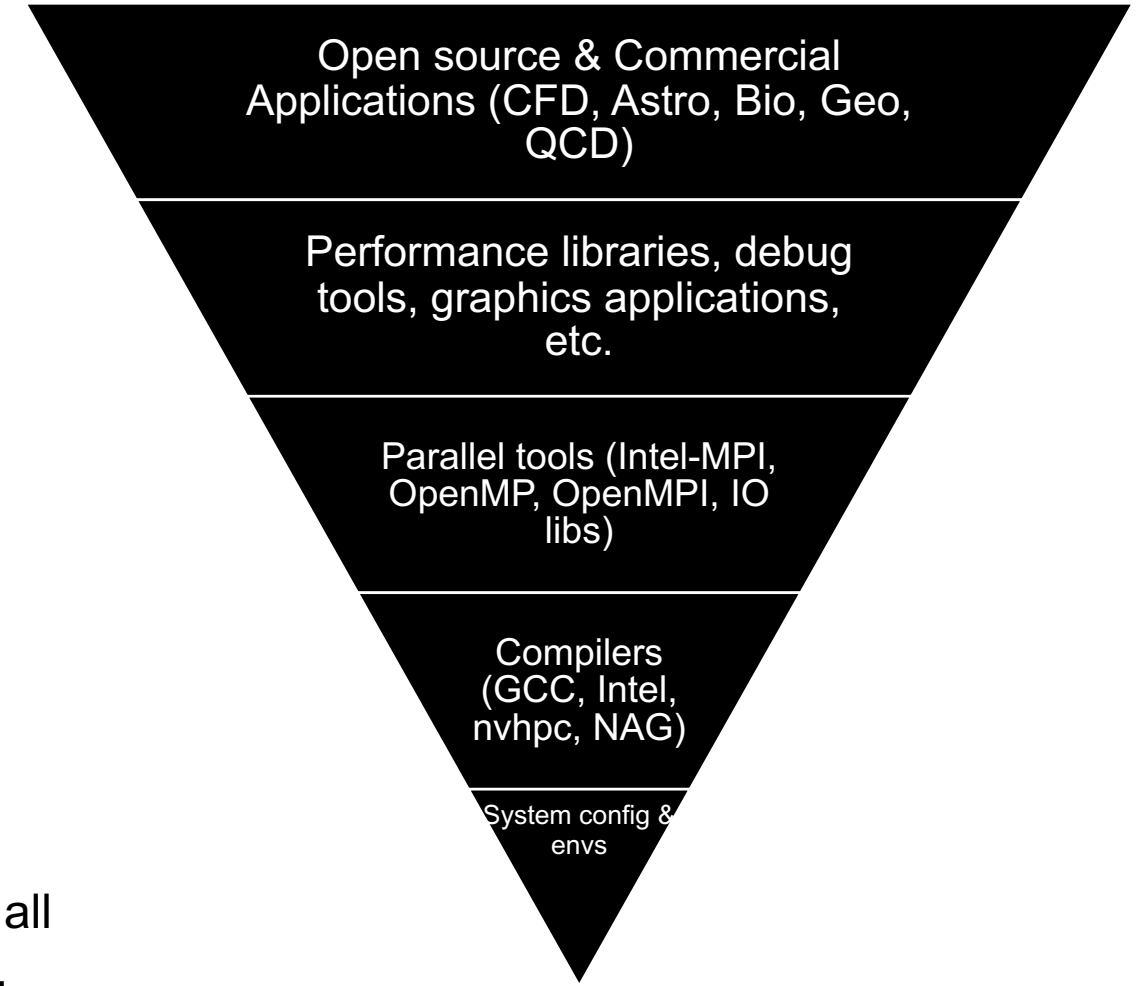


Software stack on compute resources

- About ~400 environment modules on these machines,
  - SuperMUC-NG Phase I & II
  - CoolMUC-2
  - CoolMUC-3
  - Housing Clusters

- Libraries and applications are build for,
  - Specific architectures
    - Saphirerapids
    - Skylake_avx512
    - Haswell
    - KNL
  - General build
    - x86_64

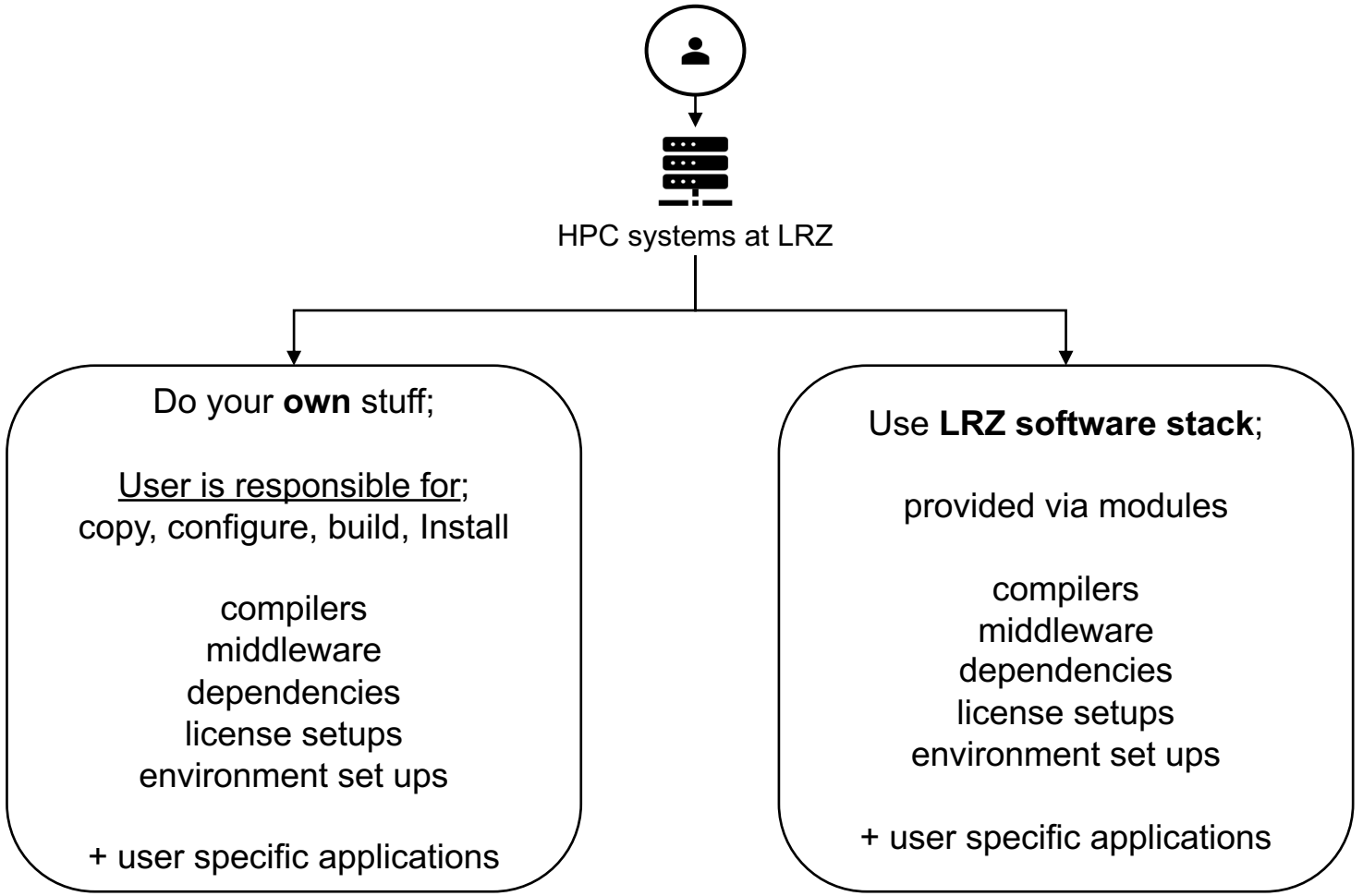
- Baring most commercial applications, **about >90%** of all software/libraries are provided with the help of **Spack**.



Catering software needs for HPC users

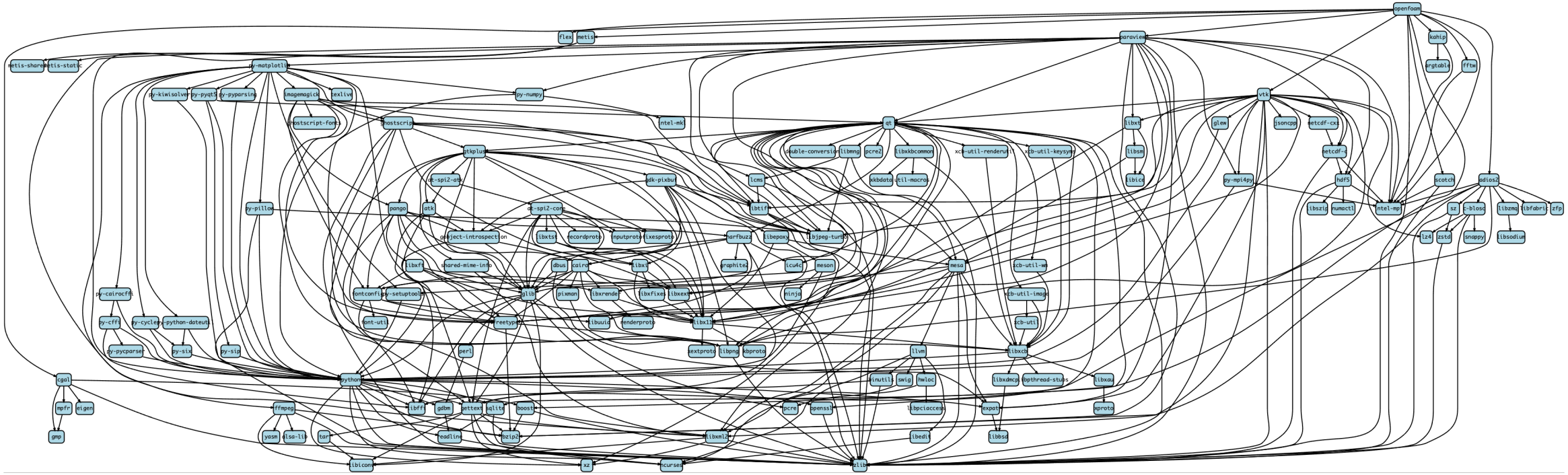
- Why do we provide software support at all?
  - We have conventional HPC users + novice users + extremely complicated systems (multiple of them)
  - We provide software support to facilitate and expedite the usage of HPC as easily as possible.
  - Efficient and correct software enables users to effectively utilize the system.
- To cater large and highly diverse group of users, we in CXS+HPC group manage HPC software either,
  - Manually by each individual application maintainer
  - In semi-automated manner, using Spack
- Each software (module) that you see on LRZ systems, someone from CXS or HPC department is responsible
  - for managing the state of installations
  - providing modules
  - support users with individual software

Getting user application ready for HPC



Why do we provide software? The dependency-hell

- A high-level application may just be the „tip of an iceberg“ when considering a feature-rich configuration of the software with all it‘s dependencies.
- Feature-rich CFD-Package **OpenFOAM** incl. vtk & paraview (with 140 dependencies) looks like,

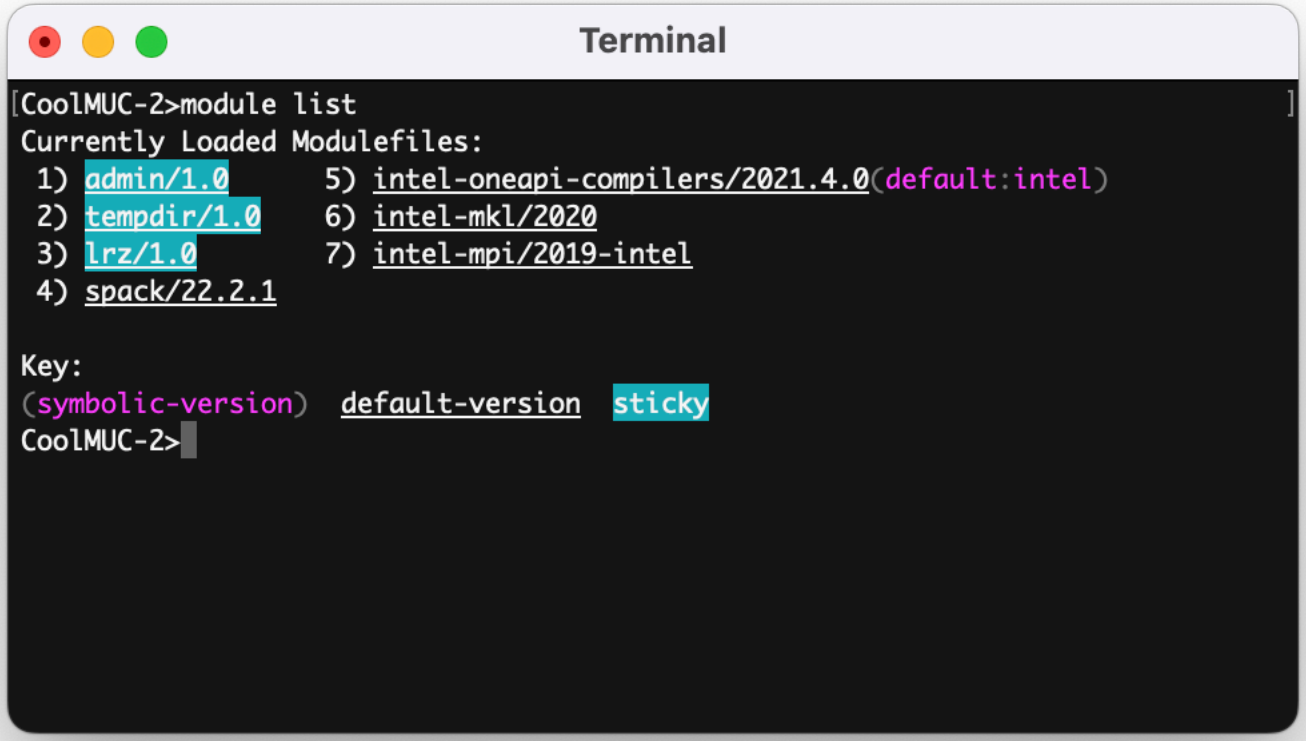


Software available as Environmental Modules!

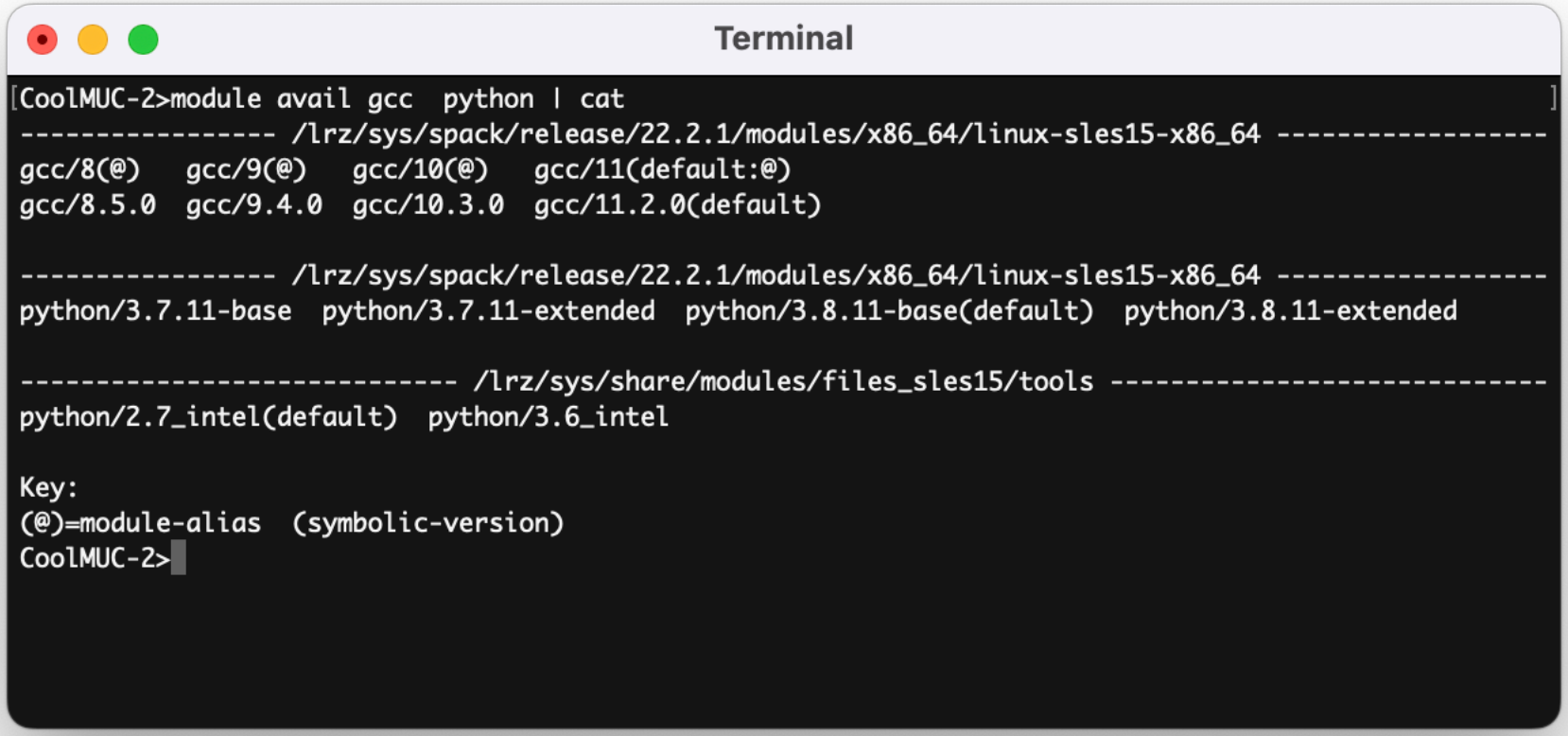
- Modules provide a flexible way to configure and access various applications, compilers, tools and libraries dynamically by managing the shell environment

- In other words modules allow for the dynamic modification of environment variables which handles;
  - library paths
  - binary paths
  - license server settings
  - application specific configurations
  - header files
  - pkg-config files
  - wrapper commands or scripts

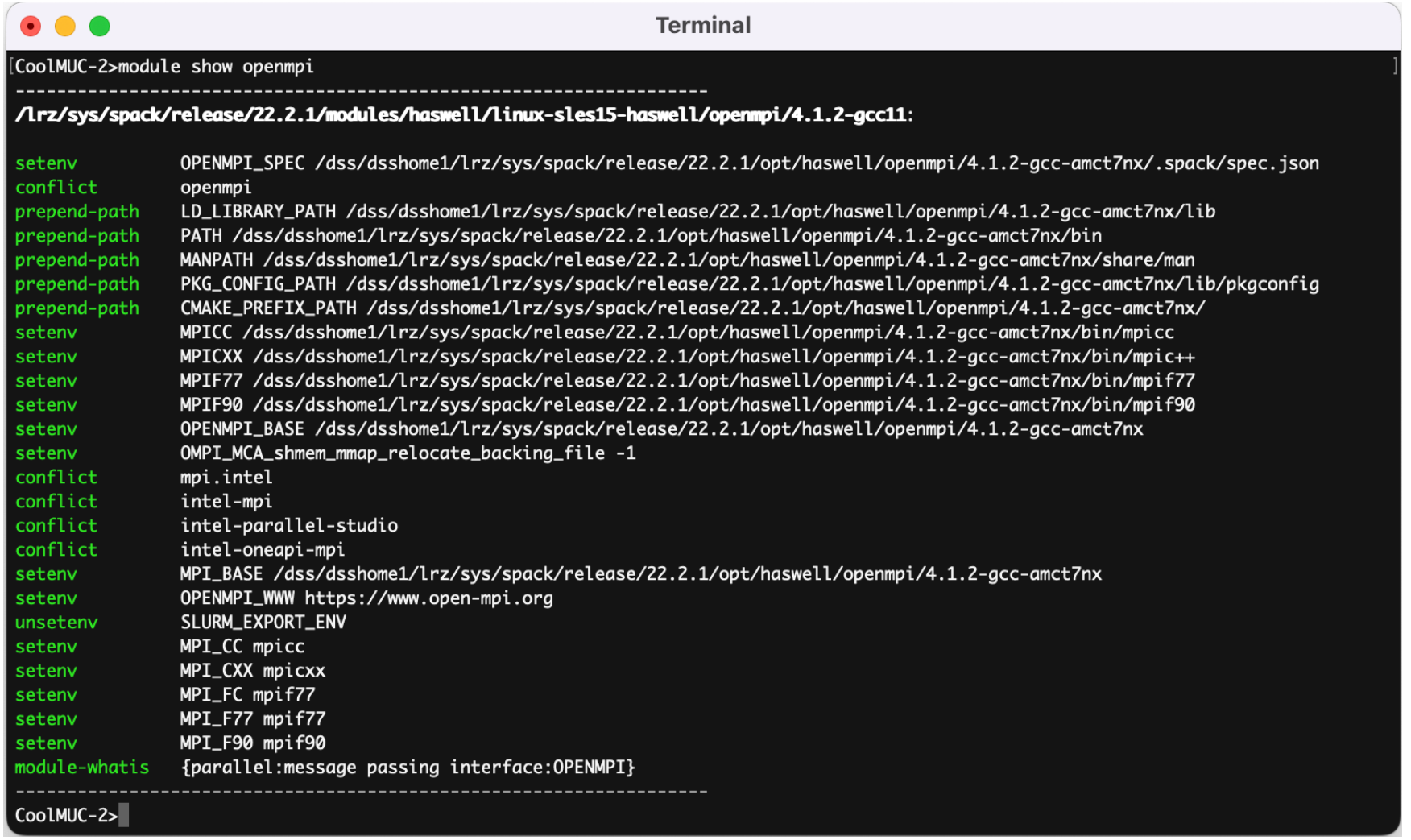
Examples of using modules



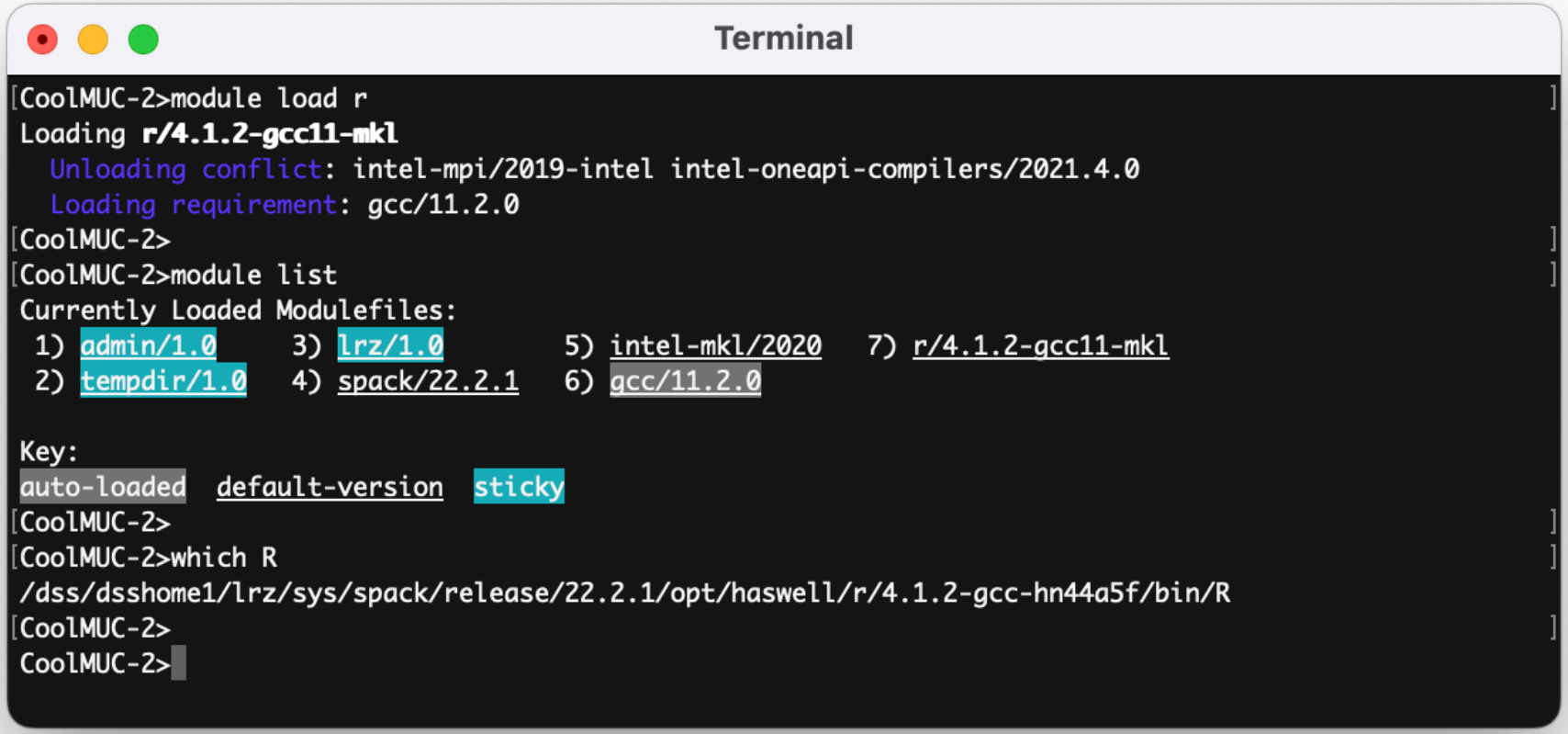
Examples of Modules



Examples of Modules



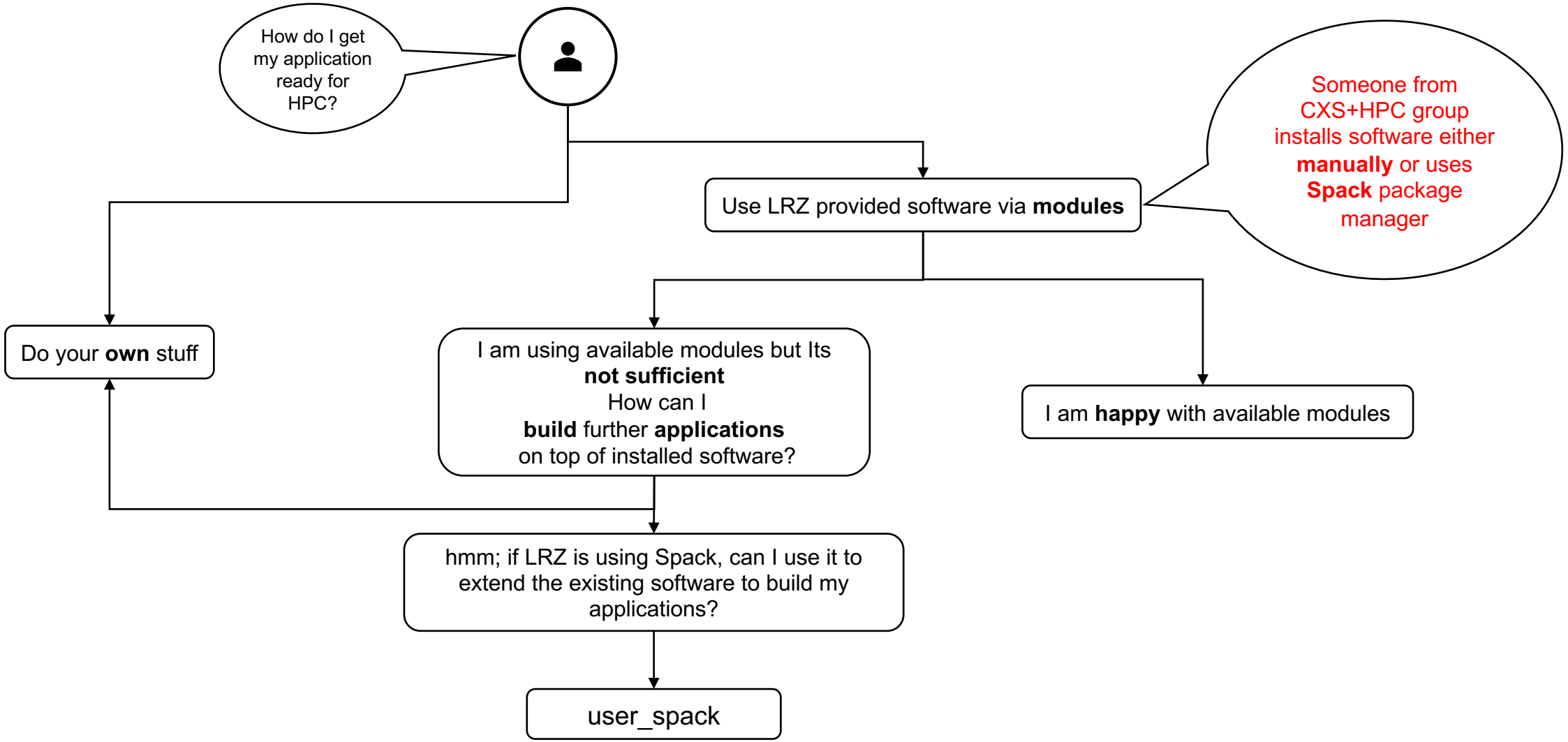
Examples of Modules



Screenshot of modules in the software stack

```
------------------------------------------------- /lrz/sys/spack/release/23.1.0/modules/compilers -------------------------------------------
gcc/8.5.0  gcc/10.4.0  gcc/12.2.0    intel/19.1.2  intel/2021.4.0  intel/2023.1.0  llvm/10.0.1  nag/7.1     nvhpc/22.11  nvhpc/23.3
gcc/9.5.0  gcc/11.3.0  intel/18.0.5  intel/20.0.4  intel/2022.2.0  llvm/9.0.1      llvm/16.0.2  nvhpc/22.9  nvhpc/23.1

------------------------------------------------- /lrz/sys/spack/release/23.1.0/modules/MPI -------------------------------------------
intel-mpi/2018.4.274   intel-mpi/2021.7.0   openmpi/3.1.6-intel23  openmpi/4.0.7-intel23  openmpi/4.1.5-intel23
intel-mpi/2019.12.320  intel-mpi/2021.9.0   openmpi/3.1.6-nag7.1   openmpi/4.0.7-nag7.1   openmpi/4.1.5-nag7.1
intel-mpi/2021.6.0     openmpi/3.1.6-gcc12  openmpi/4.0.7-gcc12    openmpi/4.1.5-gcc12

------------------------------------------------- /lrz/sys/spack/release/23.1.0/modules/x86_64 -------------------------------------------
ace/7.1.0              cube/4.8            imagemagick/7.0.8-7       mercurial/5.8                  perl/5.36.0              scons/3.1.2
anaconda3/2020.02      cuda/11.8.0         intel-mkl/2020.4.304      mesa/22.1.6                    pkg-config/0.29.2        scons/4.0.1
anaconda3/2022.10      cuda/12.0.0         intel-mkl/2021.4.0        miniconda3/22.11.1             pkgconf/1.8.0            scons/4.1.0.post1
autoconf/2.69          doxygen/1.9.6       intel-mkl/2023.1.0        molden/6.7                     plplot/5.15.0            scons/4.2.0
autoconf/2.71          dyninst/12.3.0      intel-toolkit/2021.4.0    nano/7.2                       prng/3.0.2               scons/4.5.2
automake/1.16.1        emacs/27.2-console  intel-toolkit/2023.1.0    ncurses/6.4                    protobuf/3.22.2          squashfs/4.5.1
automake/1.16.5        emacs/27.2-gtk      jmol/14.31.0              openjdk/1.8.0_202-b08          py-pymol/2.5.0           strace/5.19
autotools/v1           emacs/28.2-console  libszip/2.1.1             openjdk/11.0.17_8              py-testing-lrz/0.2.0     subversion/1.14.1
binutils/2.31.1        emacs/28.2-gtk      libtirpc/1.2.6            parallel/20220522              python/3.7.16-base       tcsh/6.24.00
bison/3.8.2            flex/2.6.3          libtool/2.4.6             paraview-prebuild/5.6.0        python/3.7.16-extended   texlive/2019
cgdb/0.8.0             gaussian/16-C.02    libtool/2.4.7             paraview-prebuild/5.6.0_mesa   python/3.8.16-base       tmux/3.3a
charliecloud/0.30      gdal/3.6.4          ltrace/0.7.3              paraview-prebuild/5.8.0        python/3.8.16-extended   vmd/1.9.3
charliecloud/0.32      gdb/13.1            m4/1.4.18                 paraview-prebuild/5.8.0_mesa   python/3.10.10-base      vmd/1.9.3-bash
clingo/5.4.0           git/2.40.0          m4/1.4.19                 paraview-prebuild/5.10.0       python/3.10.10-extended  xerces-c/3.2.1
cmake/3.14.5           gmake/4.4.1         matlab-mcr/R2022a_Update5 paraview-prebuild/5.10.0_mesa  qt/5.15.9-gl             yaml-cpp/0.7.0
cmake/3.26.3           gmp/6.2.1           matlab-mcr/R2022b_Update5 paraview-prebuild/5.11.0       qt/5.15.9-gl-gtk         zlib/1.2.13
coccinelle/gh-201904   grace/5.1.25        matlab-mcr/R2023a_Update1 paraview-prebuild/5.11.0_mesa  redis/7.0.5

------------------------------------------------- /lrz/sys/spack/release/23.1.0/modules/skylake_avx512 -------------------------------------
adios/1.13.1-gcc12-impi                hdf5/1.8.23-intel23-impi                numactl/2.0.14-intel23
adios2/2.9.0-intel23-impi              hdf5/1.10.9-gcc12                       openblas/0.3.23-gcc12
```

Flow chart to get my Scientific application ready for HPC



Spack is one of many package-managers

- Functional Cross-Platform Package Managers:
  e.g Nix (NixOs), Gnu Guix (Gnu Guix Linux) … use hashes in install-dirs

- Build-from-source Package Managers
  e.g. HomeBrew/LinuxBrew

- Package Managers for specific scripting languages
  e.g. Pip (Python), NPM (Javascript)

- Easy Build:
  installation framework for managing scientific software on HPC-systems

- Conda:
  popular binary package managers for Python and R (but also for other rpm–like packaging in user-space). Easy to use.
  In general no architecture optimized binaries, not targeted at HPC

If you want to use Spack,
you might want to know about it

What is Spack? In a nutshell

- Spack (package manager and not build tool) automates the fetch, configure and installation of scientific software.
- Spack works out of the box, Simply clone Spack repository to get going.

```
>git clone https://github.com/spack/spack
Cloning into 'spack'...
remote: Enumerating objects: 355317, done.
remote: Total 355317 (delta 0), reused 0 (delta 0), pack-reused 355317
Receiving objects: 100% (355317/355317), 159.99 MiB | 23.13 MiB/s, done.
Resolving deltas: 100% (151370/151370), done.
Updating files: 100% (9087/9087), done.
>
>. spack/share/spack/setup-env.sh
>
>spack install zlib
==> Installing zlib-1.2.11-fy5vijvec3tmqbdnsh4q3wkmqsqpzb5a
==> No binary for zlib-1.2.11-fy5vijvec3tmqbdnsh4q3wkmqsqpzb5a found: installing from source
==> Fetching https://mirror.spack.io/_source-cache/archive/c3/c3e5e9fdd5004dcb542feda5ee4f0ff0744628baf8ed2dd5d66f8ca1197cb1a1.tar.gz
==> No patches needed for zlib
==> zlib: Executing phase: 'install'
==> zlib: Successfully installed zlib-1.2.11-fy5vijvec3tmqbdnsh4q3wkmqsqpzb5a
  Fetch: 0.29s.  Build: 2.18s.  Total: 2.48s.
[+] /dss/dsshome1/09/di36pex/demo_spack/spack/opt/spack/linux-sles15-haswell/gcc-8.4.0/zlib-1.2.11-fy5vijvec3tmqbdnsh4q3wkmqsqpzb5a
```

What is Spack? In a nutshell

- Spack can install many different variants of the same package, to name a few:
  - package-versions
  - different compilers
  - different MPI-implementations
  - different build options

- Most important terminology using Spack is “**spec**”
  - Spec is what comes after “spack install” command.
  - Specs refer to a particular build configuration of a package.
  - “Specs” are more than package name. It can contain the compiler, compiler version, architecture, compile options, and dependency options for a build.

- Installation locations are separated via unique hashes
  - installations may peacefully coexist (dynamic linking with RPATH)

- The installation location of any package will also contain,
  - dump (in form of text files) of environment during installation, output from installation, configure arguments, and concretized spack-specs

What is Spack? In a nutshell

- Install a package
  - spack install hdf5
- Install a particular version by appending @
  - spack install hdf5**@**1.12.1
- Specify a compiler (and its version), with %
  - spack install hdf5**%**gcc**@**11.2.0
- Add special variants with +
  - spack install hdf5**@**1.12.1**%**gcc**@**11.2.0 +fortran +hl
- Add compiler flags using the conventional names
  - spack install hdf5**%**gcc**@**11.2.0 +cxx cppflags="-O3 -floop-block"
- Add micro-architecture with target (for cross compiling)
  - $ spack install hdf5**@**1.12.1**%**gcc target=skylake_avx512

Finding versions and variants of a package

- Spack will fetch all the information from a “package file” of a package.
  - ➢ Spack info <package name>

```
>spack info hdf5
CMakePackage:   hdf5

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.

Homepage: https://portal.hdfgroup.org

Maintainers: @lrknox @brtnfld @byrnHDF @ChristopherHogan @epourmal @gheber @hyoklee @lkurz @soumagne

Externally Detectable:
    False

Tags:
    e4s

Preferred version:
    1.12.1          https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.12/hdf5-1.12.1/src/hdf5-1.12.1.tar.gz

Safe versions:
    develop-1.13    [git] https://github.com/HDFGroup/hdf5.git on branch develop
    1.8.10          https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.8/hdf5-1.8.10/src/hdf5-1.8.10.tar.gz

Deprecated versions:
    None

Variants:
    Name [Default]              When    Allowed values          Description
    ==========================  ====    ====================    ==========================================

    api [default]               --      default, v114, v112,    Choose api compatibility for earlier version
                                        v110, v18, v16
    build_type [RelWithDebInfo] --      Debug, Release,         CMake build type
                                        RelWithDebInfo,
                                        MinSizeRel
    cxx [off]                   --      on, off                 Enable C++ support
    fortran [off]               --      on, off                 Enable Fortran support
    hl [off]                    --      on, off                 Enable the high-level library
    ipo [off]                   --      on, off                 CMake interprocedural optimization
    java [off]                  --      on, off                 Enable Java support
    mpi [on]                    --      on, off                 Enable MPI support
    threadsafe [off]            --      on, off                 Enable thread-safe capabilities
    tools [on]                  --      on, off                 Enable building tools

Installation Phases:
    cmake    build    install

Build Dependencies:
    cmake  java  mpi  numactl  szip  zlib

Link Dependencies:
    mpi  numactl  szip  zlib

Run Dependencies:
    java  pkgconfig
```

Concretization: dependency tree of a package

- Spack will fetch all the information from a “package file” of a package.
  - ➢ Spack info <package name>
- Listing a dependency graph before you go ahead with an installation
  - ➢ Spack spec –INtl <package name>
- Spack spec gives you full concretized map of the package

```
>spack spec -lINt hdf5 +fortran +hl
[Input spec
--------------------------------
 -   [    ]  .hdf5+fortran+hl

Concretized
--------------------------------
 -   qhat2bv  [    ]  builtin.hdf5@1.12.1%gcc@8.4.0~cxx+fortran+hl~ipo~java+mpi+shared~szip~threadsa
 -   i5dqbwv  [b   ]      ^builtin.cmake@3.22.2%gcc@8.4.0~doc+ncurses+openssl+ownlibs~qt build_type=
 -   lg6uazi  [bl  ]          ^builtin.ncurses@6.2%gcc@8.4.0~symlinks+termlib abi=none arch=linux-sl
 -   daqug65  [b r ]              ^builtin.pkgconf@1.8.0%gcc@8.4.0 arch=linux-sles15-haswell
 -   2zfrt75  [bl  ]          ^builtin.openssl@1.1.1m%gcc@8.4.0~docs certs=system arch=linux-sles15-
 -   zveogua  [b r ]              ^builtin.perl@5.34.0%gcc@8.4.0+cpanm+shared+threads arch=linux-sle
 -   t5a6kgl  [bl  ]                  ^builtin.berkeley-db@18.1.40%gcc@8.4.0+cxx~docs+stl patches=b2
 -   dpfohvz  [bl  ]                  ^builtin.bzip2@1.0.8%gcc@8.4.0~debug~pic+shared arch=linux-sle
 -   ji7jj7r  [b   ]                      ^builtin.diffutils@3.8%gcc@8.4.0 arch=linux-sles15-haswell
 -   rtusfmn  [bl  ]                          ^builtin.libiconv@1.16%gcc@8.4.0 libs=shared,static ar
 -   b3okknr  [bl  ]                  ^builtin.gdbm@1.19%gcc@8.4.0 arch=linux-sles15-haswell
 -   tjsehf3  [bl  ]                      ^builtin.readline@8.1%gcc@8.4.0 arch=linux-sles15-haswell
[+]  fy5vijv  [bl  ]                  ^builtin.zlib@1.2.11%gcc@8.4.0+optimize+pic+shared arch=linux-
 -   sklqwij  [bl  ]      ^builtin.numactl@2.0.14%gcc@8.4.0 patches=4e1d78cbbb85de625bad28705e748856
f599cfabf0740518b91ec8daaf18e8f288b19adaae5364dc1f6b2296 arch=linux-sles15-haswell
 -   kyq3asj  [b   ]          ^builtin.autoconf@2.69%gcc@8.4.0 patches=35c449281546376449766f92d49fc
5bac3b62daa0ff688ab4d508d71dbd2f4f8d7e2a02321926346161bf3ee arch=linux-sles15-haswell
 -   lx4bpfg  [b r ]              ^builtin.m4@1.4.19%gcc@8.4.0+sigsegv patches=9dc5fbd0d5cb1037ab1e6
89 arch=linux-sles15-haswell
 -   6zmn6dv  [bl  ]                  ^builtin.libsigsegv@2.13%gcc@8.4.0 arch=linux-sles15-haswell
 -   y77iduv  [b   ]          ^builtin.automake@1.16.5%gcc@8.4.0 arch=linux-sles15-haswell
 -   kyyaw3y  [b   ]          ^builtin.libtool@2.4.6%gcc@8.4.0 arch=linux-sles15-haswell
 -   l7ukpnd  [bl  ]      ^builtin.openmpi@4.1.2%gcc@8.4.0~atomics~cuda~cxx~cxx_exceptions+gpfs~inte
e+vt+wrapper-rpath fabrics=none schedulers=none arch=linux-sles15-haswell
 -   gzlecom  [bl  ]          ^builtin.hwloc@2.7.0%gcc@8.4.0~cairo~cuda~gl~libudev+libxml2~netloc~nv
 -   qxup3jg  [bl  ]              ^builtin.libpciaccess@0.16%gcc@8.4.0 arch=linux-sles15-haswell
 -   kg3gaar  [b   ]                  ^builtin.util-macros@1.19.3%gcc@8.4.0 arch=linux-sles15-haswel
 -   wwlegep  [bl  ]              ^builtin.libxml2@2.9.12%gcc@8.4.0~python arch=linux-sles15-haswell
 -   4syc2vd  [bl  ]                  ^builtin.xz@5.2.5%gcc@8.4.0~pic libs=shared,static arch=linux-
 -   4sogqx7  [bl  ]          ^builtin.libevent@2.1.12%gcc@8.4.0+openssl arch=linux-sles15-haswell
 -   jitrjio  [  r ]          ^builtin.openssh@8.8p1%gcc@8.4.0 arch=linux-sles15-haswell
 -   jkppbin  [bl  ]              ^builtin.libedit@3.1-20210216%gcc@8.4.0 arch=linux-sles15-haswell
```

Useful Spack commands

```
$ spack help
usage: spack [-hkV] [--color {always,never,auto}] COMMAND ...

A flexible package manager that supports multiple versions,
configurations, platforms, and compilers.

These are common spack commands:

query packages:
  list                  list and search available packages
  info                  get detailed information on a particular package
  find                  list and search installed packages

build packages:
  install               build and install packages
  uninstall             remove installed packages
  gc                    remove specs that are now no longer needed
  spec                  show what would be installed, given a spec

configuration:
  external              manage external packages in Spack configuration

environments:
  env                   manage virtual environments
  view                  project packages to a compact naming scheme on the filesystem.

create packages:
  create                create a new package file
  edit                  open package files in $EDITOR

system:
  arch                  print architecture information about this machine
  audit                 audit configuration files, packages, etc.
  compilers             list available compilers
```

Can I use Spack
to extend the existing software stack?

• User_spack: Spack in User space

- Chaining the existing Installations (software stack provided by the LRZ) into your own Spack Environment
- How do I activate user_spack?
  - ➢ module load user_spack
- Why do I use it?
  - making use of already installed packages via chaining of software stack provided by the LRZ
  - avoids recompiling low level packages in many situations
  - has working defaults configurated for some essential dependencies (e.g. MPI)
- What does loading “user_spack” module do actually?
  - loads a setup script that adds the “spack” command to your shell.
  - spack version matches the version the default software stack has been built with
  - LRZ specific configurations are preconfigured for you

• User_spack: Installing software

- HDF5 dependency tree is shown in the image.
- The library itself is not yet installed (-).
- But all of its dependencies are already available via the upstream LRZ installation ([^]).
- A package that you have installed locally in your home directory with Spack is marked by [+].

```
>spack spec -INlt hdf5 +fortran ~threadsafe
Input spec
--------------------------------
 -   [    ]  .hdf5+fortran~threadsafe

Concretized
--------------------------------
 -   ohhl2i4  [    ]  builtin.hdf5@1.10.7%gcc@11.2.0~cx
h=linux-sles15-haswell
[^]  kdeymh2  [b   ]      ^builtin.cmake@3.21.4%gcc@11.
[^]  47oirry  [bl  ]          ^builtin.ncurses@6.2%gcc@
[^]  6dqwgxm  [b r ]              ^builtin.pkgconf@1.8.
[^]  oeopnpj  [bl  ]          ^builtin.openssl@1.1.1l%g
[^]  rken265  [b r ]              ^builtin.perl@5.34.0%
[^]  dp5zt2x  [bl  ]                  ^builtin.berkeley-
05aff41de522 arch=linux-sles15-haswell
[^]  ybteog6  [bl  ]                  ^builtin.bzip2@1.
[^]  gweivqh  [b   ]                      ^builtin.diff
[^]  q6m4kz6  [bl  ]                          ^builtin.
```

• User_spack: Installing software

- Dependencies are chained from the LRZ installation
  - see /dss/…/lrz/… paths
- Spack checks if the source tar ball is available in the LRZ cache
- If not present, tar files can be download from external site
- installation location of the library in $HOME/spack/opt/...

```
>spack install hdf5 +fortran ~threadsafe
[+] /dss/dsshome1/lrz/sys/spack/release/22.2.1/opt/haswell/pkg
[+] /dss/dsshome1/lrz/sys/spack/release/22.2.1/opt/haswell/be
[+] /dss/dsshome1/lrz/sys/spack/release/22.2.1/opt/haswell/li
[+] /dss/dsshome1/lrz/sys/spack/release/22.2.1/opt/haswell/zl
[+] /dss/dsshome1/lrz/sys/spack/release/22.2.1/opt/haswell/au
[+] /dss/dsshome1/lrz/sys/spack/release/22.2.1/opt/haswell/cm
[+] /dss/dsshome1/lrz/sys/spack/release/22.2.1/opt/haswell/au
[+] /dss/dsshome1/lrz/sys/spack/release/22.2.1/opt/haswell/nu
==> Installing hdf5-1.10.7-ohhl2i4g4ztco53okwghc5zcre332lk2
==> No binary for hdf5-1.10.7-ohhl2i4g4ztco53okwghc5zcre332lk
==> Fetching file:///lrz/sys/spack/cache/_source-cache/archive
==> Ran patch() for hdf5
==> hdf5: Executing phase: 'cmake'
==> hdf5: Executing phase: 'build'
==> hdf5: Executing phase: 'install'
==> hdf5: Successfully installed hdf5-1.10.7-ohhl2i4g4ztco53o
  Fetch: 0.04s.  Build: 2m 2.08s.  Total: 2m 2.12s.
[+] /dss/dsshome1/09/di36pex/spack/opt/linux-sles15-haswell/h
```

User_spack: Generating Modules for your software

- We have configured Spack such that modules will only be generated for explicitly installed packages.

```
>spack module tcl refresh hdf5
==> You are about to regenerate tcl mo

-- linux-sles15-haswell / gcc@11.2.0 --
ohhl2i4 hdf5@1.10.7

==> Do you want to proceed? [y/n] y
==> Regenerating tcl module files
```

- Modules are installed in
  - $HOME/spack/modules/$LRZ_INSTRSET/ <architecture>/<package>/<version>
- The path to the modules is added to $MODULEPATH when you load the user_spack module, but only if it already exists. You might want to reload user_spack.
- One could also add module path manually
  - ➢ module use <path to local modules>

User_spack: Configuring your Spack Instance

- LRZ provides a configuration that is very similar to the one the software stack was built with.
- You may want to change some or all of these settings to serve your needs, e.g. for the package selection, generation of modules, etc.
- Your individual configuration files are stored in the directory ~/.spack/.
  - spack config edit repos
  - spack config edit config
  - spack config edit modules
- User config files take precedence over system provided config file, that is they are loaded after the system config files and overwrite their settings.

User_spack: Generating modules for upstream packages

- For packages that have been implicitly installed in the upstream software stack (LRZ installed) no modules are generated by default.
- You can configure your `user_spack` such that you generate modules in your $HOME directory.
- You may want to change some or all of these settings to serve your needs, e.g. for the package selection, generation of modules, etc.
  - spack config edit modules
  - cat ~/.spack/modules.yaml

```
modules:
  tcl:
    exclude_implicits: false
    hash_length: 7
```

- Now modules are generated for all installed packages and each newly created module gets a hash suffix of length 7 to avoid naming conflicts.
- Modules are generated with
  - spack module tcl refresh --upstream-modules

User_spack: User configurations

- **Adding compilers**
- Spack searches for compilers on your machine automatically the first time it is run. It does this by inspecting your $PATH
- One could use “spack compiler add” command,
  - spack compiler add /path/to/my_compiler
  - spack compiler list
- For example the output of spack compiler list could look like,

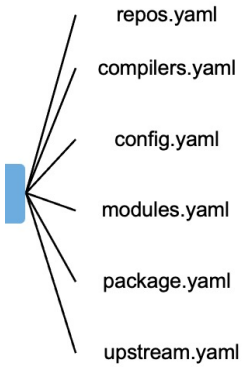
```
==> Available compilers

-- gcc sles15-x86_64 ---------------------------------------------
gcc@8.4.0  gcc@7.5.0


-- intel sles15-x86_64 -------------------------------------------
intel@19.0.5.281
```

- **Spack package repositories**
- Spack supports external package repositories
  - Separate directories of package files
- Many reasons for doing this,
  - You want to write a package file for an in-house code that you may not want to release publicly.
  - Overwrite default package files with site specific versions or restrictions
- One could use “spack repo create” command,
  - spack repo create /path/to/my_repo
  - spack repo add my_repo
  - spack repo list
- This will show 2 package repositories.
  - my_repo /path/to/my_repo builtin
  - spack/var/spack/repos/builtin

Chaining: Chain Spack Installations

- You can point your Spack installation to another installation to use any packages that are installed there.
- To register the other Spack instance, you can add it as an entry to upstreams.yaml

```
upstreams:
  spack-instance-1:
    install_tree: /path/to/other/spack/opt/spack
  spack-instance-2:
    install_tree: /path/to/another/spack/opt/spack
```




```
>spack spec -INlt hdf5 +fortran~hl
Input spec
--------------------------------
 -   [    ]  .hdf5+fortran~hl

Concretized
--------------------------------
 -   zt4lxrl  [    ]  fixes015x.hdf5@1.10.7%gcc@8.4.0~cxx~debug+fc
[^]  cyojcvv  [bl   ]      ^fixes015x.intel-mpi@2019.8.254%gcc@8.4.
[^]  wz47lgr  [bl   ]      ^builtin.numactl@2.0.12%gcc@8.4.0 arch=l
[^]  jns7liw  [b    ]          ^builtin.autoconf@2.69%gcc@8.4.0 arc
[^]  6vxvnrt  [b r  ]              ^builtin.m4@1.4.18%gcc@8.4.0+sig
c8 arch=linux-sles15-haswell
[^]  gv36h32  [bl   ]                  ^builtin.libsigsegv@2.12%gcc
[^]  bhpjih4  [b r  ]              ^builtin.perl@5.30.3%gcc@8.4.0+c
[^]  szzheyp  [bl   ]                  ^builtin.gdbm@1.18.1%gcc@8.4
[^]  3kfx6pu  [bl   ]                      ^builtin.readline@8.0%gc
[^]  6qhv5ta  [bl   ]                          ^fixes015x.ncurses@6
[^]  cfijkws  [b    ]                              ^builtin.pkgconf
[^]  zzoup2h  [b    ]          ^builtin.automake@1.16.2%gcc@8.4.0 a
[^]  4nya677  [b    ]          ^builtin.libtool@2.4.6%gcc@8.4.0 arc
[^]  m2bfsoy  [bl   ]      ^builtin.zlib@1.2.11%gcc@8.4.0+optimize+
```

Spack Environments

- A spack environment is used to group together a set of specs for the purpose of building, rebuilding and deploying in a coherent fashion.
- An Environment that is built as a whole can be loaded as a whole into the user environment.
- **spack.yaml** (example: python-extended.yaml) describes a project requirements
  - Spack stores metadata in the .spack-env directory. User interaction will occur through the spack.yaml
  - When the environment is concretized, Spack will create a file spack.lock. This file describes exactly what versions /configurations were installed, allows them to be reproduced.
- You can give this file to any one in the project and he would get the exact same customized sets of packages installed, without any differences. A very robust reproducible software environment!

```yaml
spack:
  config:
    install_tree:
      root: $spack/../opt/${LRZ_TARGET}/autootols/v1
  specs:
    - autotools@v1
    - autoconf@2.71
    - automake
    - libtool
    - m4
    - pkg-config
    - autoconf@2.69
    - automake@1.16.1
    - libtool@2.4.6
    - m4@1.4.18

  view:
    autotools:
      root: ../../../views/autotools/v1
      exclude:
        - autoconf@2.69
        - automake@1.16.1
        - libtool@2.4.6
        - m4@1.4.18
  concretizer:
   unify: false
```

Conda Environments

- What is Conda?
  - A package manager and environment management system.
  - Ideal for creating isolated environments for projects.
- Benefits of Using Conda Environments
  - Avoids version conflicts between libraries.
  - Ensures project reproducibility.
- Creating and Activating a Conda Environment
  - Create: **conda create --name myenv python=3.8**
  - Activate: **source activate myenv**
- Installing mpi4py
  - mpi4py allows Python programs to use MPI for parallel processing.
- Install with: **conda install mpi4py**
- Exporting and Sharing Environments
  - Export with: conda env export > environment.yml.
  - Share the environment.yml for reproducible setups.

```
di36pex@cm2login1:~> conda activate myenv
(myenv) conda install mpi4py
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 22.9.0
  latest version: 23.10.0

Please update conda by running

    $ conda update -n base -c defaults conda



## Package Plan ##

  environment location: /dss/dsshome1/09/di36pex/.conda/envs/myenv



    xz-5.4.2                  |       h5eee18b_0         642 KB
    zlib-1.2.13               |       h5eee18b_0         103 KB
    ------------------------------------------------------------
                                           Total:        51.5 MB

The following NEW packages will be INSTALLED:

  _libgcc_mutex      pkgs/main/linux-64::_libgcc_mutex-0.1-main None
  _openmp_mutex      pkgs/main/linux-64::_openmp_mutex-5.1-1_gnu None
  bzip2              pkgs/main/linux-64::bzip2-1.0.8-h7b6447c_0 None
  ca-certificates    pkgs/main/linux-64::ca-certificates-2023.08.22-h06a4308_0 None
  ld_impl_linux-64   pkgs/main/linux-64::ld_impl_linux-64-2.38-h1181459_1 None
  libffi             pkgs/main/linux-64::libffi-3.4.4-h6a678d5_0 None
  libgcc-ng          pkgs/main/linux-64::libgcc-ng-11.2.0-h1234567_1 None
  libgfortran-ng     pkgs/main/linux-64::libgfortran-ng-7.5.0-ha8ba4b0_17 None
  libgfortran4       pkgs/main/linux-64::libgfortran4-7.5.0-ha8ba4b0_17 None
  libgomp            pkgs/main/linux-64::libgomp-11.2.0-h1234567_1 None
  libstdcxx-ng       pkgs/main/linux-64::libstdcxx-ng-11.2.0-h1234567_1 None
  libuuid            pkgs/main/linux-64::libuuid-1.41.5-h5eee18b_0 None
  mpi                pkgs/main/linux-64::mpi-1.0-mpich None
```

Software Stack on SMNG Phase 2: Spack/24.1.0

- Rolled out Software Stack Spack/24.1.0 on SuperMUC-NG Phase II.

- Compilers and MPI's:
  - OneAPI 24x release, support for Intel PVC, will be made available.
  - AI toolkit from OneAPI will be made available.
  - Intel compiler drivers – LLVM based (e.g., icx, ifx, icpx, etc.) will replace traditional drivers (ifort, icpc, and icc).
  - Intel oneAPI AI toolkits for AI BD workloads will be provided as modules.
  - Both, generic build and optimized software builds for Sapphirerapids will be made available

- Improved Module Interactions
  - We have made significant changes to enhance the maintainability and long-term support of the software stack, particularly in terms of module interactions.
    - Simplified module names
    - Adhering the compatibility of compilers and MPI with all its dependents by adding meaning prerequisites
    - Have provided bundle modules; collection of frequently used software in a single module.

- `user_spack` enabling the possibility to address unique needs of users to install / build on top of LRZ software stack.

Questions?