OAK RIDGE National Laboratory

National Center for Computational Sciences Software Provisioning (NSP)

Asa Rentschler, Nicholas Hagerty, Elijah Maccarthy and Fernando Posada

Systems Acceptance & User Environment

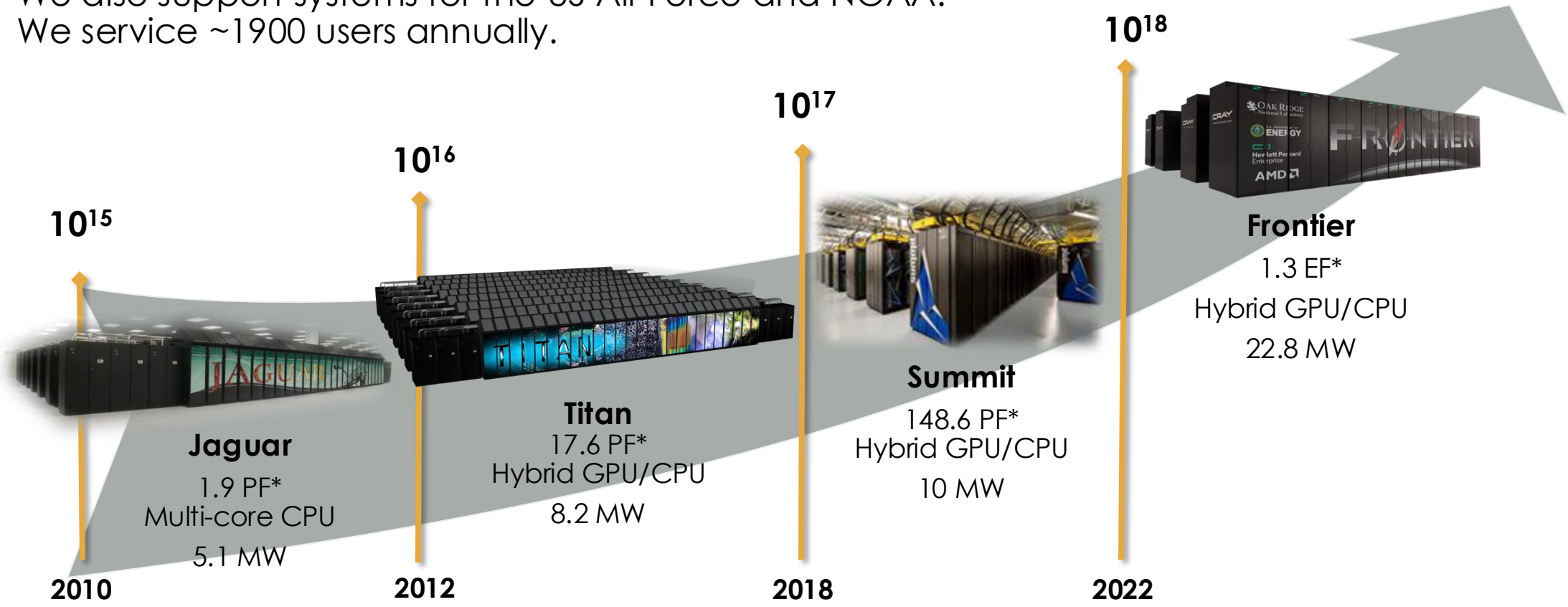
Oak Ridge National Laboratory

ORNL is managed by UT-Battelle LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

HPC History at NCCS

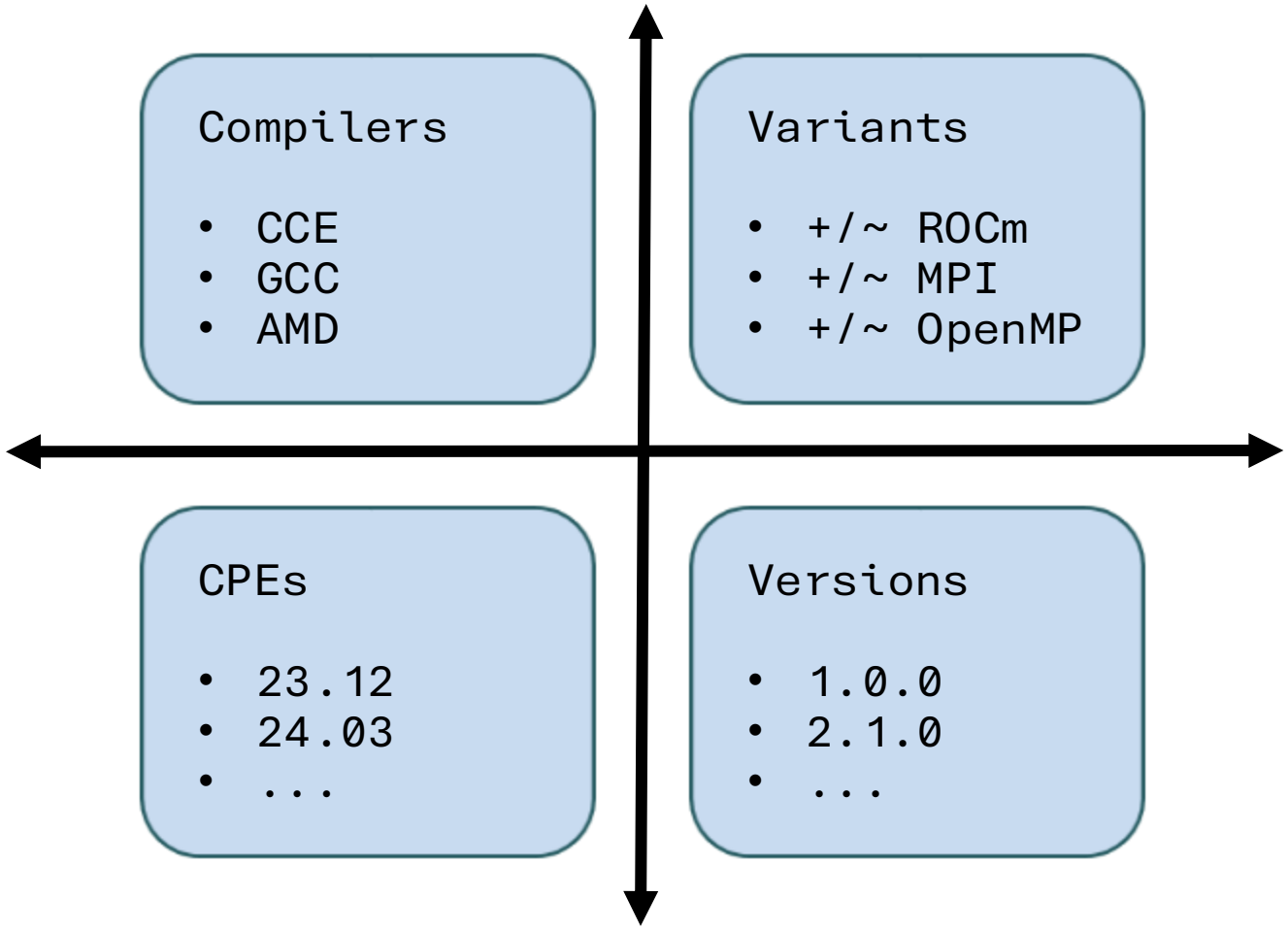
- NCCS host the Oak Ridge Leadership Computing Facility (OLCF) which provides leadership scale computing resources for open science.
- We also support systems for the US Air Force and NOAA.
- We service ~1900 users annually.



* Latest available TOP500 double-precision HPL achieved performance

Software Stack Complexity on Frontier

- Recent projects such as ECP*/E4S* have dramatically increased the amount of available software for large scale computing.
- We maintain a stack of around ~100 packages.
- All these factors combine to create a software stack of combinatorial complexity.



*Exascale Computing Project (ECP)
*Extreme-Scale Scientific Software Stack (E4S)

Difficulties in Deploying Software

Integrating Site and Vendor Modules

- Module files for compilers, MPI and GPU libraries are managed by Cray PE.
- The paths that Spack generates for modules are practically random because of hashes.
- Spack does not support "non-virtual" specs (like ROCm) in the module hierarchy.

Managing Environments in Spack

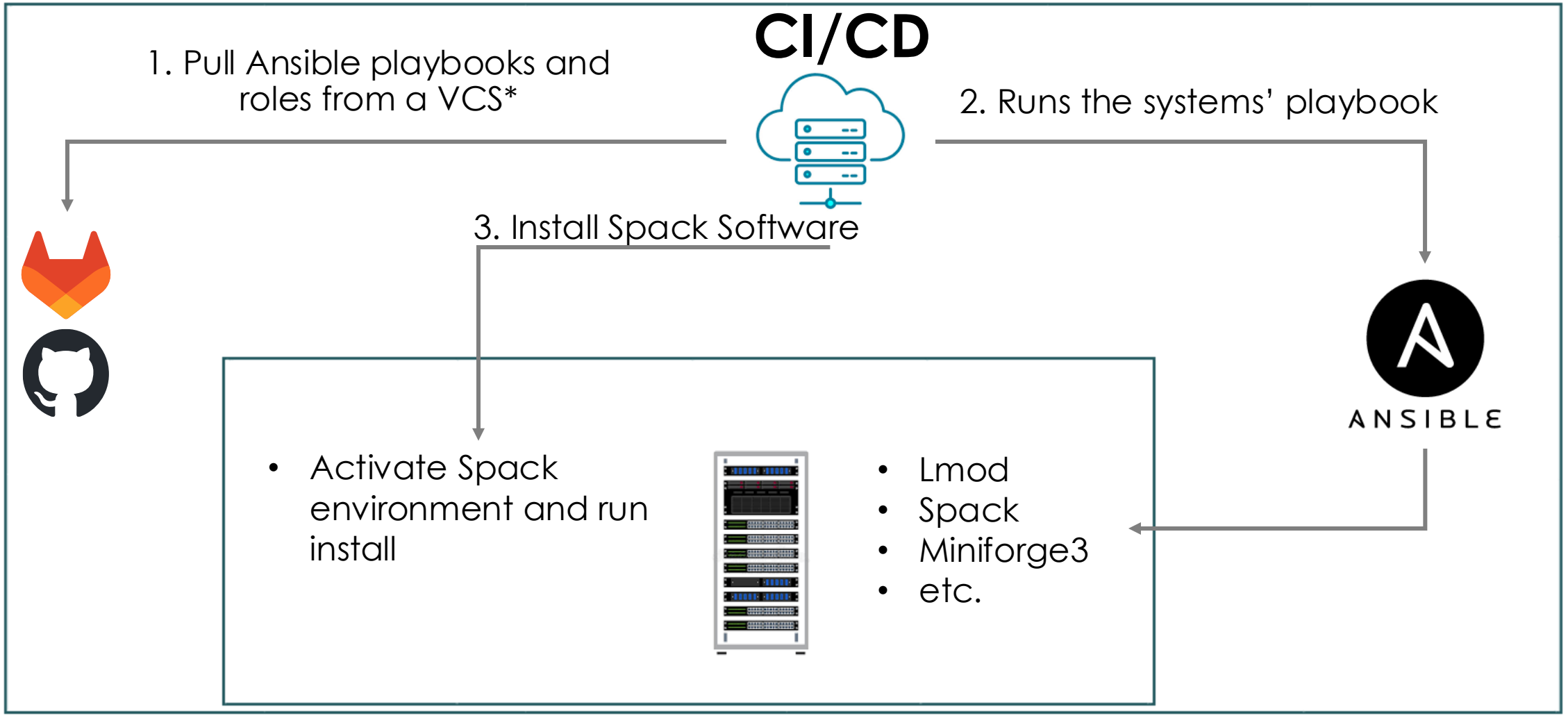
- Multiple CPE version on the machine at the same time can lead to a BIG spack.yaml file (e.g 1800+ lines).
- Managing transitions to new versions of Spack can cause an environment to become messy.

Open slide master to edit

What is NSP?

- A collection of tools and strategies for software stacks.
  - Trackable
  - Reproducible
- Tools:
  - Ansible
  - Spack
  - Lmod
- Strategies:
  - Breaking the software stack into multiple templated Spack environments.
  - Configuring Spack to create custom projections for our module files.
  - Using Lmod hooks to integrate CPE and site modules.

NSP Workflow



*Version Control System

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

OpenSlide master to edit

Ansible Roles and Configuration

NSP Ansible Roles
(Public)

Ansible Playbooks for NCCS Systems
(Private)

```
.
├── ansible.cfg
├── spack
│   ├── defaults
│   ├── meta
│   ├── tasks
│   └── templates
├── lmod
│   ├── defaults
│   ├── meta
│   ├── tasks
│   └── templates
...
```

```
.
├── ansible.cfg -> roles/ansible.cfg
├── odo
│   ├── files
│   ├── playbook.yaml
│   └── spack
├── frontier
│   ├── files
│   ├── init
│   ├── lmod
│   ├── playbook.yaml
│   └── spack
├── README.md
└── roles (.gitmodules)
```


Open slide master to edit

NSP Ansible Roles

- Install/configure the respective piece of software.
- We ensure that these roles follow certain guidelines such as uniform install prefixes.
- We maintain a variety of roles in ansible.
  - Spack
  - Lmod
  - GCC
  - Miniforge3
  - …

Open slide master to edit

The Spack & Lmod Roles

Spack

- Clones and patches a version(s) of Spack.
- Renders templated Spack environments.
- Renders a templated help script called "***spacktivate***."
  - Selects the correct version of spack and activates the requested environment.

Lmod

- Implements a hook to dynamically add paths to ***MODULEPATH*** based on the currently loaded modules.
- Implements a custom ***spider*** to show all available Spack built software.
- Logs module usage to an HTTP API end point.

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY
Open slide master to edit

Templating Spack Environments

```
cray_dyninst: 12.3.4
cray_fftw: 3.3.10.9
cray_hdf5: 1.14.3.3
cray_hdf5_parallel: 1.14.3.3
cray_mrnet: 5.1.4
```

+

```
packages:
  cray-dyninst:
    buildable: false
    externals:
     - spec: cray-dyninst@{{ cray_dyninst }}
       modules:
         - cray-dyninst/{{ cray_dyninst }}
  cray-fftw:
    buildable: false
    externals:
      - spec: cray-fftw@{{ cray_fftw }}
        modules:
          - cray-fftw/{{ cray_fftw }}
```

=

```
packages:
  cray-dyninst:
    buildable: false
    externals:
      - spec: cray-dyninst@12.3.4
        modules:
          - cray-dyninst/12.3.4
  cray-fftw:
    buildable: false
    externals:
      -  spec: cray-fftw@3.3.18.9
        modules:
          - cray-fftw/3.3.18.9
```


OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Open slide master to edit

Module Projections

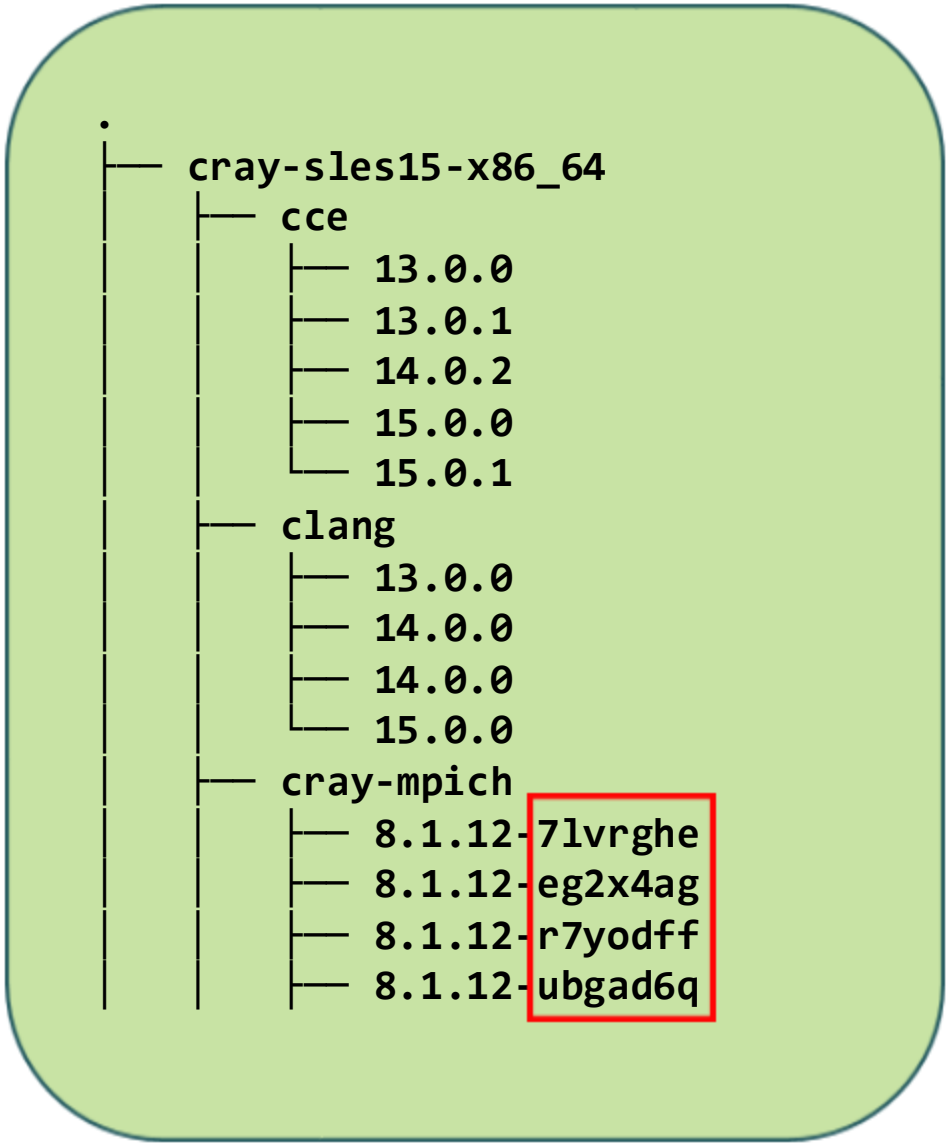
- A module ***projection*** is simply the path to the module file.
  - Built with ***gcc/14.2.0***: /.../gcc-14.2.0/hdf5/1.14.5.lua
  - Built with ***llvm/19.1.1***: /.../llvm-19.1.1/hdf5/1.14.5.lua
- We call this changing part of the projection the ***hierarchy***.
- We generally have three components in our hierarchy.
  - Compiler
  - MPI
  - GPU
- Example: /.../cray-mpich-8.1.31/rocm-6.2.4/cce-18.0.1/amrex/24.05.lua

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Open slide master to edit

Default Module Projections with Spack

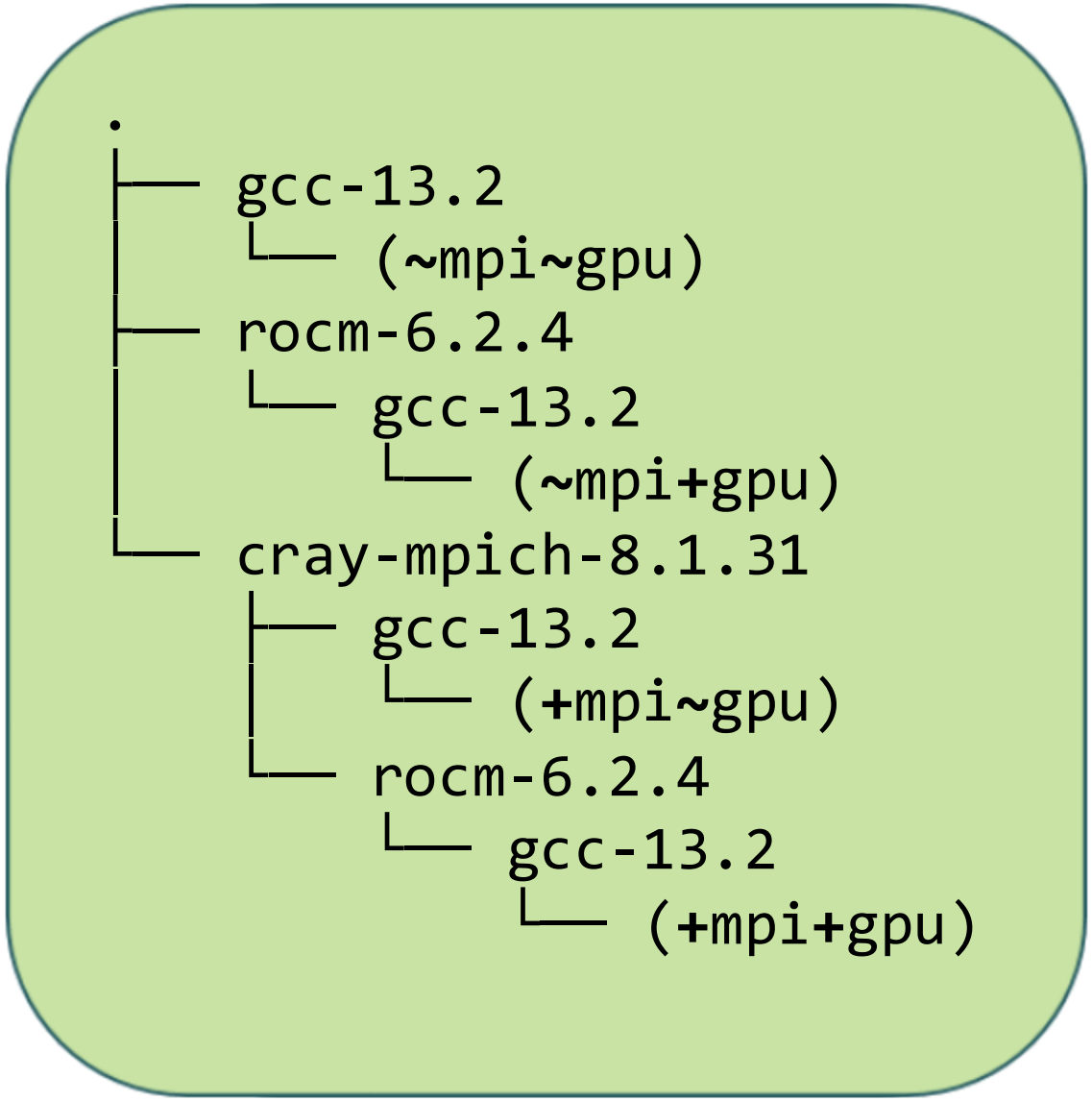
- Spack adds hashes to the module projections (sometimes multiple hashes for the same version of a package).
- Because Spack doesn't generate our cray-mpich modules, all these paths must be added manually as patches to the CPE module files.
- Spack's hierarchy only supports "virtual" packages which excludes ROCm from the hierachy.



OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY
Open slide master to edit

NSP Projections

- NO hashes.
- Paths to modules do not have to be known ahead of time. They can be dynamically constructed.
- ROCm is in the hierarchy.



OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Open slide master to edit

Configuring Spack Projections

```
arch_folder: false
lmod:
  all:
    autoload: none
    exclude_implicits: true
  hash_length: 0
  hierarchy:: [ ]
  projections:
    ^llvm-amdgpu ^mpi: "{^mpi.name}-{^mpi.version}/rocm-{^llvm-amdgpu.version}/..."
    ^llvm-amdgpu: "rocm-{^llvm-amdgpu.version}/{compiler.name}..."
    ^mpi: "{^mpi.name}-{^mpi.version}/{compiler.name}-{compiler.version}/..."
    all: "{compiler.name}-{compiler.version}/{name}/{version}"
```


Open slide master to edit

Lmod Hooks

- A ***hook*** is a function that is called at a specific point in Lmod's execution (e.g. the ***load*** hook is called whenever a module is loaded).
- Hooks are stored in a file called ***SitePackage.lua***.

```
local Hook = require("Hook")

function my_hook(t)
    -- do something
end

Hook.register("load", my_hook)
```

- The location of this file can be set with ***LMOD_PACKAGE_PATH***.

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Open slide master to edit

Configuring the NSP Hook

- The NSP hook watches for changes in modules and add/removes paths to Spack built modules based on the provided path projections.

```
NSP_LMOD_spack_modules: "{{ path to spack modules }}"
NSP_LMOD_hierarchy:
    compiler:
        members: [ 'cce', 'amd', 'gcc-native' ]
        paths:
            - { path: '|compiler.name|-|compiler.version|', weight: 20 }
            - { path: '|mpi.name|-|mpi.version|/|compiler.name|-|compiler.version|', weight: 30 }
            - { path: '|gpu.name|-|gpu.version|/|compiler.name|-|compiler.version|', weight: 40 }
    gpu:
        members: [ 'rocm' ]
        paths:
        ...
    mpi:
        members: [ 'cray-mpich' ]
        ...
NSP_LMOD_nv_mappings:
    # '%s' substitutes in the value provided by the module
    amd/6.2.4: { name: 'rocmcc', version: '%s' }
    gcc-native/13.2: { name: 'gcc', version: '%s' }
```


Open slide master to edit

Integrating Spack Modules into `module spider`

- Dynamic additions to ***MODULEPATH*** do not allow Lmod to track ***ALL*** available modules.
- We wrote a custom spider command integrated into Lmod (via a Hook).
  - Uses the default spider method to generate a cache of all non-Spack installed software.
  - Crawls our Spack installed software to generate an additional cache.
  - Combines the two caches and prints out the requested info to the user.

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY
Open slide master to edit

NSP Take Aways

- Ansible
  - Template configuration files (especially Spack config files).
  - Track changes to system playbooks through version control.
  - Dramatically improves the ease of reproducing a software stack!
- Spack
  - Break software stacks into multiple environments.
  - Leverage projections to create a custom module layout.
- Lmod
  - Use hooks to create a responsive module environment integrating site and vendor modules.

Open slide master to edit


2025 OLCF
User Meeting
August 5-6, 2025
• Facility updates & expert talks
• Deep-dive discussions with OLCF staff
• Networking with top computational scientists
Register Today!
www.olcf.ornl.gov/olcf-user-meeting
OAK RIDGE National Laboratory

Questions?

- GitHub: https://github.com/olcf/nccs-software-provisioning
- Docs: https://olcf.github.io/nccs-software-provisioning
- Email: rentschleraj@ornl.gov

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.