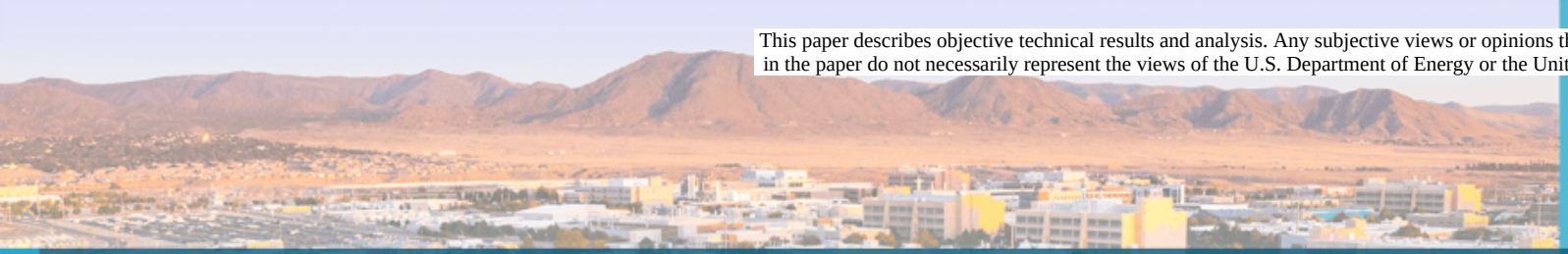


Sandia National Laboratories

SAND2021-6190C

Spack Configuration Manager: Automating Toolchain Installations

Joe Frye; Miranda Mundt; Henry Swantner; Jon Pellegrini

US-RSE Virtual Workshop, May 2021

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Overview

- Department of Software Engineering and Research at Sandia National Labs
  - Software engineering and research for scientific computation
  - Software engineering principles and best practices
- Spack from Lawrence Livermore National Labs
  - Package manager for HPCs
  - High level of complexity for users

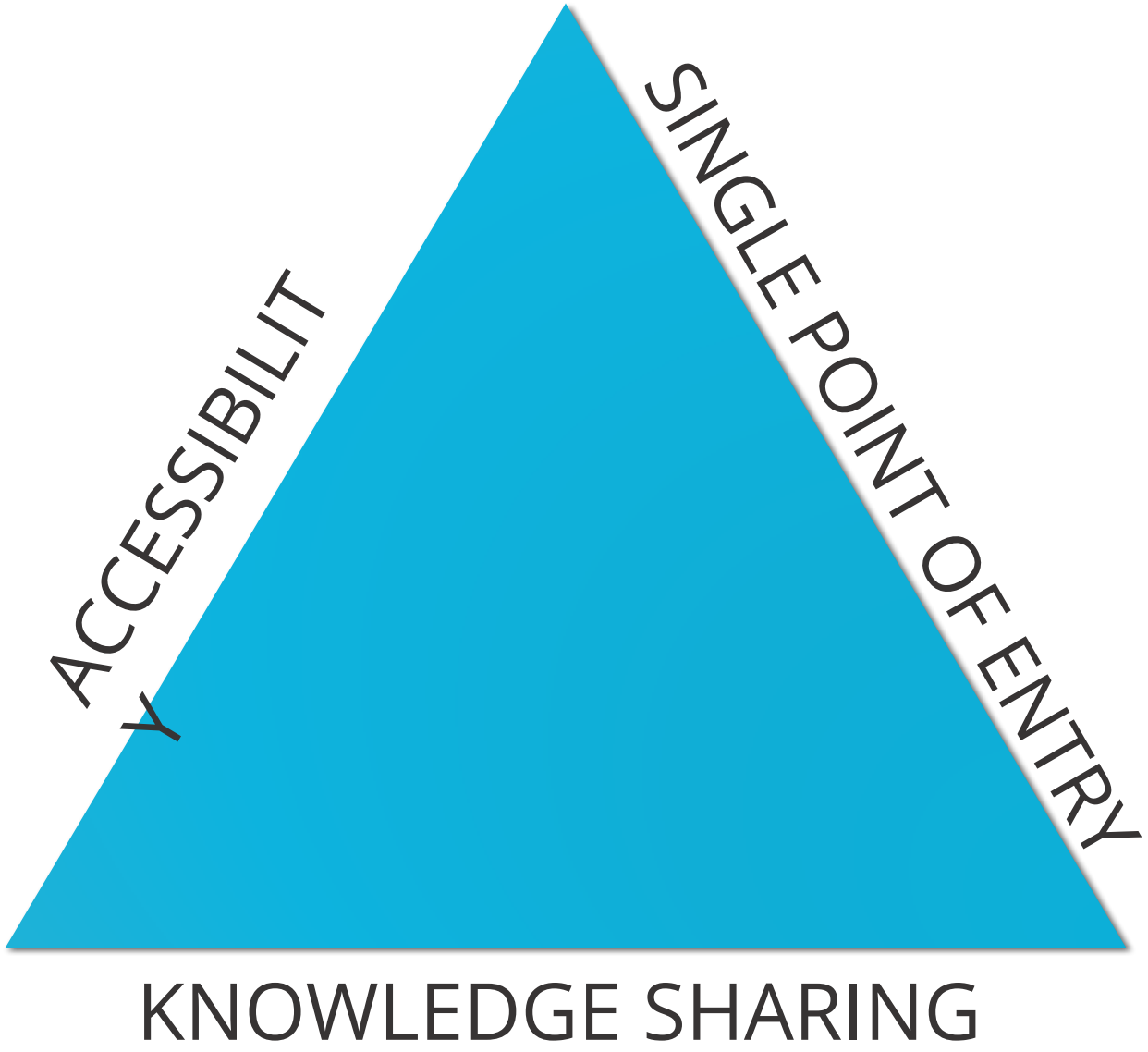
Software Engineering & Research

Department 1424

History

- Originally created a homegrown tool for third-party library installation
  - Spack was in its infancy
  - Consistent environment across multiple platforms
  - High maintenance cost
- Spack improvements over the years became an attractive option
  - More maintainers / contributors
  - More platform support
  - Package configuration lives with a maintainer, not the tool creator
- Needed to handle the complexities introduced by Spack

Three Core Requirements



Tool Details

- Dependencies
  - Spack
  - Python 3.6+
  - PyYAML

- Light wrapper around Spack
  - Utilizes core mechanisms of Spack
  - Remove knowledge barrier for configuration specifics

DEMO: Setup

```
(env_python) (base)  $ spack-cm setup -h
usage: spack-cm setup [-h] [-p PROJECT] [-m ALTHOSTNAME] [--spack SPACKVERSION]

Run set up routine for a new project/machine combination.

optional arguments:
  -h, --help            show this help message and exit
  -p PROJECT, --project PROJECT
                        REQUIRED: Project for which to install TPLs (e.g., trilinos, pyomo, etc.).
  -m ALTHOSTNAME, --machine ALTHOSTNAME
                        OPTIONAL: Designate an alternate machine name (i.e., not the hostname of the machine).
  --spack SPACKVERSION  OPTIONAL: Version of spack. Default: 0.16.1
```

DEMO: Manifest

BASE_COMPILER: Compiler to install base packages
BASE_PACKAGES: Base packages that will be installed once/used in later steps
COMPILERS: Compilers to install
EXTERNAL_COMPILERS: Compilers already on the system to use
UTILITY_COMPILER: Compiler to install utilities
UTILITIES: Utilities (single-installation packages like Cmake, git)
MPIS: MPIs to install
EXTERNAL_MPIS: MPIs already on the system to use
CUDAS: Cudas to install
EXTERNAL_CUDAS: Cudas already on the system to use
TPLS: Third-party library packages to install (using compilers, MPIs, and cudas)
EXCLUDE_COMBOS: Broken/excluded packages

```
BASE_COMPILER:
- ''
BASE_PACKAGES:
- autoconf
- bzip2
- curl
- readline
- xz
- zlib
- zstd
COMPILERS:
- gcc@7.3.0
- gcc@10.1.0
EXTERNAL_COMPILERS:
- gcc@4.8.5
UTILITY_COMPILER:
- gcc@7.3.0
UTILITIES:
- cmake
- git
- ninja
- python
MPIS:
- openmpi@4.0.5
EXTERNAL_MPIS:
- ''
CUDAS:
- ''
EXTERNAL_CUDAS:
- ''
TPLS:
- hdf5@1.10.6
- boost
- cgns@3.4.0
- metis@5.1.0
- parallel-netcdf@1.9.0
- parmetis@4.0.3
- zlib@1.2.11
- superlu-dist@5.4.0
- superlu@4.3
EXCLUDE_COMBOS:
- superlu@4.3%gcc@4.8.5
```

DEMO: Installation

```
(env_python) (base)  $ spack-cm install -h
usage: spack-cm install [-h] [-p PROJECT] [-m ALTHOSTNAME] [-r ROOT_PATH] [-s STAGE] [--spack SPACKVERSION] [--install-spack-deps] [-d] [-e]
                        [--no-project-modules] [--add-machine-to-install-path]

Run install routine for a project/machine combination.

optional arguments:
  -h, --help            show this help message and exit
  -p PROJECT, --project PROJECT
                        REQUIRED: Project for which to install TPLs (e.g., trilinos, pyomo, etc.).
  -m ALTHOSTNAME, --machine ALTHOSTNAME
                        OPTIONAL: Designate an alternate machine name (i.e., not the hostname of the machine).
  -r ROOT_PATH, --root ROOT_PATH
                        REQUIRED: Root path in which to install TPLs (e.g. ~/install/path, /shared/mount, etc.).
  -s STAGE, --stage STAGE
                        OPTIONAL: Select a single stage of the install to run. By default, all stages will run. Available choices: [base, compiler, utility,
                        tpl]
  --spack SPACKVERSION  OPTIONAL: Version of spack. Default: 0.16.1
  --install-spack-deps  OPTIONAL: Install spack system dependencies.
  -d, --debug           OPTIONAL: Enable "spack --debug" install mode.
  -e, --external        OPTIONAL: Allow spack to find and use system packages.
  --no-project-modules  OPTIONAL: Turn off use of project name in modulefile generation.
  --add-machine-to-install-path
                        OPTIONAL: Add the machine name to the install path.
```

DEMO: Configuration Files

DEMO: Module Files

```
$ module av

------------------------------------------ /where/ever/you/install/your/modulefiles/linux-rhel7-ppc64le/Core ------------
   gcc/7.3.0      gcc/10.1.0 (D)


$ module load gcc/7.3.0
$ module av

------------------------------------------ /where/ever/you/install/your/modulefiles/linux-rhel7-ppc64le/gcc/7.3.0 ------------
   boost/1.73.0    boost/1.74.0 (D)    metis/5.1.0    openmpi/4.0.4    openmpi/4.0.5 (D)    superlu/4.3    zlib/1.2.11

------------------------------------------ /where/ever/you/install/your/modulefiles/linux-rhel7-ppc64le/Core ------------
   gcc/7.3.0 (L)    gcc/10.1.0 (D)


$ module load openmpi/4.0.5
$ module av

---------------------------------- /where/ever/you/install/your/modulefiles/linux-rhel7-ppc64le/openmpi/4.0.5-ly7lkxy/gcc/7.3.0 -
   cgns/3.4.0    hdf5/1.10.6    parmetis/4.0.3    superlu-dist/5.4.0    valgrind/3.15.0

------------------------------------------ /where/ever/you/install/your/modulefiles/linux-rhel7-ppc64le/gcc/7.3.0 ------------
   boost/1.73.0    boost/1.74.0 (D)    metis/5.1.0    openmpi/4.0.4    openmpi/4.0.5 (L,D)    superlu/4.3    zlib/1.2.11

------------------------------------------ /where/ever/you/install/your/modulefiles/linux-rhel7-ppc64le/Core ------------
   gcc/7.3.0 (L)    gcc/10.1.0 (D)
```

Conclusion

- Future Steps:
  - Add additional functionality (to address more general use cases)
  - Plan to open-source (ETA: End of CY21)
- Long-term Goals:
  - Become obsolete