SPACK

Maximilian Bauregger

UHH

December 16, 2020

Outline

Introducion

Building software

From source code to binary executable.
A lot of complexity to build software.

- compilers
- build systems (cmake, ninja, mesos, etc.)
- arch (AMD64, AARCH64, PowerPC)
- etc.

Libraries

But wait! There is more complexity.

- system libs (glibc)
- BLAS, numpy
- Other hardware (mpi, cuda)
- etc.

Packaging

"Windows style"

- Don't manage deps centrally
- Every program has to install/update/pull deps by itself
- You don't know what you get

Software repositories

- Prevalent on unix systems
- For OS or programming language (apt, dnf, pip)
- Central tooling to manage packages
- Manages dependencies
- "Standard" packages

Docker

- "Bring your own OS"
- Isolation
- Bundle app and libs
- Depends on Linux namespaces
- Still OS packaging
- Focuses on portability

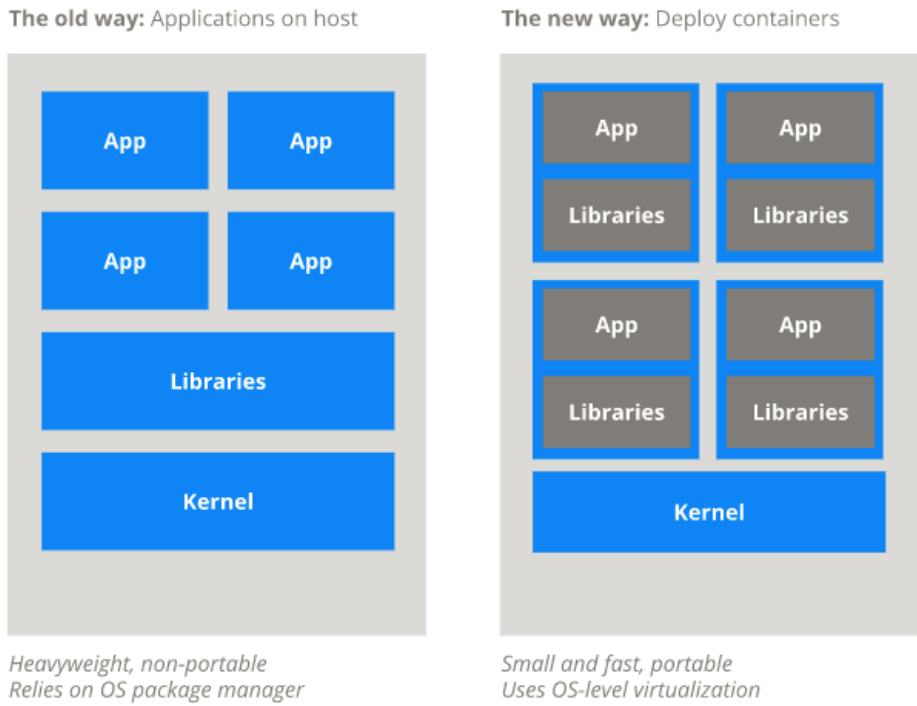

Figure 1: Bundling app via docker compared to installing it directly on the host. From Virdee 2018

HPC needs

Specific hardware

- A lot of machines
- Specific architectures
- MPI implementations
- GPGPU (cuda, etc.)
- File systems

Specific software

- Specific compilers (and versions thereof)
- Speed over portability
- Old versions (F77...)
- "Works on my machine"

All in all a lot of complexity!

How to manage HPC software?

- Thousands of ways to build software
- Custom repositories?
- Special tooling?

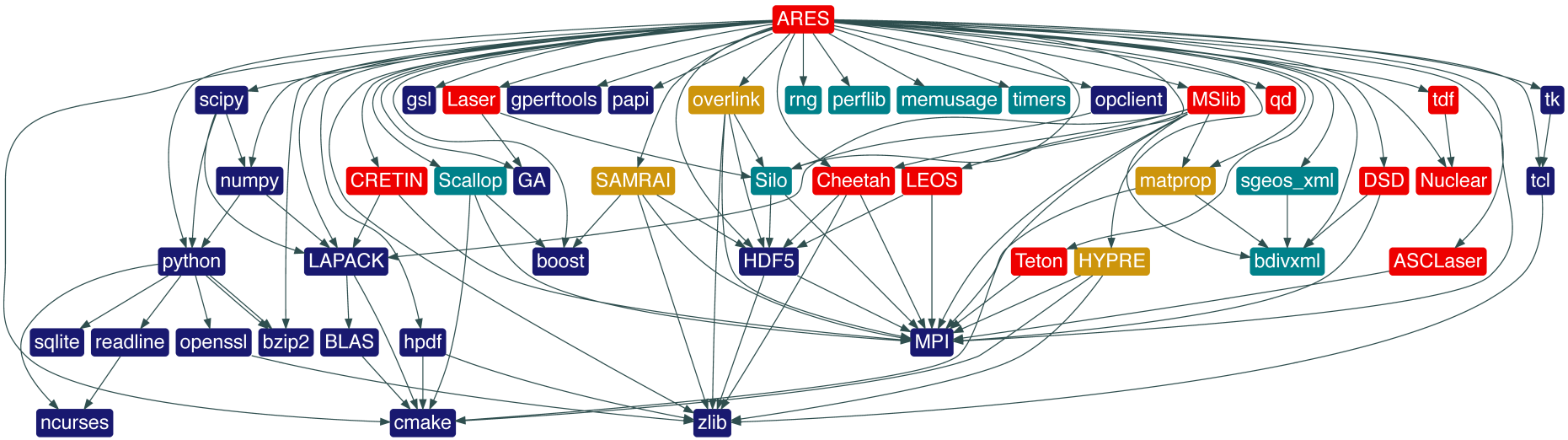

Figure 2: Dependency graph of one configuration of ARES, a Livermore-developed hydrodynamics code[..]. From *Spack: A Flexible Package Manager for HPC Software* 2020

Spack

Introduction to Spack

- Supercomputer PACKage manager
- Started 2013 by Todd Gamblin at Lawrence Livermore National Laboratory
- Written in Python
- Open Source since 2014
- Linux & macOS

Adoption

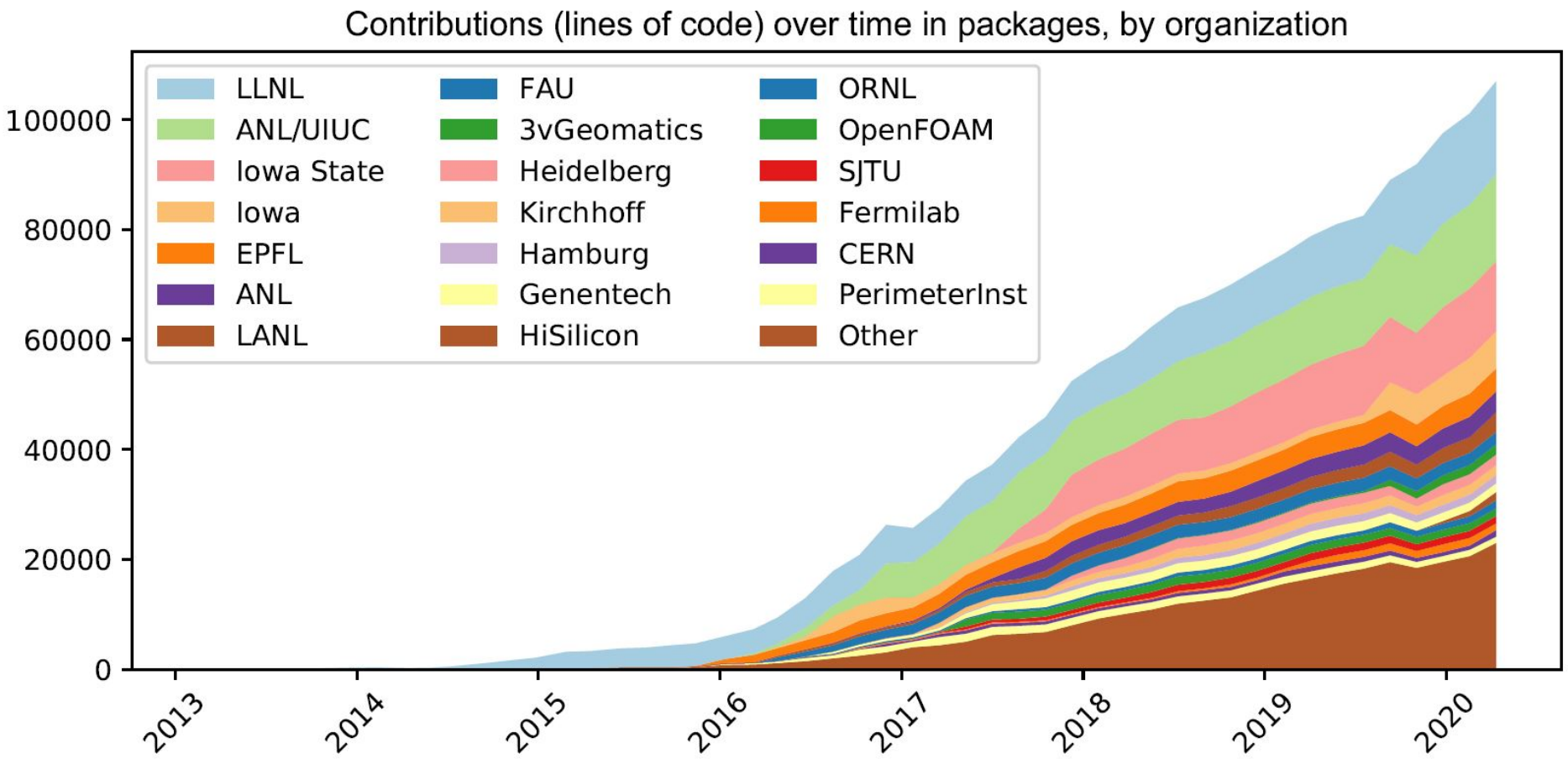

Figure 3: External contributions to Spack increased rapidly after its open-source release in 2014. From *Spack: A Flexible Package Manager for HPC Software* 2020

Spec syntax

- Own syntax to describe constrains
- Specify compiler, deps, flags.

```
spack install zlib
spack install zlib @1.2
spack install zlib @1.2 %gcc@4.7.2
spack install zlib @1.2 +pic -shared
spack install zlib ^libelf
spack install zlib ^libelf @0.8 %gcc
spack install zlib target=broadwell cflags=-O3
```

Figure 4: Different specs for installing zlib

DAGs

- Directed Acyclic Graph
- Dependencies
- No cycles

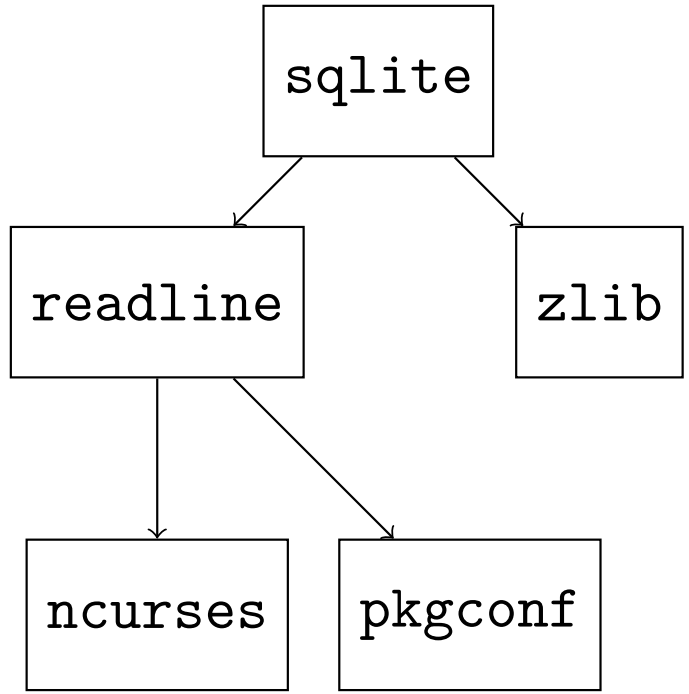

Figure 5: Dependencies of SQLite package

Concretization

- "Heart" of Spack's dependency management
- "Fill out the blanks"
- Virtual deps
- Hashing of whole configuration
- Coexistence of different version of a package
- NP-Complete (-> SAT)

```
sqlite@3.33.0%gcc@10.2.0+column_metadata+fts~functions~rtree arch=linux-archrolling-broadwell
    ^readline@8.0%gcc@10.2.0 arch=linux-archrolling-broadwell
        ^ncurses@6.2%gcc@10.2.0~symlinks+termlib arch=linux-archrolling-broadwell
            ^pkgconf@1.7.3%gcc@10.2.0 arch=linux-archrolling-broadwell
    ^zlib@1.2.11%gcc@10.2.0+optimize+pic+shared arch=linux-archrolling-broadwell
```

Figure 6: Output of `spack spec sqlite`

package.py

- Single python class
- `do_install()`
- `install()`
- Spack manages build systems
- Dependencies linked via `RPATH` (no `LD_LIBRARY_PATH`)

```
class Zlib(Package):
    """A free, general-purpose, legally unencumbered lossless
    data-compression library.
    """

    homepage = "http://zlib.net"
    # URL must remain http:// so Spack can bootstrap curl
    url = "http://zlib.net/fossils/zlib-1.2.11.tar.gz"

    version('1.2.11', sha256='c3e5e9fdd5004dcb542feda5ee4f0ff0744628baf8ed2dd5d66f8ca1197cb1a1')
    # Due to the bug fixes, any installations of 1.2.9 or 1.2.10 should be
    # immediately replaced with 1.2.11.
    version('1.2.8', sha256='36658cb768a54c1d4dec43c3116c27ed893e88b02ecfcb44f2166f9c0b7f2a0d')
    version('1.2.3', sha256='1795c7d067a43174113fdf03447532f373e1c6c57c08d61d9e4e9be5e244b05e')

    variant('pic', default=True,
            description='Produce position-independent code (for shared libs)')
    variant('shared', default=True,
            description='Enables the build of shared libraries.')
    variant('optimize', default=True,
            description='Enable -O2 for a more optimized lib')

    patch('w_patch.patch', when="@1.2.11%cce")

    @property
    def libs(self):
        shared = '+shared' in self.spec
        return find_libraries(
            ['libz'], root=self.prefix, recursive=True, shared=shared
        )

    def setup_build_environment(self, env):
        if '+pic' in self.spec:
            env.append_flags('CFLAGS', self.compiler.cc_pic_flag)
        if '+optimize' in self.spec:
            env.append_flags('CFLAGS', '-O2')

    def install(self, spec, prefix):
        config_args = []
        if '~shared' in spec:
            config_args.append('--static')
        configure('--prefix={0}'.format(prefix), *config_args)

        make()
        if self.run_tests:
            make('check')
        make('install')
```

Figure 7: zlib's package.py

Demo Time

Spack + Docker

- Docker is good at packaging
- Spack is good at building
- Build a docker image with spack
- `spack.yaml` to define specs
- `spack containerize > Dockerfile`

```
FROM spack/centos:7

WORKDIR /build
COPY spack.yaml .
RUN spack install
```

Figure 8: A simple Dockerfile that uses spack.yaml to install a large number of packages. From Gamblin 2019

Conclusion

Conclusion

- Bring order to Chaos
- Unify tooling
- Portable packages
- Explicit dependencies

Literature

Literature I

T. Gamblin et al. "The Spack package manager: bringing order to HPC software chaos." In: *SC '15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 2015, pp. 1–12. DOI: 10.1145/2807591.2807623. URL: https://ieeexplore.ieee.org/abstract/document/7832814 (visited on 12/15/2020).

M. Melara et al. "Using Spack to manage software on Cray supercomputers." In: *Proceedings of Cray User Group* (2017). URL: https://cug.org/proceedings/cug2017_proceedings/includes/files/pap153s2-file1.pdf (visited on 12/15/2020).

J. Virdee. *Why You Should be Using Containers*. SNP Technologies Inc. Jan. 2, 2018. URL: https://www.snp.com/blog/why-you-should-be-using-containers (visited on 12/15/2020).

Literature II


G. Becker. *Managing HPC Software Complexity with Spack*. Talk. Lawrence Livermore National Laboratory (LLNL), Aug. 20, 2019. URL: https://www.youtube.com/watch?v=z7ZdnCkaPCY.


T. Gamblin. *Spack: A Package Manager for HPC Systems*. 2019 R&D 100 Award Entry. Lawrence Livermore National Laboratory (LLNL), 2019.


A. Holly. "Software Installation Simplified." In: *Science & Technology Review* (July 2020). URL: https://str.llnl.gov/content/pages/2020-07/pdf/07.20.4.pdf (visited on 12/15/2020).


About Spack. URL: https://spack.io/about/ (visited on 12/15/2020).

Literature III

Spack Documentation. URL: https://spack.readthedocs.io/en/v0.16.0/ (visited on 12/15/2020).

Spack GitHub. URL: https://github.com/spack/spack (visited on 12/15/2020).

Spack: A Flexible Package Manager for HPC Software. Lawrence Livermore National Laboratory (LLNL). URL: https://computing.llnl.gov/projects/spack-hpc-package-manager (visited on 12/15/2020).