Shifter and Docker SLURM Integration

Jean-Baptiste Aubort, Gilles Fourestey, Vittoria Rezzonico, Ricardo Silva

EPFL - SCITAS

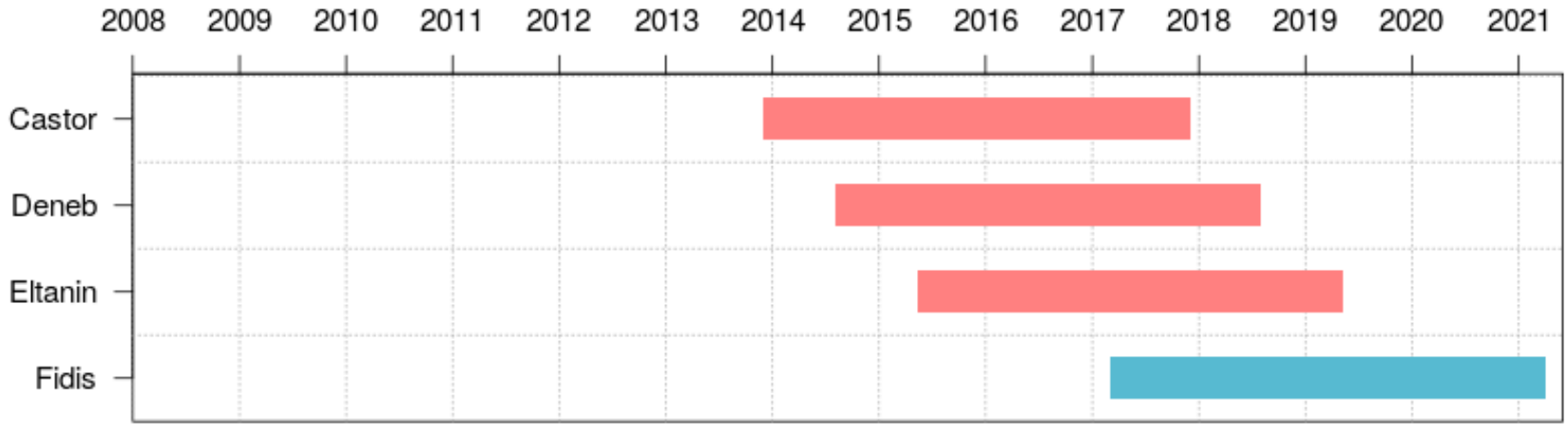
October 27, 2017

SCITAS

- manage HPC clusters at EPFL
- since 2013 is it's current form
  - bringing together systems and applications specialists
  - providing real *end-to-end* support
- besides HPC, provide other services to support research in general
  - training: not only on the use of the clusters but also other areas, for example data management
  - code hosting and CI: `https://c4science.ch`
- prepare for future trends: accelerators, ML

Hardware

- >28k cores spread across 4 clusters
- ~1 cluster per year, 4 year typical lifetime
- classical HPC configuration
  - InfiniBand network
  - dedicated /scratch filesystem (GPFS)
- 3 PB shared storage (GPFS)
- scheduler: Slurm



Reproducibility

Three main dimensions

Reproducibility

Three main dimensions

- Code

Reproducibility

Three main dimensions

- Code
  - version control + `https://c4science.ch`

Reproducibility

Three main dimensions

- Code
  - version control + `https://c4science.ch`
- Software

Reproducibility

Three main dimensions

- Code
  - version control + `https://c4science.ch`
- Software
  - `https://spack.readthedocs.io/` + automation
    but not everything is packaged and is not as decoupled as we (and our users) would like from the OS

Reproducibility

Three main dimensions

- Code
  - version control + `https://c4science.ch`
- Software
  - `https://spack.readthedocs.io/` + automation
    but not everything is packaged and is not as decoupled as we (and our users) would like from the OS
- Data

Reproducibility

Three main dimensions

- Code
  - version control + `https://c4science.ch`
- Software
  - `https://spack.readthedocs.io/` + automation
    but not everything is packaged and is not as decoupled as we (and our users) would like from the OS
- Data
  - *to be continued...*

Reproducibility vs Productivity

If reproducibility sits at one end of the *change* velocity spectrum.
Why doesn't my job run anymore? It worked perfectly back in 2004!

Reproducibility vs Productivity

If reproducibility sits at one end of the *change* velocity spectrum.
Why doesn't my job run anymore? It worked perfectly back in 2004!

We change too often!

Reproducibility vs Productivity

If reproducibility sits at one end of the *change* velocity spectrum. *Why doesn't my job run anymore? It worked perfectly back in 2004!*

We change too often!

The other recurring *complaint* is how *old* our software environment is. *Why isn't the latest version of X available/supported?*

Reproducibility vs Productivity

If reproducibility sits at one end of the *change* velocity spectrum. *Why doesn't my job run anymore? It worked perfectly back in 2004!*

We change too often!

The other recurring *complaint* is how *old* our software environment is. *Why isn't the latest version of X available/supported?*

We don't change often enough!

An approach:

Containers!

Slurm integration: options

A few options have popped up in the last couple of years which integrate *containers* with a classical HPC schedulers:

- NERSC: Shifter
- LBL: Singularity
- LANL: CharlieCloud
- Univa: Grid Engine CE

Slurm integration: options

A few options have popped up in the last couple of years which integrate *containers* with a classical HPC schedulers:

- NERSC: Shifter
- LBL: Singularity
- LANL: CharlieCloud
- ~~Univa: Grid Engine CE~~ *uses Docker and UGE*

Slurm integration: options

A few options have popped up in the last couple of years which integrate *containers* with a classical HPC schedulers:

- NERSC: Shifter
- LBL: Singularity
- ~~LANL: CharlieCloud~~ *only just popped on our radar*
- ~~Univa: Grid Engine CE~~ *uses Docker and UGE*

Slurm integration: options

A few options have popped up in the last couple of years which integrate *containers* with a classical HPC schedulers:

- NERSC: Shifter
- ~~LBL: Singularity~~ *uses tar balls*
- ~~LANL: CharlieCloud~~ *only just popped on our radar*
- ~~Univa: Grid Engine CE~~ *uses Docker and UGE*

Shifter: Deployment

There are three parts to deploying shifter: the Image Gateway, the Shifter Runtime and the integration with Slurm:

Shifter: Deployment

There are three parts to deploying shifter: the Image Gateway, the Shifter Runtime and the integration with Slurm:

Image Gateway

- built the RPM from the repository
- on a node with access to the shared storage, but not necessarily part of the Slurm cluster
- the only machine that needs the `Docker` tools installed
- we did it with Puppet, along with it's dependencies: `Redis` and `MongoDB`
- most of the work was to figure out a consistent set of versions of Python modules

Shifter: Deployment

There are three parts to deploying shifter: the Image Gateway, the Shifter Runtime and the integration with Slurm:

Shifter Runtime

- built the RPM from the repository
- installed on the login and worker nodes

Shifter: Deployment

There are three parts to deploying shifter: the Image Gateway, the Shifter Runtime and the integration with Slurm:

Shifter Runtime

- built the RPM from the repository
- installed on the login and worker nodes

Slurm integration

- one configuration line in /etc/slurm/plugstack.conf:
  `optional /usr/lib64/shifter/shifter_slurm.so shifter_config=/etc/shifter/udiRoot.conf`

Shifter: Configuration

One main configuration file:

`/etc/shifter/udiRoot.conf`

- paths to tools and working directories
- URL of the Image Gateway
- Local filesystems that will be mounted

How to use it: running jobs

Compiled executables

Compile your application with the same docker image you are planning to run it.

Prerequisites

Docker images mount */home* directories when launched via SLURM.

Example: running *Hello world*

- We will use the `debian:latest` image on Dockerhub:

```
shifterimg pull debian:latest
```

SLURM file: hello.slurm

```
1  #!/bin/bash
2
3  #SBATCH --nodes=1
4  #SBATCH --ntasks-per-node=1
5  #SBATCH --cpus-per-task=1
6  #SBATCH --image=debian:latest
7
8  srun shifter echo "Hello, world!"
```

- submit the job with

```
sbatch hello.slurm
```

Example: MPI jobs

- We will use the `rezzonic/spack-mpich` image on Dockerhub:

  `shifterimg pull rezzonic/spack-mpich`
- It contains mpich installed via *spack*.
- We will need a helper file to compile our application:

Helper file: `prepare-mpich.sh`

```
1  #!/bin/bash
2
3  source /spack/share/spack/setup-env.sh
4  module avail
5  module load mpich
6
7  mpicc hello.c -o hello.exe
```

Example: MPI jobs

We can now create the compilation job

SLURM file: prepare-mpich.slurm

```
1 #!/bin/bash
2
3 #SBATCH --nodes=1
4 #SBATCH --ntasks-per-node=1
5 #SBATCH --cpus-per-task=1
6 #SBATCH --image=rezzonic/spack-mpich
7
8 srun shifter ./prepare-mpich.sh
```

and submit the compilation with

```
sbatch prepare-mpich.slurm
```

Example: MPI jobs

We are ready to run our MPI application:

SLURM file: run-mpich.slurm

```
1 #!/bin/bash
2
3 #SBATCH --nodes=2
4 #SBATCH --ntasks-per-node=1
5 #SBATCH --cpus-per-task=1
6 #SBATCH --image=rezzonic/spack-mpich
7
8 srun --mpi=pmi2 shifter ./hello.exe
```

with

```
sbatch run-mpich.slurm
```

Shifter: what next?

- Put it in production in our more recent cluster
  - test the more recent release of Shifter
  - solve a couple of small operational issues (ex: `/etc/passwd` + LDAP)
- profit from the recent contributions to shifter regarding GPUs and MPI
- create a few reference containers we support, with software built as we build it in our clusters
- test popular upstream containers
- private registries for our users

Thank you! Questions?