• Scientific IT and Application support unit at EPFL

• created in February 2014

• today:
  5 system engineers + 6 application experts
SCITAS activities

responsible for
the central EPFL computing platforms

• 2’453 compute nodes
• 45’316 compute cores
• 1.1 PFLOPs computing power
• 3 PB of storage
SCITAS activities

Users’ training

• Base (intro to Linux, clusters usage)
• Advanced HPC techniques (MPI, OpenMP, CUDA)
• Best practices (version management with Git)
SCITAS activities

Application support

- code parallelization, porting, optimization
- dedicated application development

50% of an FTE awarded in a competitive way for short term projects
software

- massively parallel jobs/classical HPC: these users usually also go to CSCS
- scientific computing
- data analysis/big data
• even if Big Data and HPC are different in nature, they can run on the same infrastructure

• new features at SCITAS:
  - spark (production)
  - docker containers (test)
  - jupyterhub (test)
most recurrent reproducibility issue we encounter:

*my code used to work, it doesn’t work anymore*

At this point, we need to check for changes in:

- environment
- code
- input
reproducibility

ideal situation

inputs

code

environment

one-click reproducibility
reproducibility

SCITAS environment is:

- available
- reproducible

... both at build-time and run-time
reproducibility

We use the spack package manager and our software builds

• are fully automated
• require limited human intervention
With spack you can install HPC packages

- on your laptop
- on a docker image
...you can get the specs from the SCITAS clusters to build exactly in the same way as on we did on our machines

♥ module list

Currently Loaded Modules:
1) gcc/5.3.0  2) openmpi/1.10.2

♥ cat $OPENMPI_ROOT/.spack/spec.yaml
reproducibility

code

• must be versioned
• and tested periodically or at every revision
• you can host your code in Switzerland with the c4science.ch platform
reproducibility

versioning
review
statistics

code

continuous integration

paper
reproducible results

authentication

SWITCHaai
local
Google
GitHub

reproducibility

researchers

programmers

authentication

authentication

authentication

collaborator
reproducibility

inputs must be

• stored in a safe place
• accessible
• fully documented
storage, present situation

our current configuration with

- *scratch* space
- *home* directories plus *work*, with backup
storage, present situation

we see an increasing need for

• long-term archive
• a way to transfer acquisition data to our clusters
storage, the ideal future

- The live data can be also shared with the outside world.
- Data is automatically replicated among the different zones, based on a set of user-defined rules.
data workflows

acquisition, analysis, archive

- data from the acquisition site and automatically replicated to the live data area

- it can then be directly copied to the long-term archive, or next to the clusters (GPFS home or work) for analysis

- the analysis results are in scratch. When the researcher is happy with the results, they can move the new data to the live data zone for sharing and archiving.
the archived data contains the environment, a link to the correct version of the code, the input files as well as the final results

in order to check the integrity of the archived data, we will

- transfer the data from archive to home via the live data zone
- re-run the analysis
- compare the results with the ones in the archived data
Different types of clusters can be plugged in to the architecture. For instance:

- a big data cluster running spark/hadoop type workloads
- with datasets stored on Ceph
links

• c4science.ch
• scitas.epfl.ch