computecanada

Providing A Unified User Environment for Canada's National Advanced Computing Centers

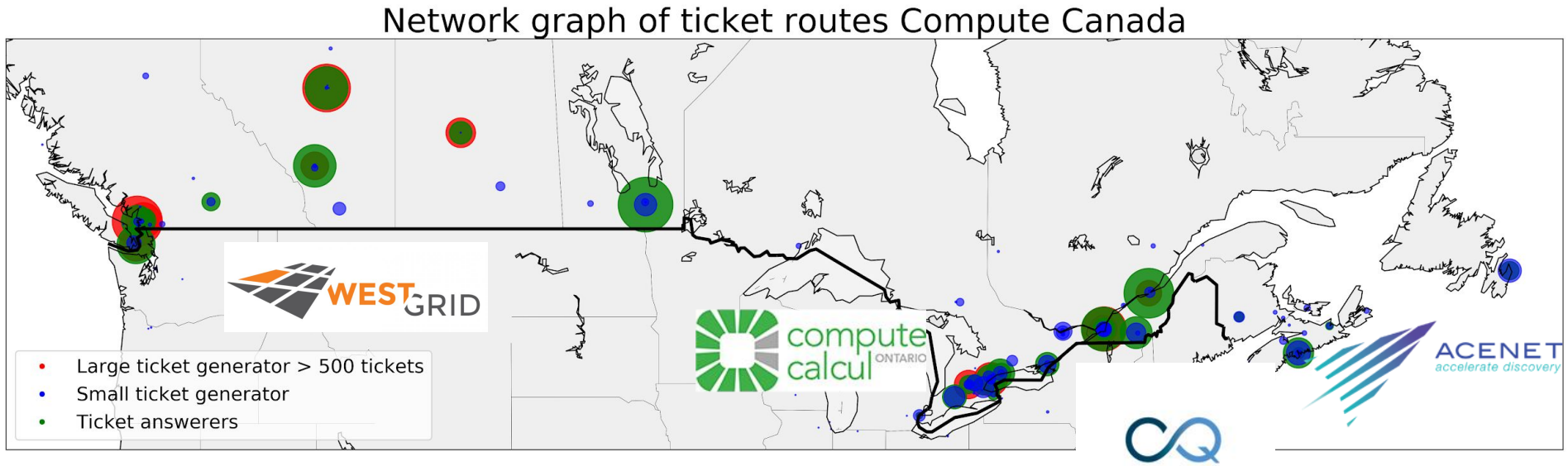
Maxime Boissonneault, Bart Oldeman, Ryan Taylor
on behalf of
Compute Canada Research Support National Team

Presentation outline

- Compute Canada, where we were, where we are
- Software installation : Goal and design overview
- Tools used :
  - CVMFS
  - Nix, EasyBuild
  - Lmod
  - Python and Anaconda
- Monitoring, demo, documentation
- Summary

What is Compute Canada ?

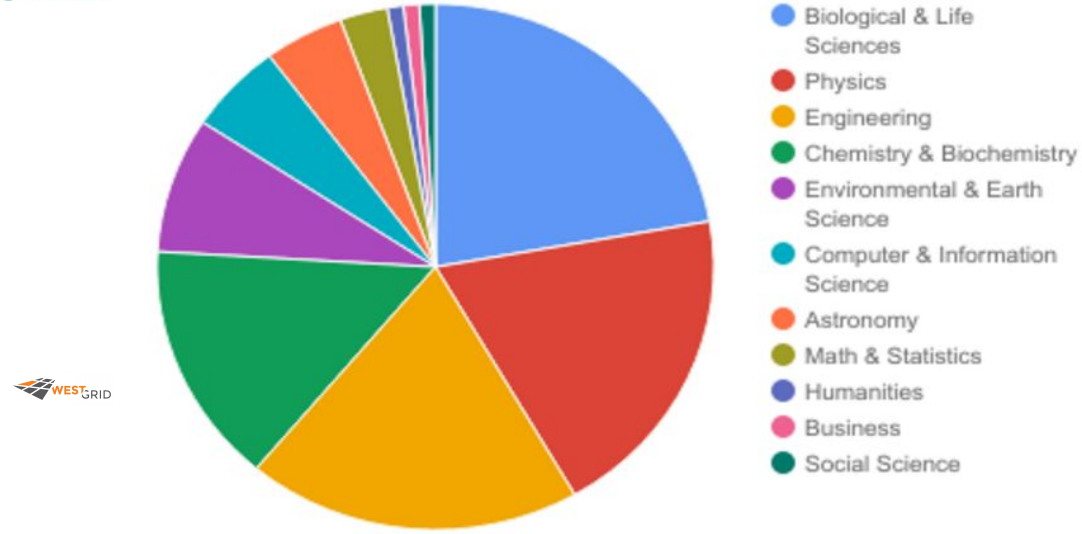
Compute Canada : the people



All research disciplines supported

Free access for any researcher at a Canadian institution

- 4 regional consortia
- 35 member institutions
- ~200 technical staff
- ~15,000 user accounts
  - 20% growth per year

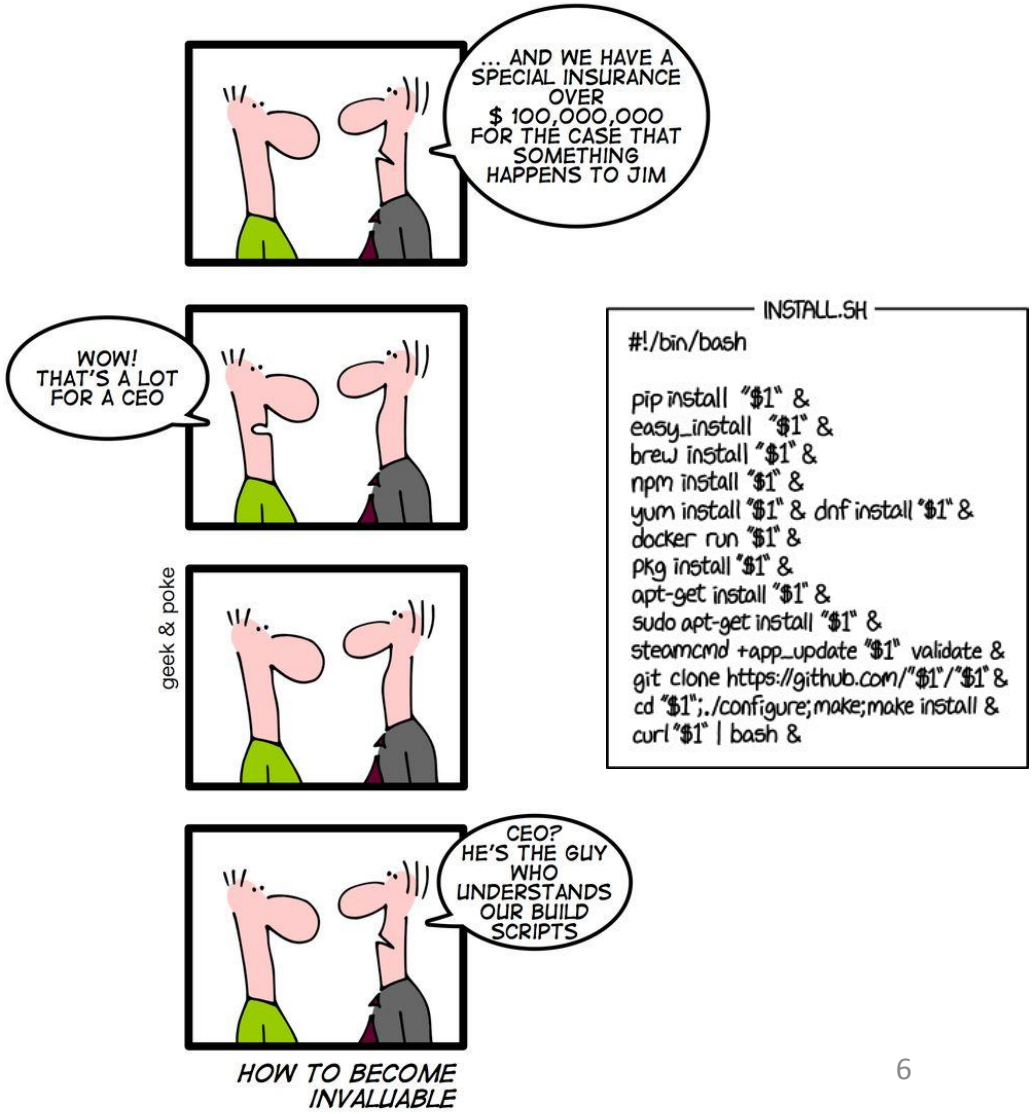


Before 2015

- Around 30 Compute Canada sites hosting hardware
- Over 50 clusters or other hardware resources
- All configured differently

Software Installation and Distribution: Then

- Typically was:
  - on local cluster filesystem
  - maybe documented
  - probably not automated
  - probably not standardized
- Repeat for each compiler, software version, instruction set extension
- That's just on one cluster



In 2015

- New funding to deploy larger infrastructure
- Consolidation of sites hosting equipment
- Shift in focus toward unification of user experience

Compute Canada : the hardware

5 major national systems
~15 legacy systems
200K cores, 22 PF
70 PB disk, 180 PB tape

System	Type	Network	Production
Arbutus	Cloud	10 GbE	2016 H2
Cedar	General	OPA	2017 H1
Graham	General	EDR IB	2017 H1
Niagara	Large MPI	EDR IB	2018 H1
Béluga	General	EDR IB	2019 H1

compute canada | calcul canada

National teams

National teams

- Research Support National Team (RSNT)
- CVMFS National Team
- Monitoring National Team
- Network National Team
- Scheduler National Team
- Storage National Team
- Cloud National Team

What is the Research Support National Team

- Mandates
  - Helpdesk
  - Documentation
  - Software installation
- Scope
  - Post 2015 systems
  - Existing and future national services (cloud, Globus, etc.)

Who is the RSNT ?

- Lead
  - Maxime Boissonneault (Calcul Québec)
- Helpdesk
  - Sergiy Khan (ACENET), Abdellatif Daghrach (HPC4Health)
- Documentation
  - Ross Dickson (ACENET), Daniel Stubbs (Calcul Québec),
    Nikolai Sergueev (Calcul Québec)
- Software installation
  - Bart Oldeman (Calcul Québec), Charles Coulombe (Calcul Québec), Doug Roberts (SHARCNET) + 45 other staff with publishing permissions
- Representatives
  - Belaid Moa (WestGrid), Ata Roudgar (WestGrid),
    Pawel Pomorski (SHARCNET), Paul Preney (SHARCNET),
    Sergey Mashchenko (SHARCNET), Ramses van Zon (SciNet), Hartmut Schmider (CAC)

compute canada | calcul canada

Software environment

Goal

Users should be presented with an interface that is as **consistent** and **easy to use** as possible across **all sites**. It should also offer **optimal performance**.

1. All software should be accessible on every site, reliably and performantly.
2. Software should be independent from the underlying OS stack.
3. Software installation should be tracked and reproducible via automation.
4. The user interface should make it easy to use a large and evolving software stack.

What this means

All new Compute Canada sites

1. Need a distribution mechanism
    a. CVMFS : CERN Virtual Machine File System

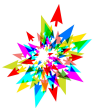
Consistency

2. Independent of the OS (Ubuntu, CentOS, Fedora, etc.)
    a. Nix
3. Automated installation (humans are not so consistent)
    a. EasyBuild

Easy to use

4. Needs a module interface that scale well
    a. Lmod with a hierarchical structure

compute canada | calcul canada



CERN Virtual Machine Filesystem (CVMFS)

The distribution mechanism

CVMFS National Team

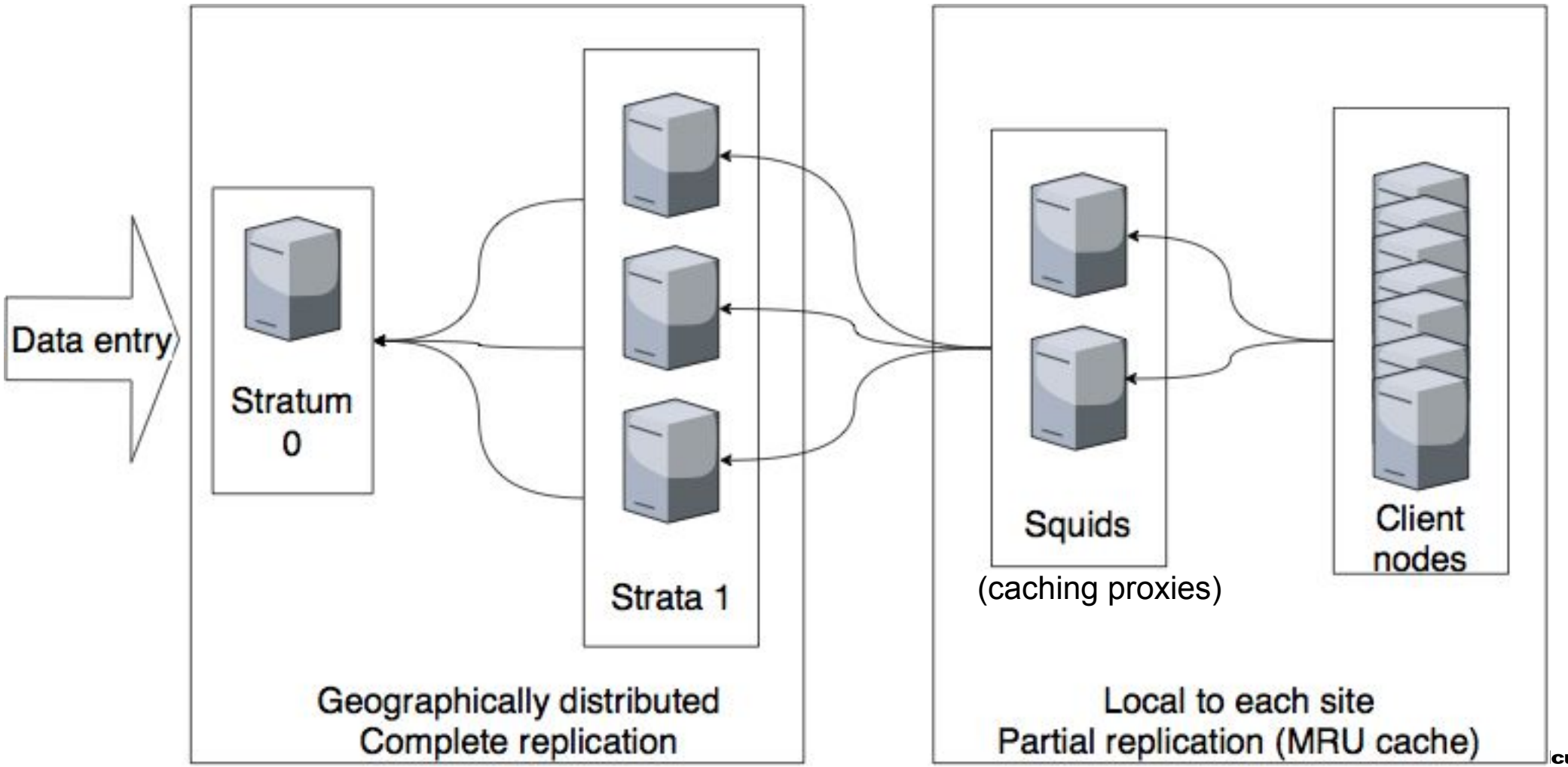
- Members
  - Ryan Taylor (lead)
  - Michel Barrette
  - Sergey Chelsky
  - Kuang Chung Chen
  - Leslie Groer
  - Mark Hahn
  - Jean-Francois Landry
  - Simon Nderitu
- Deployment & operation of CVMFS infrastructure
- Support & coordination for CVMFS client access at sites
- Scope: distribution of widely-used content (not just software)
  - data sets, container images, ...

compute canada | calcul canada

CVMFS

- Distributed file system optimized for scalable software delivery
  - originally used for High Energy Physics software from CERN
- Clients mount read-only POSIX file system via FUSE with HTTP transport
- Transparent deduplication, chunking and compression
- Clients pull files on demand, with aggressive caching
  - Minimized use of disk storage and network bandwidth
- Highly reliable: redundant servers, transparent failover
- Atomic updates. Publish once, automatically available everywhere
- https://cernvm.cern.ch/portal/filesystem
- https://cvmfs.readthedocs.io/en/stable/

CVMFS content delivery



Design overview

Software: design overview

Easybuild layer: modules for Intel, PGI, OpenMPI, CUDA, MKL, high-level applications.
Multiple architectures (sse3, avx, avx2, avx512)
`/cvmfs/soft.computecanada.ca/easybuild/{modules,software}/2017`

Easybuild-generated modules around Nix profiles (mostly deprecated):
GCC, Eclipse, Qt+Perl+Python no longer
`/cvmfs/soft.computecanada.ca/nix/var/nix/profiles/[a-z]*`

Nix layer: GNU libc, autotools, make, bash, cat, ls, awk, grep, etc.
`module nixpkgs/16.09 => $EBROOTNIXPKGS=`
`/cvmfs/soft.computecanada.ca/nix/var/nix/profiles/16.09`

Gray area: Slurm, Lustre client libraries, IB/OmniPath/InfiniPath client libraries (all dependencies of OpenMPI). In Nix layer, but can be overridden using PATH & LD_LIBRARY_PATH.

OS kernel, daemons, drivers, libcuda, anything privileged (e.g. the sudo command): always local.
Some legally restricted software too (VASP)

Nix and EasyBuild, conceptually

- Builds are performed through “recipes”
- Recipes are stored on Git. Compute Canada has its own fork of the repos :
  - Nixpkgs
  - Easybuild:
    - framework (high level Python scripts)
    - easyblocks
      - is it configure; make; make install, cmake, custom? (Python scripts)
    - easyconfigs
      - what are the configure parameters? (configuration files)

Nix:
The Compatibility Layer

Tools used : Nix

- Package, dependency & environment management system
- Deterministic builds using functional package language
- Used to provide dependencies for scientific applications
  - e.g. glibc, libxml2, etc.
- Abstraction layer between the OS and the scientific software stack, using nixpkgs/16.09 module
- Carries all* the dependencies of scientific software stack

* Exceptions: drivers, kernel modules, etc.

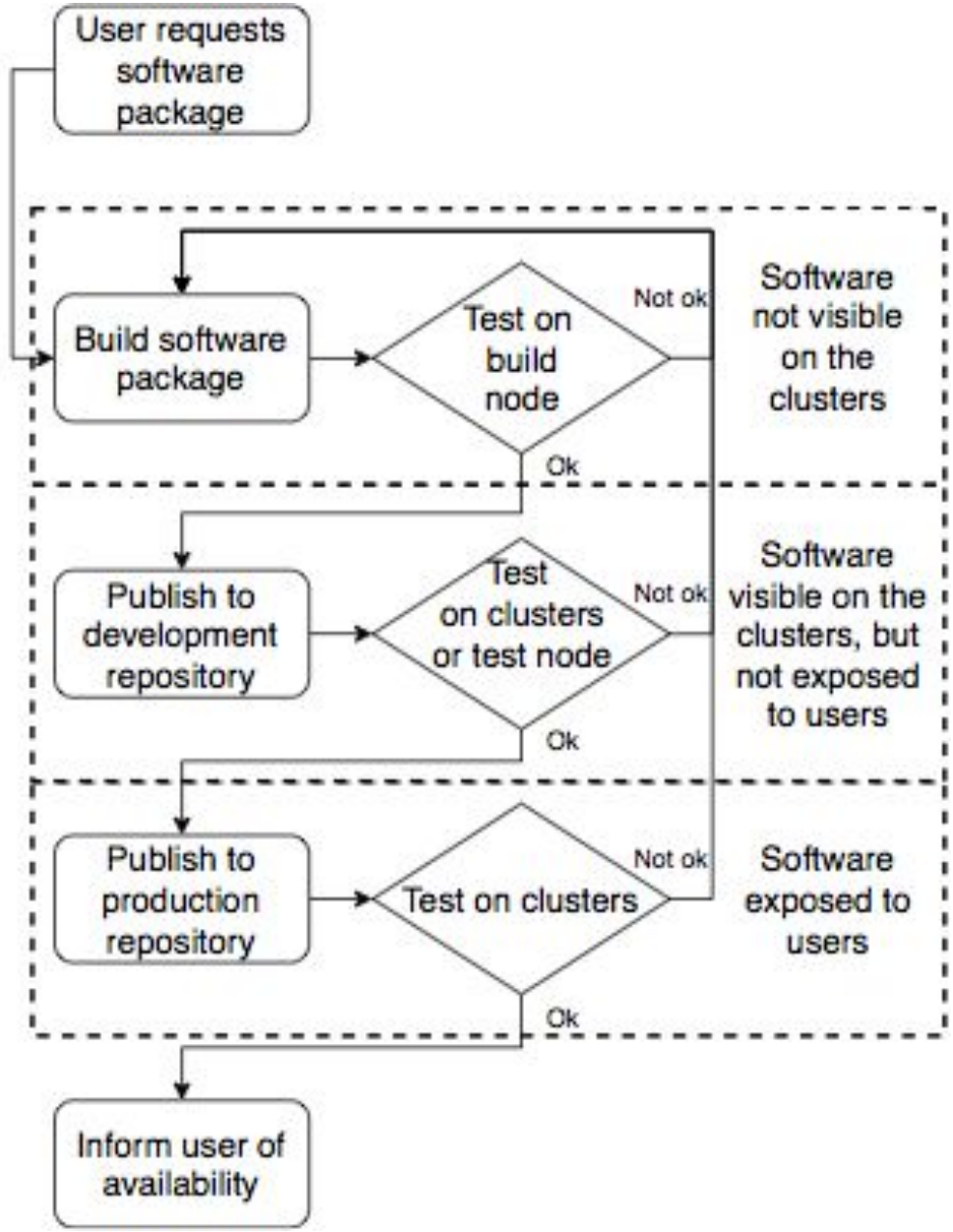
EasyBuild : The advanced research computing layer

Tools used : EasyBuild

- Build and installation framework for scientific applications
- Automation, configuration, logging, testing of builds
- Automatically commit recipes to git, update documentation
- Automatically generates module files
- Can batch build dependencies in parallel
- Thousands of community recipes
- Used for OpenMPI, GROMACS, NAMD, CUDA, etc.

Typical process

1. User requests a software to be installed
2. Staff decides whether it should be installed globally or in the user's account
   a. Globally (default route unless there is a reason not to)
   b. User's account
      i. Custom actions



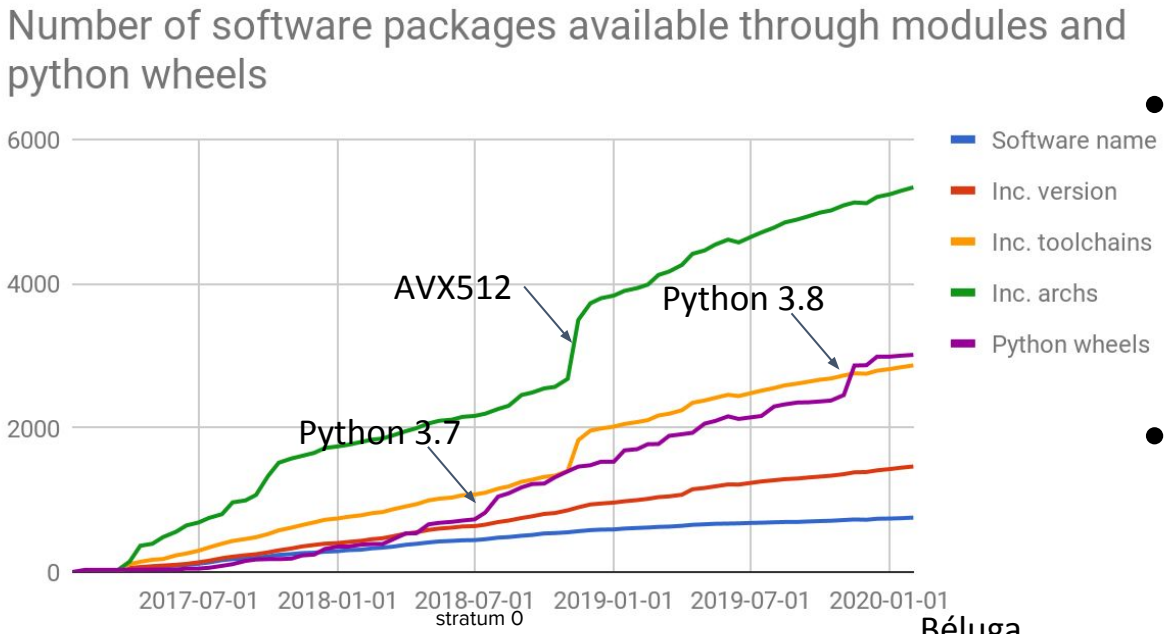
Compute Canada Software Stack

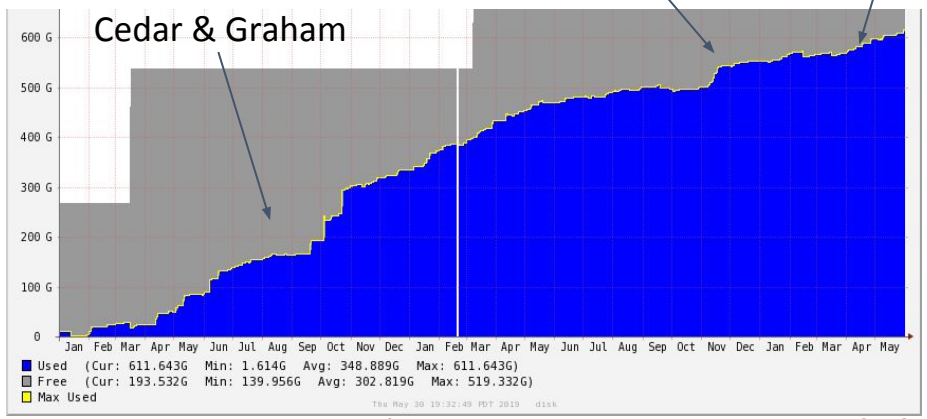
Available software

600+ scientific applications

4,000+ permutations of version/arch/toolchain

Type	Modules
AI	5
Bioinformatics	239
Chemistry	63
Data	19
Geo/Earth	23
Mathematics	82
MPI libraries	7
Physics	48
Various tools	176
Visualisation	28
Misc	38





- Two major new clusters with Skylake CPUs
- Built new modules with AVX512 for most packages
- High deduplication
- Further details

Supported licensed packages

MATLAB	VASP	GAUSSIAN
ADF	LS-DYNA	COMSOL
Materials Studio	FDTD Lumerical	ANSYS Suite
Star-CCM+	DL Poly4	CPMD
ORCA	Abaqus	Allinea
CellRanger	deMon2k	Cfour

User interface

How do you present 4000+ options to users without overwhelming them ?

LMOD

- User interface to list/load/unload and search environment modules
- Easy way to dynamically change software environment
- Hierarchical module tree, packages only visible once their dependencies are loaded
  - core
  - architecture (sse3, avx, avx2, avx512)
    - core - compiler
      - core - compiler - MPI
      - core - compiler - CUDA
        - core - compiler - CUDA - MPI

Python vs Anaconda

Python vs Anaconda

- Python is bad at packaging
  - Anaconda fixes that

Python vs Anaconda

- Python is bad at packaging
  - Anaconda fixes that

- Really ??

Python vs Anaconda

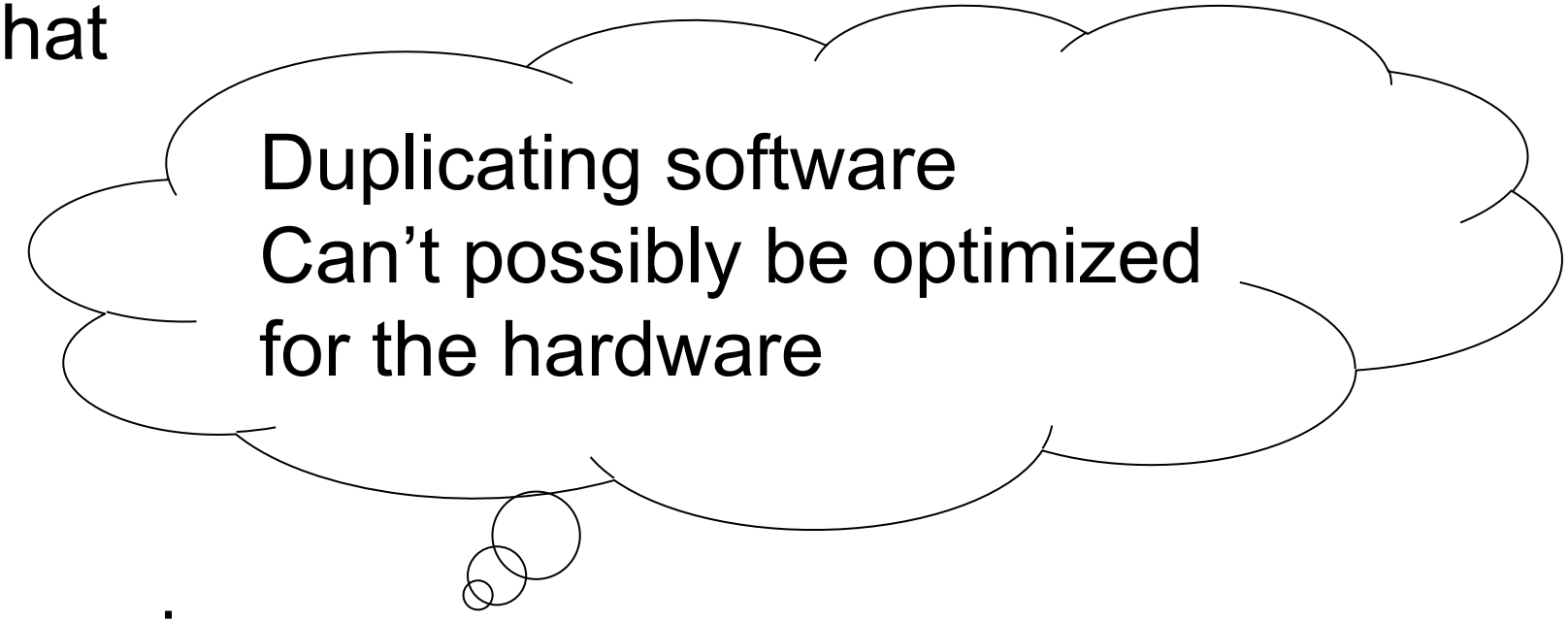
- Python is bad at packaging
  - Anaconda fixes that

- Really ??

  - conda install gcc
  - conda install openmpi
  - conda install cudatoolkit

Python vs Anaconda

- Python is bad at packaging
  - Anaconda fixes that



- Really ??
  - conda install gcc
  - conda install openmpi
  - conda install cudatoolkit

Python wheels

What are wheels?

Wheels are the new standard of Python distribution and are intended to replace eggs. Support is offered in pip >= 1.4 and setuptools >= 0.8.

Advantages of wheels

1. Faster installation for pure Python and native C extension packages.
2. Avoids arbitrary code execution for installation. (Avoids setup.py)
3. Installation of a C extension does not require a compiler on Linux, Windows or macOS.
4. Allows better caching for testing and continuous integration.
5. Creates .pyc files as part of installation to ensure they match the Python interpreter used.
6. More consistent installs across platforms and machines.
7. **You can compile your own wheels, linking against your compiled libraries**

Our supported wheels

```
[mboisson@build-node ~]$ ls
/cvmfs/soft.computecanada.ca/custom/python/wheelhouse/*/* | wc -w
3013
[mboisson@build-node ~]$ avail_wheels tensorflow_cpu
name            version    build           python    arch
--------------  ---------  -------------   --------  ------
tensorflow_cpu  2.0.0      computecanada   cp37      avx2
[mboisson@build-node ~]$ avail_wheels tensorflow_gpu
name            version    build           python    arch
--------------  ---------  -------------   --------  ------
tensorflow_gpu  2.0.0      computecanada   cp37      avx2
```

- https://docs.computecanada.ca/wiki/Available_wheels

Python vs Anaconda

“remember that anacondas are heavier and bulkier while pythons are longer and more agile”

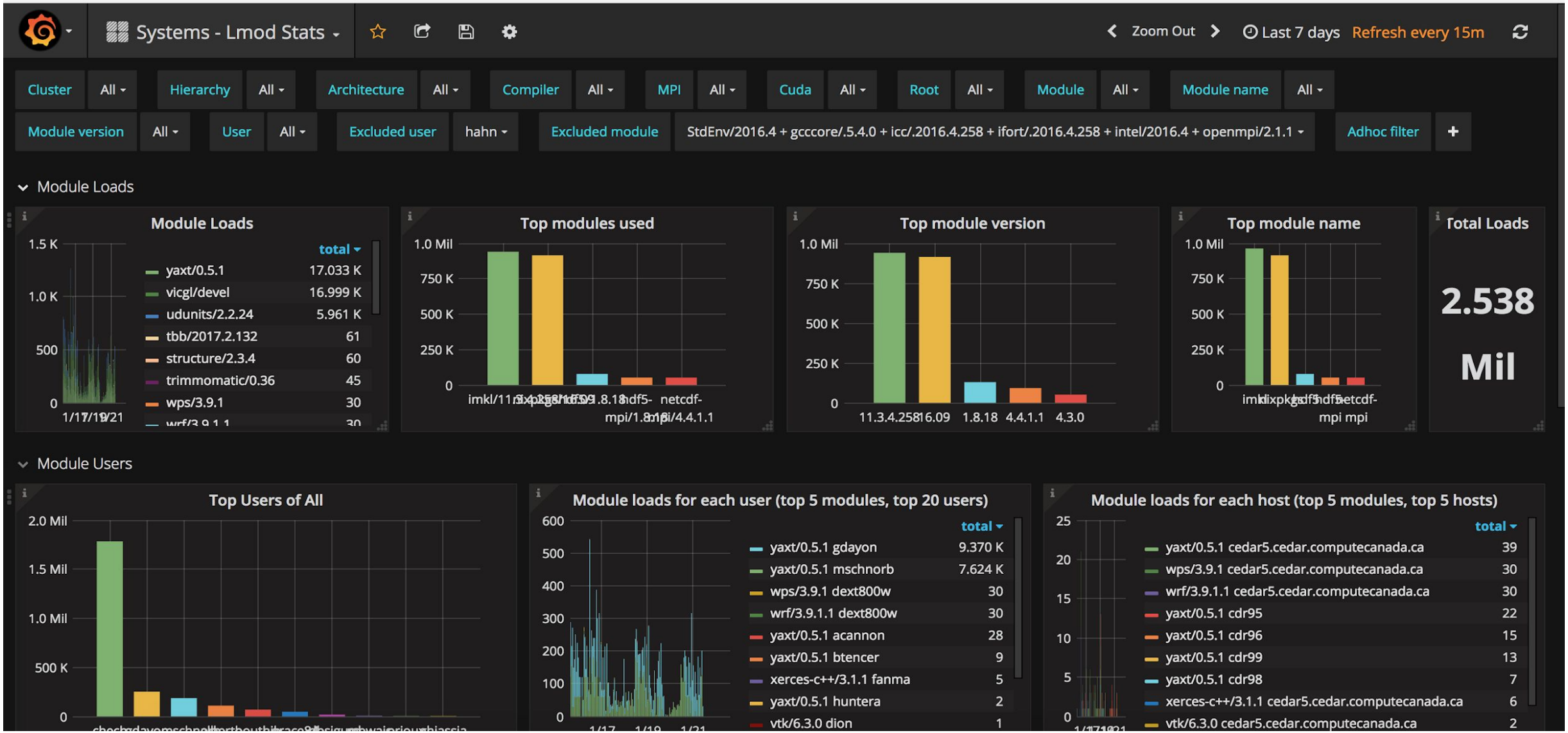
http://www.differencebetween.net/science/nature/difference-between-python-and-anaconda

Module usage tracking

What modules are our users using ?

- Every “module load” command is sent to syslogs
- Syslogs for all Compute Canada clusters are aggregated into an Elastic Search engine
- Grafana is used to produce dashboards of module usage

Module usage dashboard



Documentation

Documentation

- List of modules
  - https://docs.computecanada.ca/wiki/Available_software
- List of Python wheels
  - https://docs.computecanada.ca/wiki/Available_wheels

Demo

Cluster stack on Windows ?!

Tweet épinglé

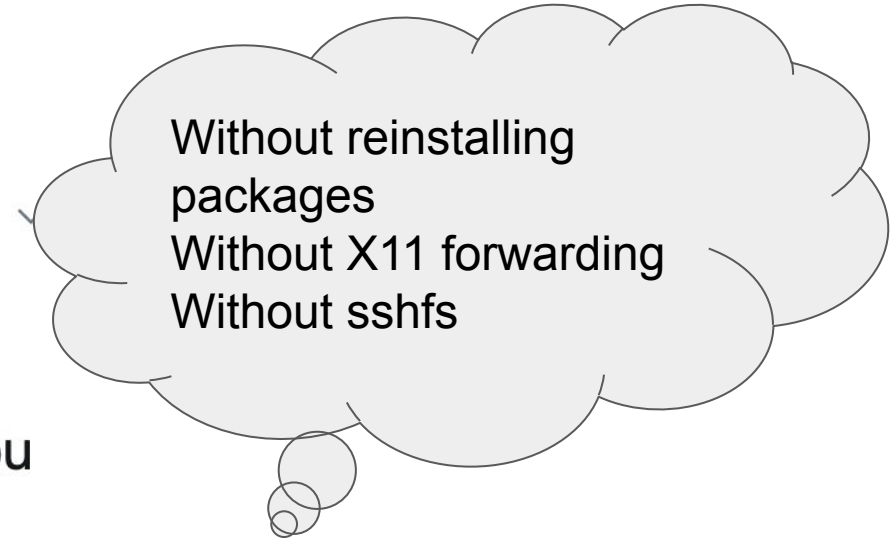
Maxime Boissonneault
@mboisso

If I told you that I want to use my HPC cluster's software environment on my Windows laptop, how crazy would you say I am ? Discover the answer during my talk at @PEARC_19

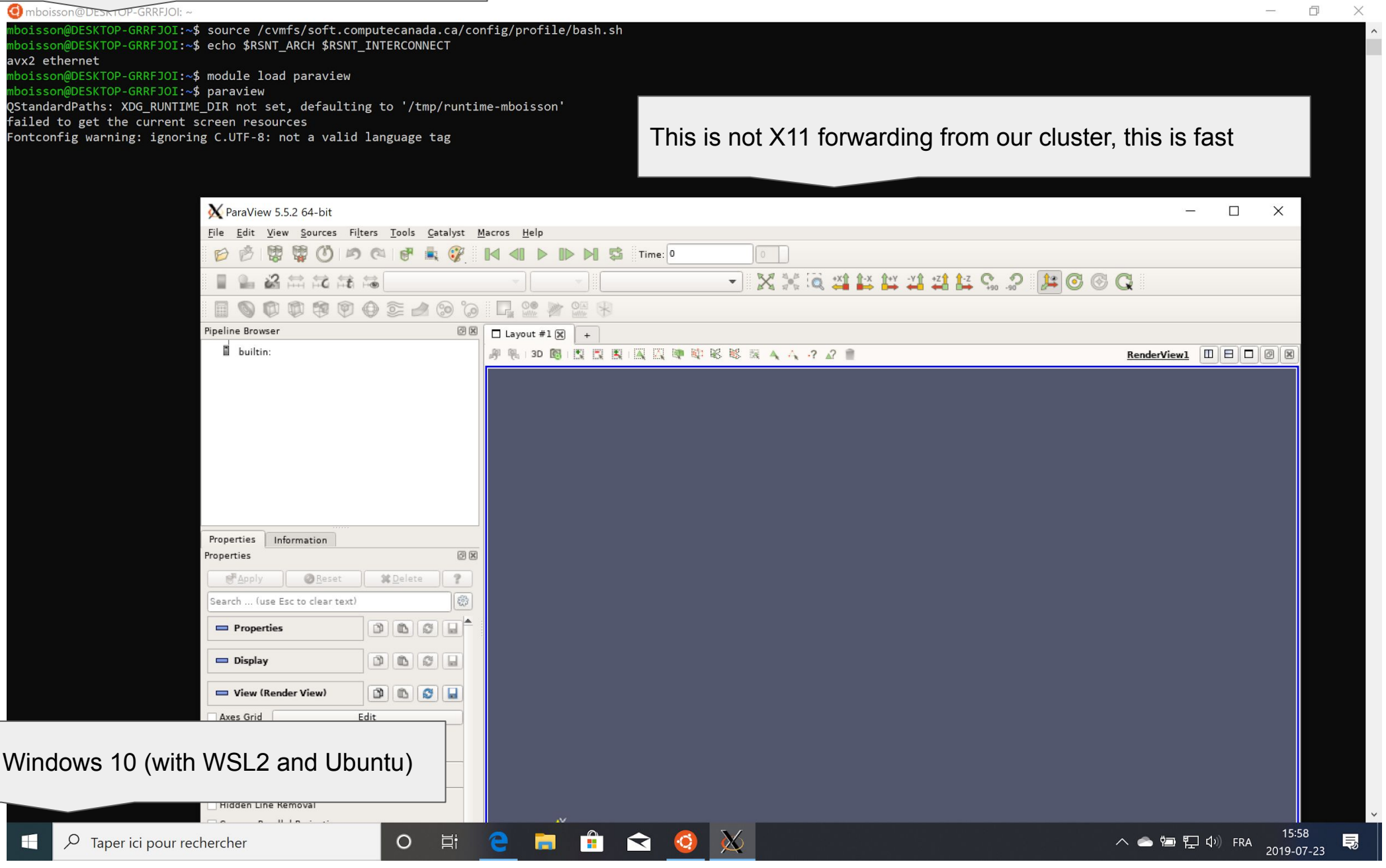
Traduire le Tweet

Are you out of your mind?	**60%**
That's a cake walk	40%

25 votes · Résultats finaux



This is not a remote server, this is my laptop



This is not X11 forwarding from our cluster, this is fast

This is Windows 10 (with WSL2 and Ubuntu)

You can use this too

- Mounting our software stack *
  - https://docs.computecanada.ca/wiki/Accessing_CVMFS

So what now ? Is it actually used ?

- In operation on all national systems since 2017
- Now being adopted
  - legacy systems
  - new local systems from our member institutions (not CC systems)
  - some advanced end users (for CICD of their code)
- Magic Castle (CC cluster in the cloud)
  - FOSDEM session : https://fosdem.org/2020/schedule/event/magic_castle/

Future work

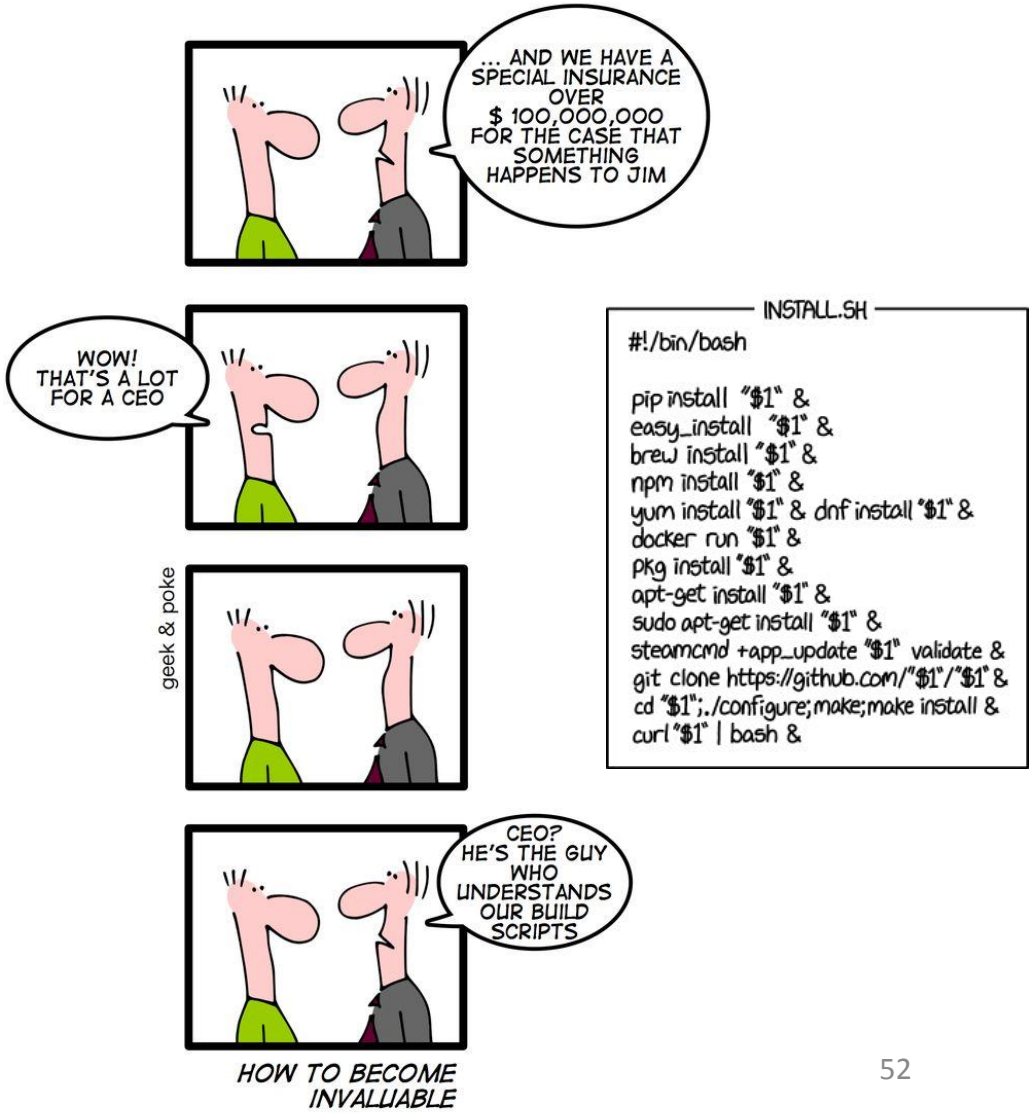
- 2020 software stack (WIP)
  - Will rely on Gentoo Prefix instead of Nix for the compatibility layer
  - Will require kernel >3.x (instead of >2.6.32)
  - May compile fat binaries
- Distribution of datasets
- Distribution of container images
- Better customization for non-Compute Canada systems

Future work

- Integration of ReFrame (WIP)
  - https://reframe-hpc.readthedocs.io/
- Integration of XALT
  - https://xalt.readthedocs.io/en/latest/
- Integration of Mii
  - https://github.com/codeandkey/mii

Software Installation and Distribution: Then

- Typically was:
  - on local cluster filesystem
  - maybe documented
  - probably not automated
  - probably not standardized
- Repeat for each compiler, software version, instruction set extension
- That's just on one cluster



Software Installation and Distribution: Now

- One unified stack distributed to all national systems
- Simple and consistent user interface
- All builds are fully reproducible, recorded, automated using recipes in a standardized framework

- Mass recompilation for new architecture or MPI version is CPU-bound
  - e.g. 48 hours to rebuild full stack with AVX, SSE3 for legacy cluster support (using 56 cores)
  - AVX512 rebuild took a few days
- More details on Nix & Easybuild

Questions ?