Combining CVMFS, Nix, Lmod, and EasyBuild at Compute Canada

Bart Oldeman, McGill HPC, Calcul Québec, Compute Canada

Motivation

1. New bigger national systems replacing many smaller local clusters, with common software stack, scheduler (Slurm), and so on, administered by national teams.
   Many sites will have no physical cluster but still support.
2. (coming) online:
   a. Arbutus: cloud system, University of Victoria, BC
   b. Cedar: https://docs.computecanada.ca/wiki/Cedar
      Simon Fraser University, Vancouver, BC
   c. Graham: https://docs.computecanada.ca/wiki/Graham
      University of Waterloo, ON
   d. Niagara: https://docs.computecanada.ca/wiki/Niagara
      University of Toronto, ON
   e. Béluga: Calcul Québec, RFP, heterogeneous system with ~40,000 cores and GPUs, Sep. 2018.

compute canada | calcul canada

Online now and coming

Cedar

- 900 nodes, most (690) with (2) 16-core Broadwell sockets at 2.1Ghz, and 128GB memory, others more
- 146 of those nodes have 4 GPUs each (584 P100s)
- 27,696 total cores, ~14PB storage
- Extension:~625 nodes with 48 Skylake cores,192GB/node

Graham

- 1100 nodes, most (1024) with (2) 16-core Broadwell sockets at 2.1Ghz, and 128GB memory, others more
- 160 of those nodes have 2 GPUs each (320 P100s)
- 35,520 total cores, ~13PB storage

Niagara:

- 1500 nodes with 40 Skylake cores,192GB/node
- 60,000 total cores, ~10PB storage

compute canada | calcul canada

Guiding principle

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

All new CC sites

1. Need a distribution mechanism
   a. CVMFS

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

Background

Most HPC clusters use enterprise Linux distributions for good reasons (vendor support for network, parallel filesystems, etc)

CentOS/RHEL 6

Linux kernel 2.6.32, GCC 4.4.7, Glibc 2.12, Python 2.6.6

CentOS/RHEL 7

Linux kernel 3.10, GCC 4.8.5, Glibc 2.17, Python 2.7.5

(with many backports of course)

compare:

Fedora 27

Linux kernel 4.13, GCC 7.2, Glibc 2.26, Python 2.7.14 & 3.6.2

Background

But users on those clusters want shiny new things and install them as if it were a Linux desktop (following documentation):

```
$ sudo apt-get install python2.7-dev
We trust you have received the usual lecture from the local
System Administrator. It usually boils down to these three
things:
    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.
[sudo] password for jsmith:
sudo: apt-get: command not found
$ sudo yum install python27-devel
[sudo] password for jsmith:
Sorry, try again.
[sudo] password for jsmith:
Sorry, user jsmith is not allowed to execute
'/usr/bin/yum install gcc-7.2' as root on lg-1r17-n01.
```

Solution: modules

Create a “modulefile” named “python/2.7.9” somewhere in $MODULEPATH

```
#%Module1.0#############################
proc ModulesHelp { } {
       puts stderr "\tAdds Python 2.7.9 to your environment"
}
module-whatis   "Adds Python 2.7 to your environment"
set  root /software/CentOS-6/tools/python-2.7.9
prepend-path MANPATH         $root/share/man
prepend-path PATH            $root/bin
prepend-path LD_LIBRARY_PATH $root/lib
prepend-path CPATH           $root/include
```

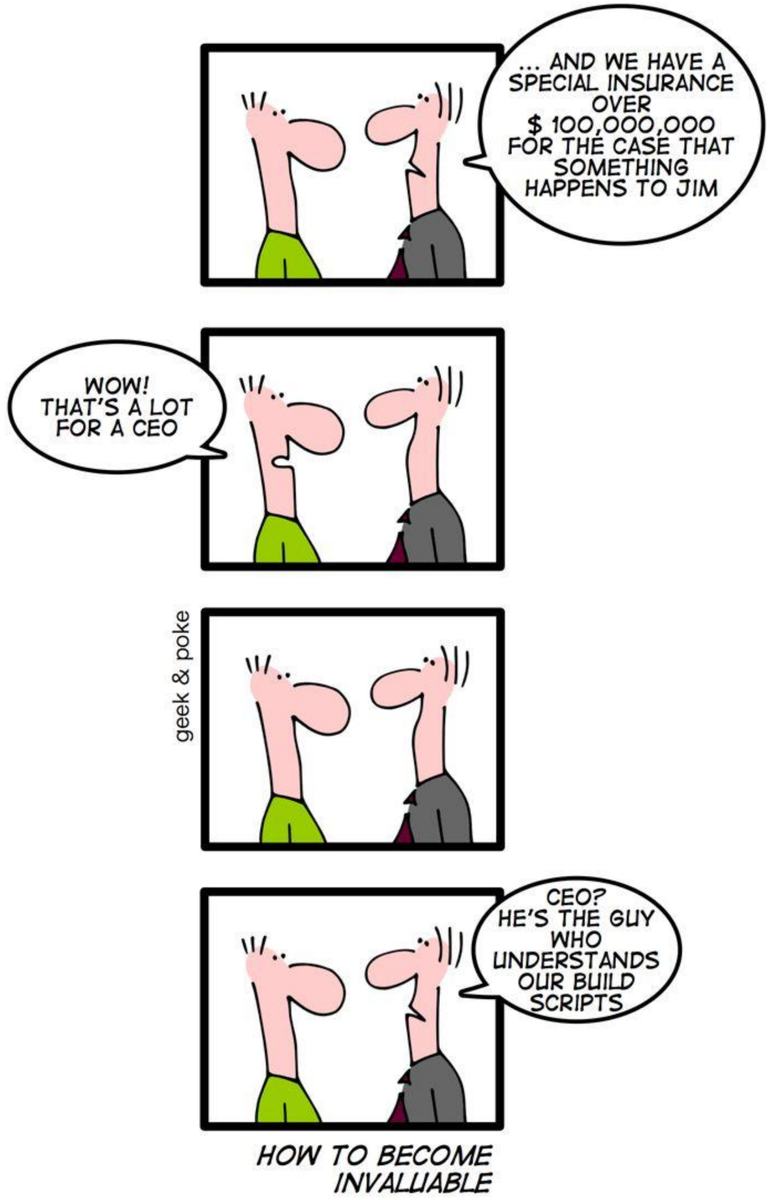
Users do “module load python/2.7.9”, which modifies their environment. “module unload python” restores it then.

Solution: modules

How were modulefiles created: by hand of course, same as how the software was installed.

How to **not** become invaluable:

https://easybuilders.github.io/easybuild/


... AND WE HAVE A SPECIAL INSURANCE OVER $ 100,000,000 FOR THE CASE THAT SOMETHING HAPPENS TO JIM
WOW! THAT'S A LOT FOR A CEO
geek & poke
CEO? HE'S THE GUY WHO UNDERSTANDS OUR BUILD SCRIPTS
HOW TO BECOME INVALUABLE

http://geekandpoke.typepad.com/geekandpoke/2010/05/how-to-become-invaluable.html

Software: design overview

Easybuild layer: modules for Intel, PGI, OpenMPI, MKL, high-level applications. Multiple architectures (sse3, avx, avx2)

`/cvmfs/soft.computecanada.ca/easybuild/{modules,software}/2017`

Easybuild-generated modules around Nix profiles:
GCC, Perl, Qt, Eclipse, 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.

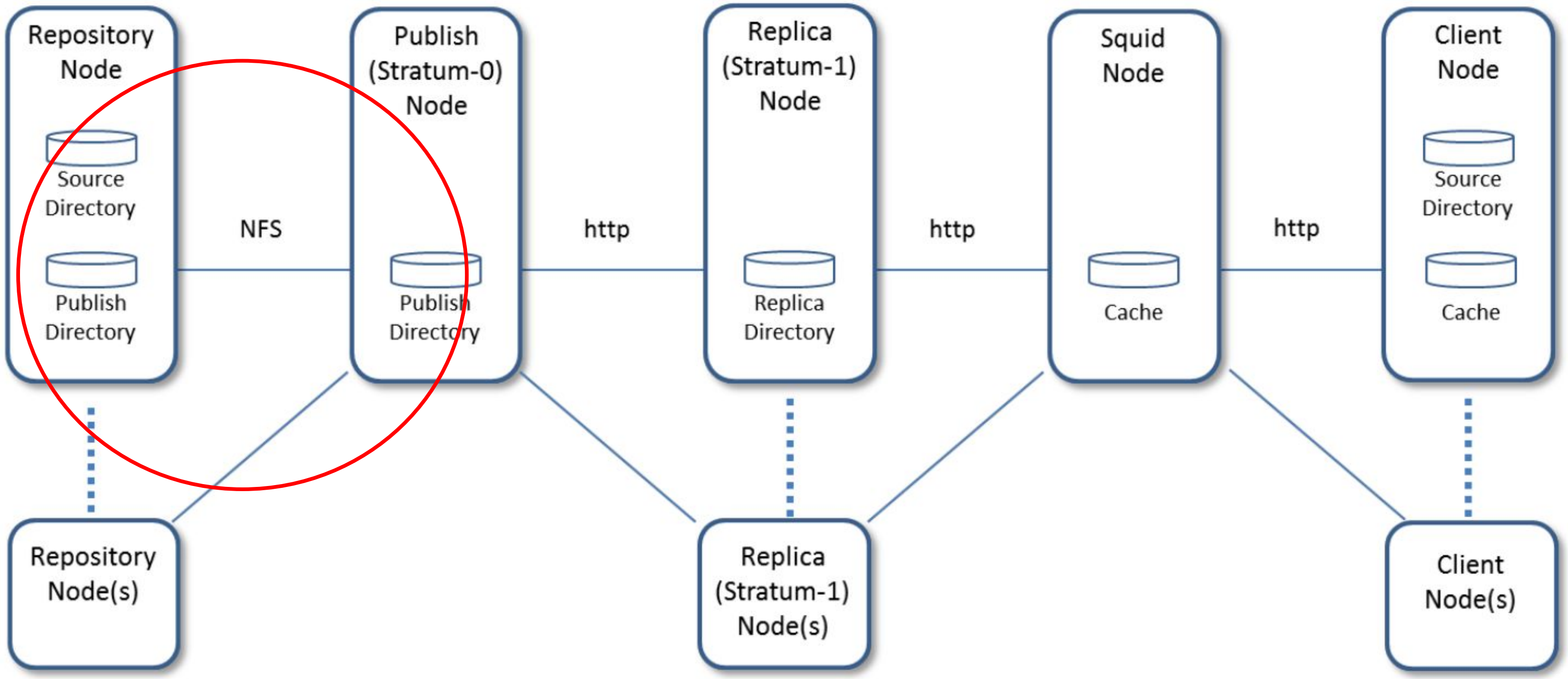
Tools used : CVMFS

CernVM File system

- File system used to distribute software, originally used for High Energy Physics (HEP) software from CERN
- https://cernvm.cern.ch/portal/filesystem
- Distribution layer
  - Redundant
  - Multiple cache layers (Stratum-0, Stratum-1, local squid)
  - Atomic deployment
  - Transparent pull model
- Deploys once => available everywhere
- Carries whatever files we put on it
- Clients mount file system read-only via a FUSE (File System in Userspace) module

compute canada | calcul canada

Tools used : CVMFS



Tools used : CVMFS

- Configuring the client
    - Needs public key
- Three main repositories:
    - /cvmfs/soft.computecanada.ca
    - /cvmfs/soft-dev.computecanada.ca
    - /cvmfs/restricted.computecanada.ca
        - commercial software, with group permissions
- Current clients:
    - cvmfs-client.computecanada.ca
    - cvmfs-client-dev.computecanada.ca
    - most cluster nodes within Compute Canada

Tools used : Nix

- Abstraction layer between the OS and the scientific software stack
- Prevents:
  - Ooops, this software requires an updated glibc
  - Ooops, libX is not installed on this cluster
- Carries all* the dependencies of the scientific software stack
- Ensures all paths are rpath’ed (technically: runpath, so LD_LIBRARY_PATH takes precedence)
- Hundreds of packages supported out of the box
- Can symlink any combination of packages into any multi-generational profile. We use a main “16.09” profile tracking the September 2016 Nixpkgs release

* Exceptions: drivers, kernel modules, etc.

Tools used : EasyBuild

- Automates installation of (mostly) scientifically oriented software and generation of modulefiles.

Tools used : Lmod

- Lua based module system
- Makes it easy to setup a software module hierarchy
  - e.g. modules that depend on MPI implementation X are only visible if you first “modue load X”.
- https://lmod.readthedocs.io/en/latest/

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)

Installing software, step by step

1. Figure out if it should be in Nix or EasyBuild
   - Is the software performance critical or depends on MPI?
     - Yes => EasyBuild
     - Multiple versions needed via modules ?
       - Yes => EasyBuild, or EasyBuild wrapping Nix, using the Nix easyblock
       - No => Nix
2. Install on build-node.computecanada.ca with the appropriate package manager (nix-env or eb)
3. Test on build-node.computecanada.ca
4. Deploy on CVMFS dev repository
5. Test on cvmfs-client-dev.computecanada.ca or with proot
6. Deploy on CVMFS production repository
7. Final testing on the production cluster

compute canada | calcul canada

Software that we put in Nix, not EB

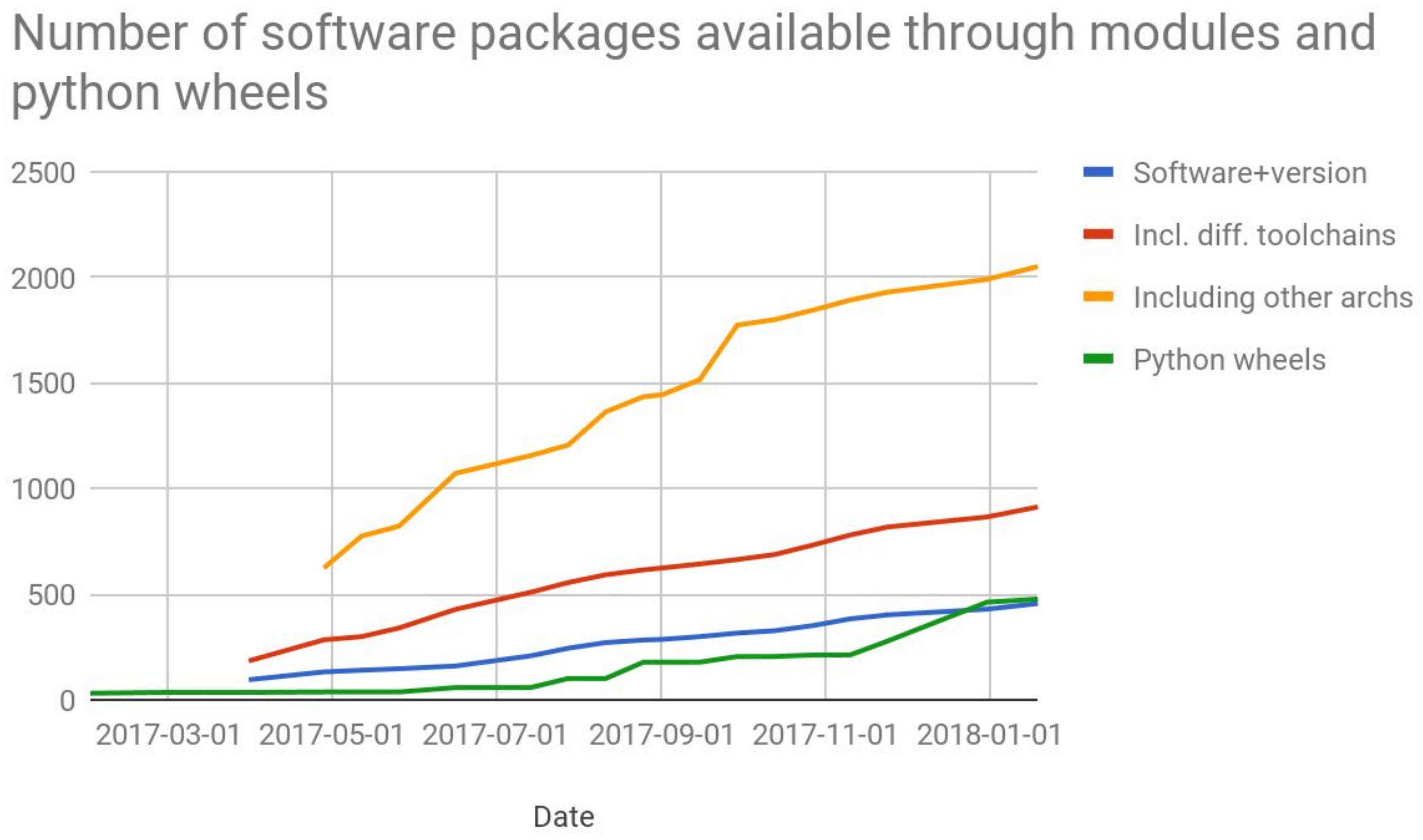
```
Bison,CMake,flex,ncurses,libreadline,bzip2,zlib,binutils,M4,
Autoconf,Automake,libtool,Autotools,Szip,libxml2,sparsehash,
SQLite,cURL,Doxygen,expat,Mesa,libGLU,SWIG,PCRE,
libjpeg-turbo,LibTIFF,libpng,XZ,ant,gettext,X11,pkg-config,
LLVM,libdrm,gperf,FLTK,fontconfig,freetype,GMP,GL2PS,gnuplot,G
raphicsMagick,MPFR,libmatheval,Tcl,Tk,CFITSIO,libX11,libXft,li
bXpm,libXext,makedepend,cairo,libiconv,FFmpeg,GLib,FLANN
```

- mostly things that are dependencies of other modules
- EasyBuild provides recipes for those because the RHEL/CentOS development RPMs were too old
- most of these are of little scientific interest (some sites hide them: the module is automatically loaded but not visible when the user lists all modules using “module avail”).

Python wheels

- Most Python modules are not installed as (Lmod) modules. They are instead provided as binary wheels, stored on the Compute Canada systems under /cvmfs/soft.computecanada.ca/custom/python/wheelhouse/
  - Examples: Tensorflow, scikit-learn, etc, etc.
- The user typically creates a virtual environment and “pip install”s the desired package
  - pip will look in our wheelhouse first before attempting to download it from the internet
  - this gives us properly optimized packages

Some statistics



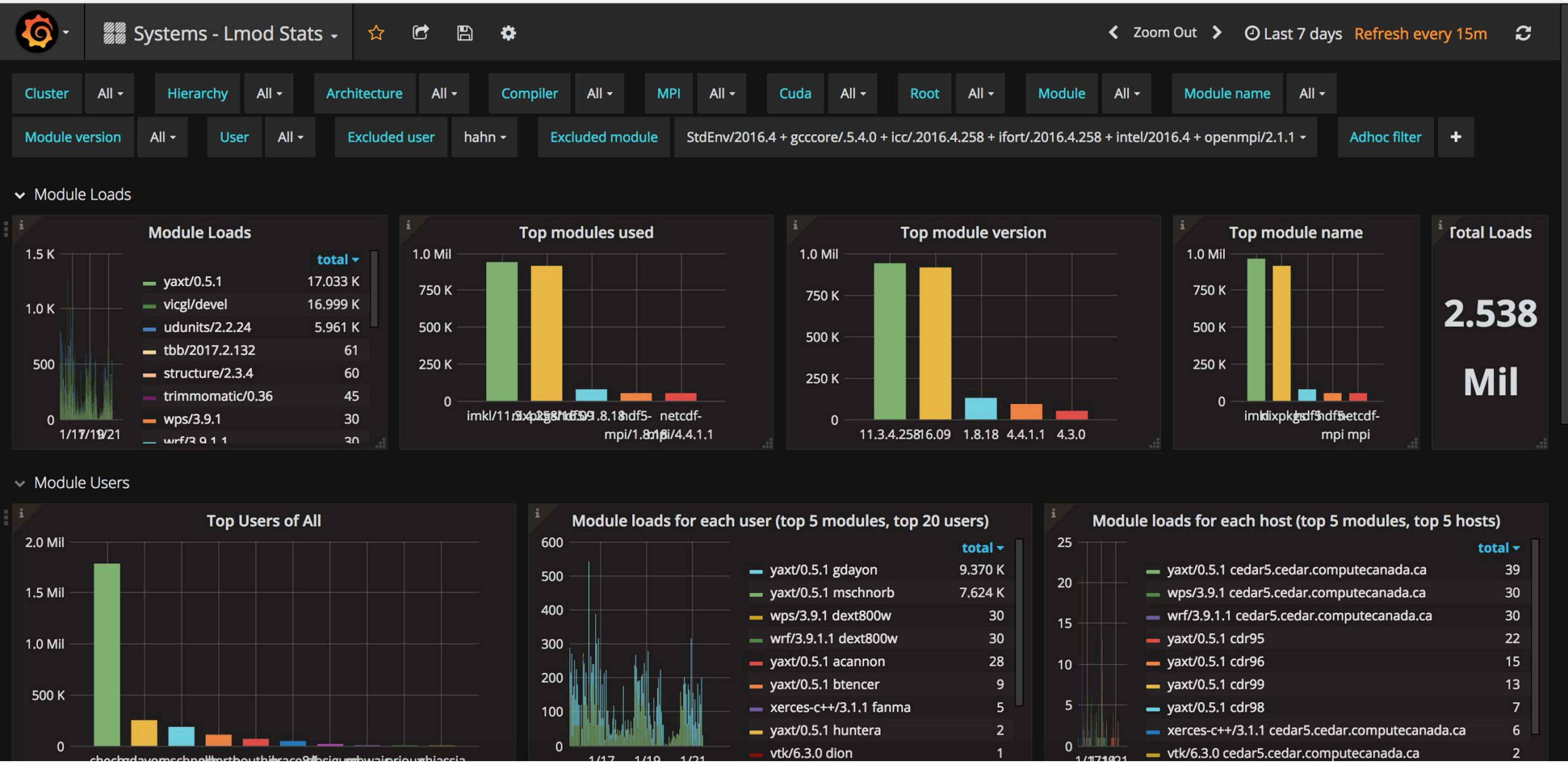
What type of software is it ?

Type of software	Number of modules (S/V)
Artificial intelligence	5
Bioinformatics	145
Chemistry	44
Geo/Earth	18
Input/output	16
Mathematics tools/software	55
MPI libraries	7
Physics software	28
Various tools	93
Visualisation	23

compute canada | calcul canada

Module usage dashboard

https://grafana.computecanada.ca/dashboard/db/systems-lmod-stats



Cluster: All
Hierarchy: All
Architecture: All
Compiler: All
MPI: All
Cuda: All
Root: All
Module: All
Module name: All
Module version: All
User: All
Excluded user: hahn
Excluded module: StdEnv/2016.4 + gcccore/.5.4.0 + icc/.2016.4.258 + ifort/.2016.4.258 + intel/2016.4 + openmpi/2.1.1
Adhoc filter

Module Loads

	total
yaxt/0.5.1	17.033 K
vicgl/devel	16.999 K
udunits/2.2.24	5.961 K
tbb/2017.2.132	61
structure/2.3.4	60
trimmomatic/0.36	45
wps/3.9.1	30

Total Loads: 2.538 Mil

Module loads for each user (top 5 modules, top 20 users)

	total
yaxt/0.5.1 gdayon	9.370 K
yaxt/0.5.1 mschnorb	7.624 K
wps/3.9.1 dext800w	30
wrf/3.9.1.1 dext800w	30
yaxt/0.5.1 acannon	28
yaxt/0.5.1 btencer	9
xerces-c++/3.1.1 fanma	5
yaxt/0.5.1 huntera	2
vtk/6.3.0 dion	1

Module loads for each host (top 5 modules, top 5 hosts)

	total
yaxt/0.5.1 cedar5.cedar.computecanada.ca	39
wps/3.9.1 cedar5.cedar.computecanada.ca	30
wrf/3.9.1.1 cedar5.cedar.computecanada.ca	30
yaxt/0.5.1 cdr95	22
yaxt/0.5.1 cdr96	15
yaxt/0.5.1 cdr99	13
yaxt/0.5.1 cdr98	7
xerces-c++/3.1.1 cedar5.cedar.computecanada.ca	6
vtk/6.3.0 cedar5.cedar.computecanada.ca	2

Software challenges

<u>Non-standard prefix</u>

`$EBROOTNIXPKGS=$NIXUSER_PROFILE=`
`/cvmfs/soft.computecanada.ca/nix/var/nix/profiles/16.09`
instead of `/usr`.
Mostly transparent to users but occasional (ab)use of `LD_LIBRARY_PATH`:

1. By users (mostly by accident in old `.bashrc` files)
2. By binary-only software and their scripts (e.g. ANSYS)
3. `setrpaths.sh` script patches (patchelf) binaries so they can work with this prefix.
4. Do not set `LD_LIBRARY_PATH` to either `/usr/lib64` or `$EBROOTNIXPKGS/lib`. Either setting will burn you.

So far mostly resolved; if all else fails, (e.g. user wants to compile GCC) use `module --force purge`. We may be able to make the stack more immune in future, e.g. using old-style RPATH, Singularity if necessary.

compute | calcul
canada | canada

Challenge: there is more than /usr!

Loading manually written nixpkgs/16.09 module by default, including

```
local root = "/cvmfs/soft.computecanada.ca/nix/var/nix/profiles/16.09"
setenv("NIXUSER_PROFILE", root)
prepend_path("PATH", "/cvmfs/soft.computecanada.ca/custom/bin")
prepend_path("PATH", pathJoin(root, "sbin"))
prepend_path("PATH", pathJoin(root, "bin"))
prepend_path("LIBRARY_PATH", pathJoin(root, "lib"))
prepend_path("C_INCLUDE_PATH", pathJoin(root, "include")) -- NOT CPATH!!
prepend_path("CPLUS_INCLUDE_PATH", pathJoin(root, "include"))
prepend_path("MANPATH", pathJoin(root, "share/man"))
prepend_path("ACLOCAL_PATH", pathJoin(root, "share/aclocal"))
prepend_path("PKG_CONFIG_PATH", pathJoin(root, "lib/pkgconfig"))
setenv("FONTCONFIG_FILE", pathJoin(root, "etc/fonts/fonts.conf"))
prepend_path("CMAKE_PREFIX_PATH", root)
prepend_path("PYTHONPATH","/cvmfs/soft.computecanada.ca/custom/python/site-packa
ges")
setenv("PERL5OPT", "-I" .. pathJoin(root, "lib/perl5") .. " -I" ..
pathJoin(root, "lib/perl5/site_perl"))
prepend_path("PERL5LIB", pathJoin(root, "lib/perl5/site_perl"))
prepend_path("PERL5LIB", pathJoin(root, "lib/perl5"))
setenv("TZDIR", pathJoin(root,"share/zoneinfo"))
setenv("SSL_CERT_FILE", "/etc/pki/tls/certs/ca-bundle.crt")
setenv("CURL_CA_BUNDLE", "/etc/pki/tls/certs/ca-bundle.crt")
setenv("LESSOPEN", "|" .. pathJoin(root, "bin/lesspipe.sh %s"))
setenv("LOCALE_ARCHIVE", pathJoin(root, "lib/locale/locale-archive"))
```

Catches most searches for EasyBuilds/Best not to have any -devel RPMs installed.

Challenge: nix store leaks

Nix provides a symlink forest:

```
.../nix/var/nix/profiles/16.09 ->

.../nix/var/nix/profiles/16.09-523-link ->
.../nix/store/cj3f56cgpms7m9fjnbl9vjkmap5fzgsi-user-environment

.../nix/store/cj3f56cgpms7m9fjnbl9vjkmap5fzgsi-user-environment/bin/ls ->
.../nix/store/cn222k5axppndcfbqlckj57939d9h0h9-coreutils-8.25/bin/ls
```

We wrap ld so all rpaths in EB/user code point to .../nix/var/nix/profiles/16.09/lib. This way Nix components can be upgraded, which changes the store hashes, and allows garbage collect / selective copying.

Sometimes that did not work:

- Python virtualenv: copies the python binary into the virtualenv with store rpaths embedded.
- Qmake: qmake -query QT_INSTALL_BINS
  /cvmfs/soft.computecanada.ca/nix/store/
  vxwrgncd38s5prw8qx99rnsfz6lgph52-qtbase-5.6.1-1/bin

compute canada | calcul canada

Credits

- Thanks to others in Compute Canada:
  - RSNT (Research Support National Team):
    - Led by Maxime Boissonneault, responsible for setting up this software stack (+ documentation + ticketing system).
  - Nix experts on the sideline (Tyson Whitehead, Servilio Afre Puentes).
  - Kuang Chung Chen, who started combining CVMFS, Nix and EasyBuild, after hitting the limits of Linux From Scratch.
- And thanks to EasyBuild: UGent, JSC, Robert Schmidt, ...