EasyBuild + Nix + CVMFS @ ComputeCanada

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.

Cedar

Supplier: Scalar

System summary:

- 900 nodes, most (690) with (2) 16-core Broadwell E5-2683 v4 processors at 2.1Ghz, and 128GB memory, others more memory
  - 27,696 total cores; 190TB total memory
- 146 of those nodes have 4 GPUs each (584 P100s)
- Interconnect: 22 fully connected "islands" of 32 base or large nodes each have 1024 cores in a fully non-blocking topology (Omni-Path fabric)
- ~14PB Lustre-based high-performance storage
- Extension: ~625 nodes with Skylake 8160 CPUs (48 cores/node), 192GB/node.

Peak theoretical speed: 3.6PF (5.6PF with ext)

Graham

Supplier: Huawei

System summary:

- 1100 nodes, most (1024) with (2) 16-core Broadwell E5-2683 v4 processors at 2.1Ghz, and 128GB memory
  - 35,520 total cores; 166TB total memory
- 160 of those nodes have 2 GPUs each (320 P100s)
- Interconnect: Mellanox FDR (GPU:56Gb/s) and EDR (CPU:100Gb/s) InfiniBand interconnect. One 324-port director switch aggregates connections from islands of 1024 cores each for CPU and GPU nodes.
- ~13PB Lustre-based high-performance storage

Peak theoretical speed: 2.6PF

Niagara

Supplier: Lenovo

System summary:

- 1500 nodes, each with (2) 20-core Skylake 6148 processors at 2.4Ghz, and 192GB memory
  - 60,000 total cores; 288TB total memory
- Interconnect: Mellanox EDR 7-wing “Dragonfly” adaptive routing (no core switch!)
- ~10PB GPFS-based high-performance storage
- 256TB burst buffer, based on NVMe-as-a-fabric, yielding up to 161GB/s

Peak theoretical speed: 4.61PF

See also:

https://wiki.scinet.utoronto.ca/wiki/images/0/00/Intro_Niagara.pdf

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
   1. CVMFS

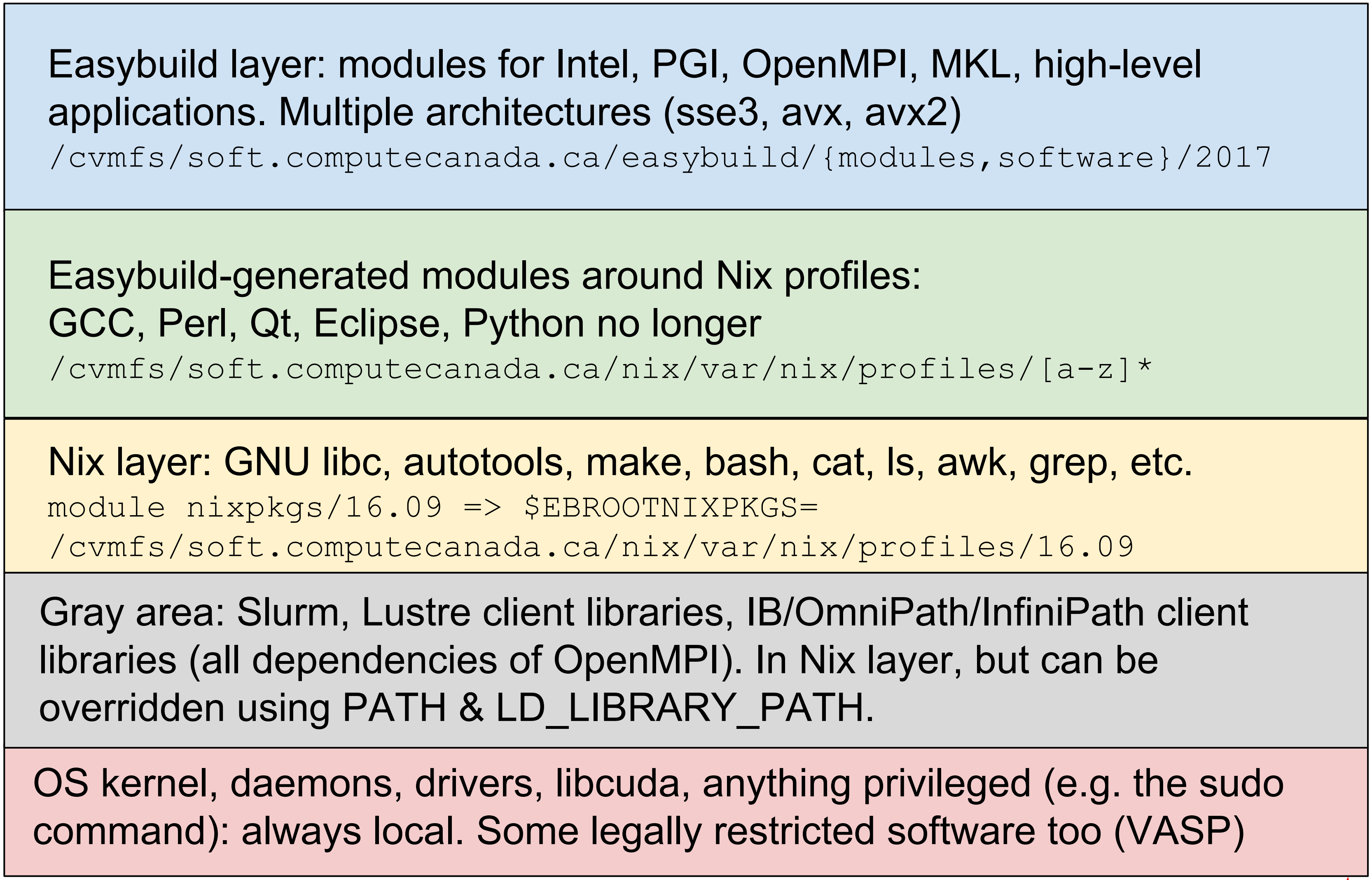
Consistency

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

Easy to use

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

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
```

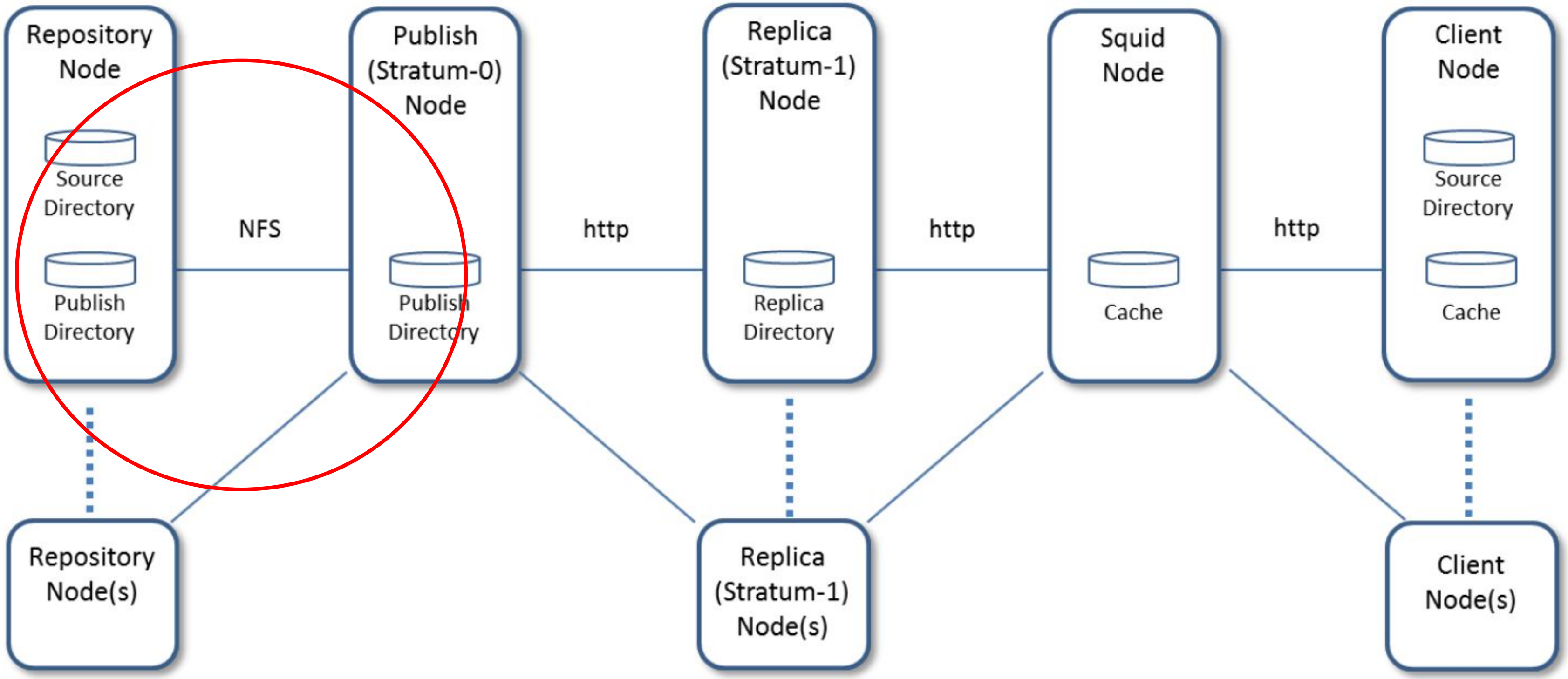
Gray area: Slurm, Lustre client libraries, IB/OmniPath/InfiniPath client libraries (all dependencies of OpenMPI). In Nix layer, but can be overridden using PATH & LD_LIBRARY_PATH.

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

Tools used : CVMFS

- 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

Tools used : CVMFS



compute | calcul
canada | canada

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

- Preaching to the choir

Tools used : Lmod

- Preaching to the choir

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

build-node.computecanada.ca

1. Nix: Searching for packages :

   ```
   nix-env -qasPp $NIXUSER_PROFILE [package name]
   ```

2. Nix: Installing existing packages (builds packages via `nix-daemon` in special chroot.)
   a. In your user's environment

      ```
      nix-env -iA <package attribute name> [--dry-run]
      ```

   b. Globally :

      ```
      sudo -i -u nixuser nix-env -iA <package attr. name>
      ```

3. EasyBuild: Searching for packages :

   ```
   eb -S REGEX
   ```

4. EasyBuild: Installing existing packages :
   a. In your user's environment `eb <name of easyconfig>`
   b. Globally : `sudo -u ebuser -i eb <easyconfig>`

Deploying to CVMFS on build-node.computecanada.ca

1. Switch to special user

```
sudo su - libuser
```

2. Start CVMFS transaction

```
sudo /etc/rsnt/start_transaction <dev|prod>
```

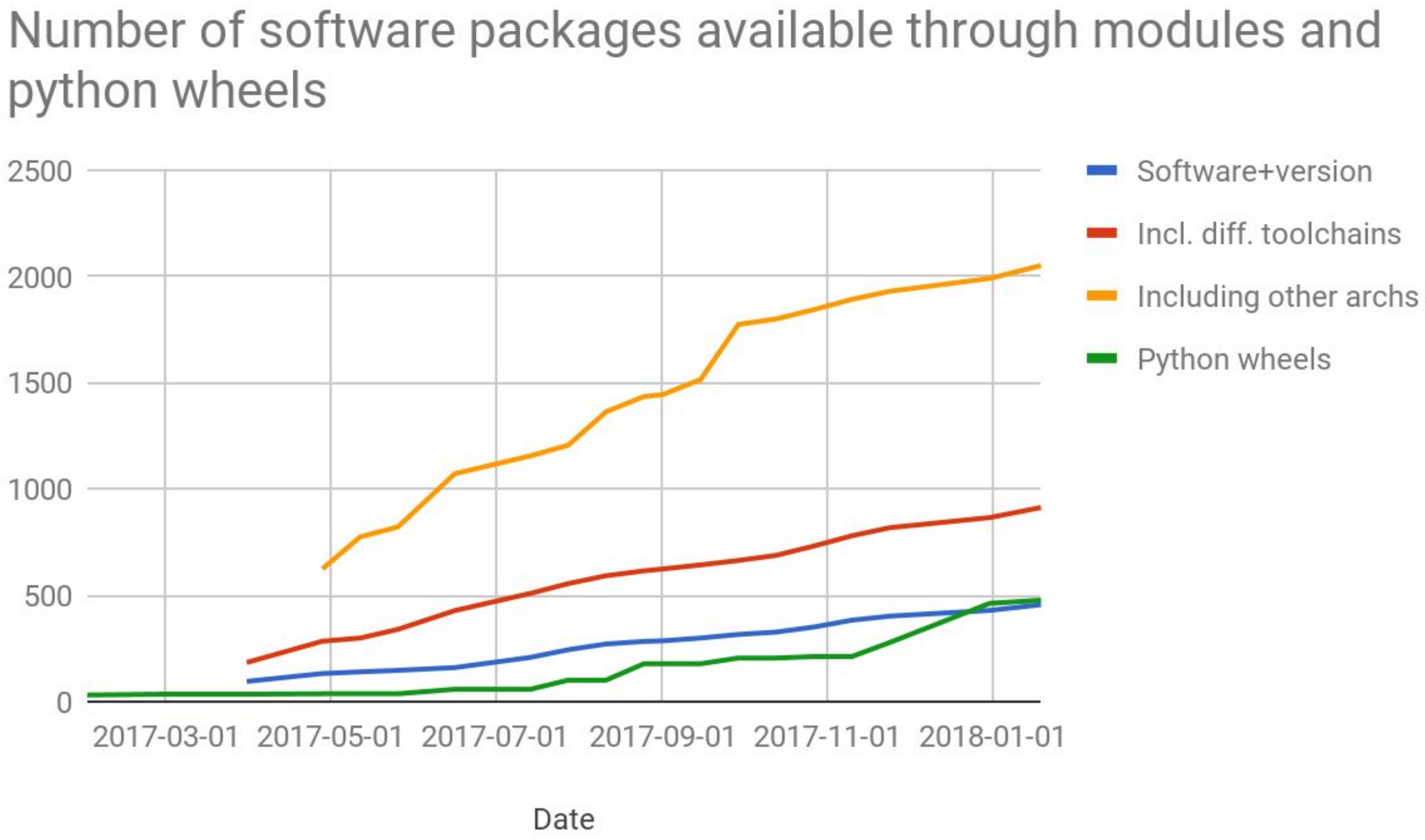
3. Synchronize the files via rsync and sshfs to stratum-0

```
/etc/rsnt/rsnt-sync \
   --what <nix|config|easybuild> \
   [--repo <dev|prod>] \
   [--path <source path>] [--dry-run] ...
```

4. Publish or abort CVMFS transaction

```
sudo /etc/rsnt/abort_transaction <dev|prod>
sudo /etc/rsnt/publish_transaction <dev|prod>
```

Some statistics



compute canada | calcul canada

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

Who is installing software ?

180	Masao Fujinaga
148	Bart Oldeman
111	Maxime Boissonneault
11	Belaid Moa
11	Ata Roudgar
10	Ali Kerrache
8	Pawel Pomorski
8	Charles Coulombe
7	Jeffrey Stafford
7	Félix-Antoine Fortin
5	Oliver Stueker
5	Robert Wagner
3	Erik Spence
2	Pier-Luc St-Onge
1	Ramses van Zon
1	Doug Roberts
1	Hartmut Schmider
1	Fei Mao
1	Chris Geroux

Number of distinct software packages (excluding multiple versions or multiple architectures) installed.
Only counts easybuild (modules)
Includes failed attempts

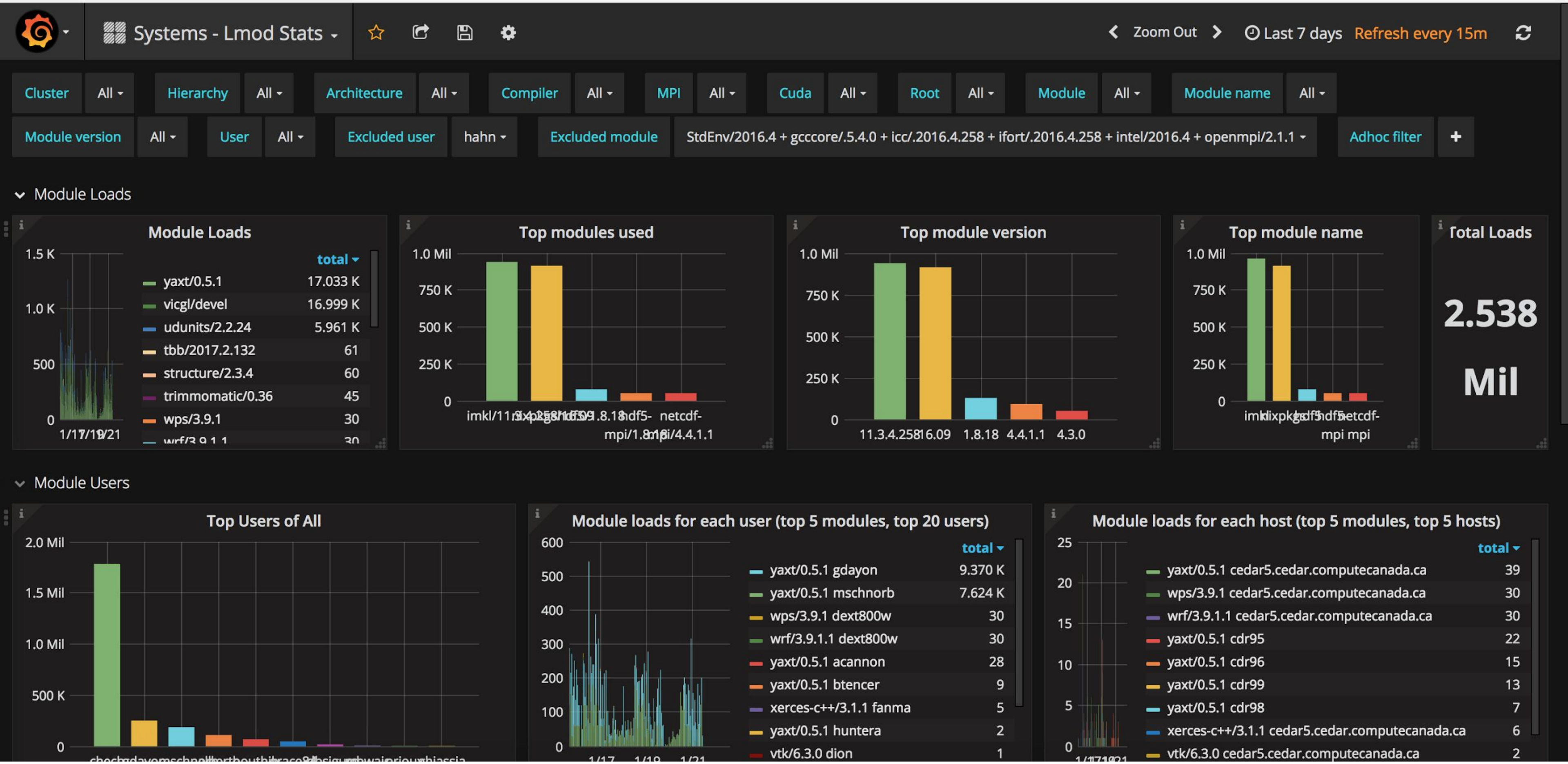
Supported/upcoming licensed packages

MATLAB	VASP(*)	GAUSSIAN (*)
ADF (*)	LS-DYNA	COMSOL
Materials Studio (WIP)	FDTD Lumerical	ANSYS Suite
Star-CCM+	DL Poly4	CPMD
ORCA	Abaqus	Allinea (*)
CellRanger	deMon2k	
CPLEX(**)	Gurobi(**)	CFOUR(**)
Wien2k(**)	MAKER(**)	

(*) special case (**) upcoming

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

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
wrf/3.9.1.1	30

Top modules used

Top module version

Top module name

Total Loads

2.538 Mil

Module Users

Top Users of All

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

What is installed via EasyBuild

Automatically generated list available here:

https://docs.computecanada.ca/wiki/Available_software

Note: Python packages via virtualenv+wheelhouse, except scipy-stack+mpi4py.

Main toolchains used:

current: iccifort & iimkl & iompi & iomkl/.2016.4.11 (+CUDA 8) =
GCCcore 5.4.0 + icc/ifort/imkl 2016.4.258 + OpenMPI 2.1.1

upcoming: iccifort & iimkl & iompi & iomkl/.2017.5.211 (+CUDA 9) =
GCCcore 6.4.0 + icc/ifort/imkl 2017.[45].239 + OpenMPI 2.1.1

Other toolchains used (GNU/Intel/PGI):

GCCcore, GCC, gcccuda, **gmkl, gmklc,** gompi, gompic, foss, gomkl, gomklc,

iccifort, iccifortcuda, **iimkl, iimklc,** iimpi, iompi, iompic, intel, iomkl, iomklc,

PGI, **pmkl**, pompi, pomkl,

(not user-visible unless users use eb themselves).

gmkl, iimkl with MKL at Core level?

- Many packages use linear algebra but not MPI.
- Examples: Julia, NAMD-verbs, Octave, ROOT, SuiteSparse
- In a hierarchical scheme such packages can be installed at the compiler level rather than at the top (MPI) level.
- MKL can be installed at the Core level without interfaces (which means no parallel FFTW interfaces and no FFTW2 wrappers but everything else is there).
- The `iimkl` toolchain can then be used to compile packages.
- Framework support for `iimkl` is now in EB 3.1.
- Upstream `iimkl` easyconfigs would probably best put MKL (`imkl`) at the compiler level.

EasyBuild configuration

```
$ cat $EASYBUILD_CONFIGFILES
[config]
buildpath = /dev/shm
modules-tool = Lmod \n module-syntax = Lua
prefix = /cvmfs/soft.computecanada.ca/easybuild
subdir-modules = modules/2017
subdir-software = software/2017
subdir-user-modules = .local/easybuild/modules/2017
suffix-modules-path =
module-naming-scheme = SoftCCHierarchicalMNS
recursive-module-unload = 1
repository = GitRepository \n repositorypath = %(prefix)s/ebfiles_repo.git
robot-paths = %(prefix)s/easyconfigs:%(prefix)s/ebfiles_repo
hide-deps = icc,ifort,GCCcore
filter-deps = Bison,CMake,flex,ncurses,libreadline,bzip2,zlib,binutils,M4,
Autoconf,Automake,libtool,Autotools,Szip,libxml2,sparsehash,SQLite,cURL,Doxy
gen,expat,Mesa,libGLU,SWIG,PCRE,libjpeg-turbo,LibTIFF,libpng,XZ,ant,gettext,
X11,pkg-config,LLVM,libdrm,gperf,FLTK,fontconfig,freetype,GMP,GL2PS,gnuplot,
GraphicsMagick,MPFR,libmatheval,Tcl,Tk,CFITSIO,libX11,libXft,libXpm,libXext,
makedepend,cairo,libiconv,FFmpeg,GLib,FLANN
ignore-osdeps = 1
filter-env-vars = LD_LIBRARY_PATH
minimal-toolchains = 1 \n add-dummy-to-minimal-toolchains = 1
hide-toolchains = GCCcore,iompi,iomkl
parallel = 8 \n use-ccache=/cvmfs/local/ccache
allow-loaded-modules=nixpkgs
```

compute canada | calcul canada

Software challenges

Non-standard prefix

`$EBROOTNIXPKGS=$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.

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: binary-only software

A script setrpaths.sh patchelf’s all binaries and libraries in a directory and all its subdirectories, to use the Nix dynamic linker and runpath. E.g. ANSYS-18.2.eb

```
postinstallcmds = [
# find all non-binary files containing  [:"]/usr/lib or [:"]/lib on one
line and remove them from the paths
"for f in $(grep -rIl '[:\"]/usr/lib\|[:\"]/lib' %(installdir)s); do
    echo Modifying file $f;
    sed -i -e '/[:\"]\/usr\/lib/s/:*\/usr\/lib[^:\"]*//g'
           -e '/[:\"]\/lib/s/:*\/lib[^:\"]*//g' $f;
done",
# rename the built-in libstdc++.so* to libstdc++.so*.bak because they
are older than what we have in Nix & cause problems with other binaries
"find %(installdir)s -name 'libstdc++.so*' -exec mv {} {}.bak \;",
# call setrpaths.sh on all subdirectories called bin,lib,lib64,lnamd64
"for d in $(find %(installdir)s -name 'bin' -o -name 'lib' -o -name
'lib64' -o -name 'lnamd64'); do
    echo Calling setrpaths.sh --path $d;
    /cvmfs/soft.computecanada.ca/easybuild/bin/setrpaths.sh
         --path $d;
done"]
```

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

Challenge: python

We use $EBROOTPYTHON to avoid PYTHONPATH from modules overriding virtualenv-installed modules… e.g. Python-3.6.3-dummy-dummy.eb

```
modextrapaths = {'PYTHONPATH':
['/cvmfs/soft.computecanada.ca/easybuild/python/site-packages']}
modluafooter = """
local arch = os.getenv("RSNT_ARCH") or ""
if arch == "avx2" then -- setup wheelhouse
    setenv("PIP_CONFIG_FILE",
"/cvmfs/soft.computecanada.ca/config/python/pip-avx2.conf") ...
```

Combine with

```
/cvmfs/soft.computecanada.ca/easybuild/python/site-packages/sitecustomize.py:
if "EBPYTHONPREFIXES" in os.environ:
    postfix = os.path.join('lib', 'python'+sys.version[:3], 'site-packages')
    for prefix in os.environ["EBPYTHONPREFIXES"].split(os.pathsep):
        sitedir = os.path.join(prefix, postfix)
        if os.path.isdir(sitedir):
            site.addsitedir(sitedir)
```

Scipy-Stack-2017b-dummy-dummy.eb (bundle with numpy, etc):

```
modextrapaths = {'EBPYTHONPREFIXES': ['']}
modluafooter = 'depends_on("python")'
```

=> This module works with all Pythons.

Challenge: multiple architectures

To deal with sse3/avx/avx2, we use a wrapper script around eb containing:

```
if [ "$RSNT_ARCH" == avx2 ]; then
  export EASYBUILD_OPTARCH='Intel:xCore-AVX2;GCC:march=core-avx2;GCCcore:GENERIC'
elif [ "$RSNT_ARCH" == avx ]; then
  export EASYBUILD_REPOSITORY='FileRepository'
  export
EASYBUILD_REPOSITORYPATH=/cvmfs/soft.computecanada.ca/easybuild/ebfiles_repo_$RSNT
_ARCH
  export EASYBUILD_ROBOT_PATHS=/cvmfs/soft.computecanada.ca/easybuild/ebfiles_repo
  export EASYBUILD_OPTARCH='Intel:xAVX;GCC:march=corei7-avx;GCCcore:GENERIC'
elif [ "$RSNT_ARCH" == sse3 ]; then …
```

where

```
--- a/easybuild/toolchains/gcccore.py
+++ b/easybuild/toolchains/gcccore.py
    NAME = 'GCCcore' ...
 +  COMPILER_FAMILY = NAME
    SUBTOOLCHAIN = DUMMY_TOOLCHAIN_NAME
```

and e.g iccifort modules do (via modluafooter)

```
prepend_path("MODULEPATH",
pathJoin("/cvmfs/soft.computecanada.ca/easybuild/modules/2017",
os.getenv("RSNT_ARCH"), "Compiler/intel2017.5"))
```

Our custom naming scheme puts both dummy & GCCcore modules at the top-level “Core” directory.

EasyBuild wrapping nix

Idea: Nix can install software in a custom profile, e.g. GCCcore:

```
.../nix/var/nix/profiles/gcc-6.4.0 ->
.../nix/var/nix/profiles/gcc-6.4.0-4-link ->
.../nix/store/7h3hph01xzri1jr9vb12hhiywc5j1wz0-user-environment ->
(symlinks into nix store),
```

using:

```
sudo -u nixuser -i nix-env -iA gfortran6.cc -p \
   .../nix/var/nix/profiles/gcc-6.4.0
```

This Nix command can be wrapped in an easyblock, and EasyBuild can then set up a module where

```
installdir=$EBROOTGCCCORE=.../nix/var/nix/profiles/gcc-6.4.0
```

We did similar things for Python (but issues with virtualenv, so not any more since November 2017)

Nix wrapping EasyBuild

Idea: Nix can use EasyBuild to build software

- Eliminates the need to translate easyblocks and easyconfigs to Nix expressions (their name for build recipes).
- More complex: needs to deal with build-dependencies and dependencies in a more isolated environment.

Another approach by Robert Schmidt:

https://github.com/rjeschmi/nix-easybuild

We borrowed some Nix expressions from there (Lmod, vsc-* Python packages).

… and for a deep dive

Using build-node.computecanada.ca guide (30 pages + appendices)

https://docs.google.com/document/d/111i1aCe79cYTkxmkpdN5tPihWk7ADcgCngaMRq0kTPA/edit#heading=h.axuun2p60as7

Guide is public

Access to build-node is restricted to CC Team members

Installation privileges are granted to CC Team members on a per-request basis.

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, ...