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 (7640 cores, September 2016)
   b. Cedar: https://docs.computecanada.ca/wiki/Cedar
      Simon Fraser University, Vancouver, BC (27696 cores, 584 GPUs, April 2017)
   c. Graham: https://docs.computecanada.ca/wiki/Graham
      University of Waterloo, ON (33472 cores, 320 GPUs April 2017)
   d. Niagara: University of Toronto, ON
      (~66000 cores, late 2017)

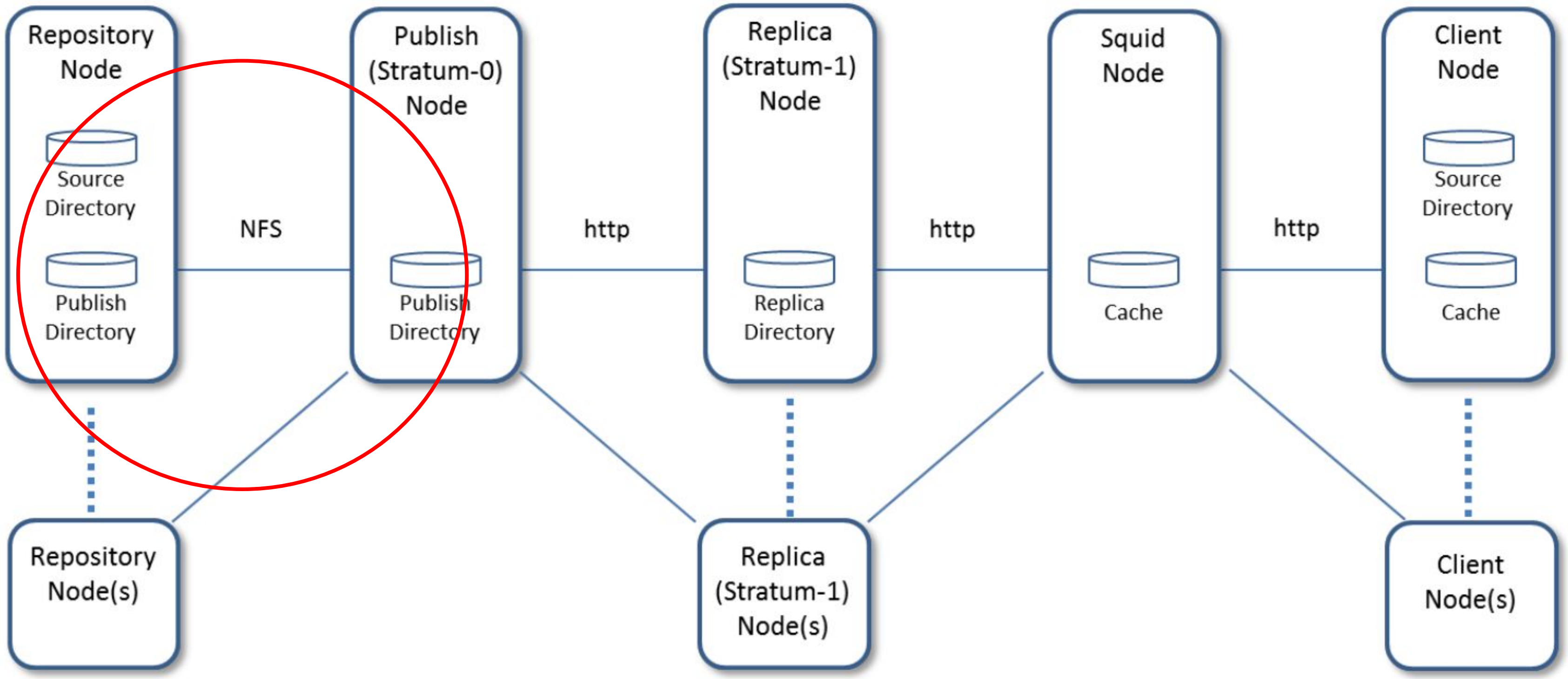
compute canada | calcul canada

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

compute canada | calcul canada

Tools used : CVMFS



Tools used : CVMFS

- Configuring the client
    - Needs public key
- Two repositories:
    - /cvmfs/soft.computecanada.ca
    - /cvmfs/soft-dev.computecanada.ca
- Current clients:
    - cvmfs-client.computecanada.ca
    - cvmfs-client-dev.computecanada.ca

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

compute canada | calcul canada

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)

compute canada | calcul canada

Installing software, step by step

1. Figure out if it should be in Nix or EasyBuild
2. Install on build-node.computecanada.ca with the appropriate package manager
3. Test on build-node.computecanada.ca
4. Deploy on CVMFS dev repository
5. Test on cvmfs-client-dev.computecanada.ca or on a cluster with proot
6. Deploy on CVMFS production repository
7. Final testing on the production cluster

Nix or EasyBuild ?

1. Is the software performance critical or depends on MPI?
    a. Yes => EasyBuild
    b. Do you expect that multiple versions will be needed, swappable through modules ?
        i. Yes => EasyBuild, or EasyBuild wrapping Nix, using the Nix easyblock
        ii. No => Nix

compute canada | calcul canada

Using Nix on build-node.computecanada.ca

1. Searching for packages :

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

2. Installing existing packages :
   a. In your user's environment
   ```
   nix-env -iA <package attribute name> [--dry-run]
   ```
   b. Globally :
   ```
   sudo -i -u nixuser nix-env -iA <package attribute name>
   ```
   This builds packages via `nix-daemon` in a special chroot.

compute canada | calcul canada

Using EasyBuild on build-node.computecanada.ca

1. Searching for packages :

```
eb -S REGEX
```

2. Installing existing packages :
   a. In your user's environment

```
eb <name of easyconfig file> --prefix=$HOME/easybuild
```

   (need to investigate `--pretend`!)

   b. Globally :

```
sudo -u ebuser -i eb <name of easyconfig file>
```

   c. Use `eb-intel-avx2` instead of `eb` to create code that only works on Haswell and up.

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>
   ```

What is installed via Nix

```
$ nix-env -qp $NIXUSER_PROFILE # the current 16.09 profile
```

```
autoconf-2.69          gettext-0.19.8          libibumad-1.3.10.2 nedit-5.6a
automake-1.15          gfortran-5.4.0-lib      libibverbs-1.1.8   net-tools-1.60_p2
bash-completion-2.1    gfortran-wrapper-5.4.0  libICE-1.0.9       0120127084908
bash-interactive-4.3   git-minimal-2.10.0      libjpeg-turbo-1.5.0nix-1.11.4
-p48                   glib-2.48.2             libpng-apng-1.6.23 numactl-2.0.10
bazaar-2.7.0           glibc-2.24              libSM-1.2.2        opa-psm2-10.2.42
bison-3.0.4            glu-9.0.0               libtiff-4.0.6      openssh-7.3p1
bzip2-1.0.6.0.1        glxinfo-8.3.0           libtool-2.4.6      openssl-1.0.2j
cairo-1.14.6           gmp-6.1.1               libunwind-1.1      pango-1.40.2
cmake-3.6.0            gnugrep-2.25            libuv-1.9.1        parallel-20160722
coreutils-8.25         gnum4-1.4.17            libX11-1.6.4       patch-2.7.5
cpio-2.12              gnumake-4.2.1           libXext-1.3.3      patchelf-0.9
curl-7.51.0            gnuplot-5.0.3           libXft-2.3.2       pcre-8.39
cvs-1.12.13            gnused-4.2.2            libxml2-2.9.4      pcre2-10.21
db-5.3.28              gnutar-1.29             libXmu-1.1.2       perl-5.22.2
diffutils-3.5          guile-2.0.13            libXt-1.1.5        pkg-config-0.29
emacs-25.1             gzip-1.8                llvm-3.9.0         proot-5.1.0
findutils-4.6.0        hwloc-1.11.2            Lmod-7.3.4         psmisc-22.21
flex-2.6.1             imake-1.0.7             lmutil-11.13.1.3   python-2.7.13
fontconfig-2.11.1      infinipath-psm-3.3      lsb-release-1.4    ...
freetype-2.6.5         less-483                man-db-2.7.5
gawk-4.1.3             lesspipe-1.82           mc-4.8.17
gdb-7.11.1             libffi-3.2.1            mercurial-3.8.2
gdbm-1.12              libgit2-0.24.3          mesa-noglu-12.0.5
```

compute canada | calcul canada

What is installed via EasyBuild

```
$ module spider | grep "^  [a-z]"
```

```
  arpack-ng: arpack-ng/3.4.0
  boost: boost/1.62.0
  bowtie: bowtie/1.1.2
  bowtie2: bowtie2/2.3.0
  cuda: cuda/8.0.44
  cudnn: cudnn/5.1
  fftw: fftw/3.3.5
  gcc: gcc/4.8.5, gcc/5.4.0
  gromacs: gromacs/2016
  gsl: gsl/2.2.1
  hdf5: hdf5/1.8.18
  hpl: hpl/2.2
  imkl: imkl/2017.1.132
  intel: intel/2016.4.258, intel/2017.1.132 (local name for "iccifort!")
  jags: jags/4.2.0
  java: java/1.8.0_121
  mcr: mcr/R2013a, mcr/R2014a, mcr/R2014b, mcr/R2015a, mcr/R2015b,
mcr/R2016a, mcr/R2016b
  metis: metis/5.1.0
  openmpi: openmpi/2.0.2
  python: python/2.7.13, python/3.5.2
  python27-scipy-stack: python27-scipy-stack/2017a
  suitesparse: suitesparse/4.5.4
  tbb: tbb/2017.2.132
```

compute canada | calcul canada

What is installed via EasyBuild

$ module --show-hidden av # after loading intel (=iccifort) and openmpi

```
 -------------------------------- MPI-dependent avx2 modules ------------
  boost/1.62.0    fftw/3.3.5    gromacs/2016    hdf5/1.8.18    hpl/2.2

------------------------------ Compiler-dependent avx2 modules -----------
  arpack-ng/3.4.0           gsl/2.2.1                python27-scipy-stack/2017a
  bowtie/1.1.2     (bio)    jags/4.2.0              suitesparse/4.5.4
  bowtie2/2.3.0    (bio)    metis/5.1.0    (D)
  cuda/8.0.44               openmpi/2.0.2 (L,m)

-------------------------------- Compiler-dependent modules --------------
  metis/5.1.0

-------------------------------------- Core Modules ----------------------
  StdEnv/2017a        (S,L)    intel/2016.4.258 (t)                 mcr/R2014b
  gcc/4.8.5           (t)      intel/2017.1.132 (L,t,D:17:2017) mcr/R2015a
  gcc/5.4.0           (t,D)    iomkl/.2017a     (H)                 mcr/R2015b
  icc/.2016.4.258     (H)      iomklc/.2017a    (H)                 mcr/R2016a
  icc/.2017.1.132     (H,L)    iompi/.2017a     (H)                 mcr/R2016b(D)
  ifort/.2016.4.258 (H)        iompic/.2017a    (H)                  python/2.7.13
  ifort/.2017.1.132 (H,L)      java/1.8.0_121                       python/3.5.2(D)
  iimkl/.2017a        (H)      mcr/R2013a                           tbb/2017.2.132
  imkl/2017.1.132              mcr/R2014a
```

compute | calcul
canada | canada

MKL at Core level???

- Many packages use linear algebra but not MPI.
- Example: our `python27-scipy-stack/2017a` module which includes `numpy` but not `mpi4py`).
- 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
module-syntax = Lua
prefix = /cvmfs/soft.computecanada.ca/easybuild
subdir-modules = modules/2017/generic
subdir-software = software/2017/generic
suffix-modules-path =
optarch = GENERIC
module-naming-scheme = SoftCCHierarchicalMNS
recursive-module-unload = 1
repository = GitRepository
repositorypath = %(prefix)s/ebfiles_repo.git
robot-paths = %(prefix)s/easyconfigs:%(prefix)s/ebfiles_repo
hide-deps = icc,ifort
filter-deps =
Bison,CMake,flex,ncurses,libreadline,bzip2,zlib,binutils,M4,Autoconf,Automak
e,libtool,Autotools,GCCcore,Szip,libxml2
filter-env-vars = LD_LIBRARY_PATH # Nix ld uses rpath
minimal-toolchains = 1
add-dummy-to-minimal-toolchains = 1
hide-toolchains = iompi,iomkl
parallel = 8
```

Challenge: there is more than /usr!

```
export NIXPKGS_CONFIG=/cvmfs/soft.computecanada.ca/nix/etc/config.nix
# this config file changes "/nix" to "/cvmfs/soft.computecanada.ca/nix"
export NIXUSER_PROFILE=/cvmfs/soft.computecanada.ca/nix/var/nix/profiles/16.09
#($NIX_PROFILE=$NIXUSER_PROFILE for nixuser, $HOME/.nix-profile for others)
export NIX_REMOTE=daemon
export
PATH=$NIXUSER_PROFILE/bin:$NIXUSER_PROFILE/sbin:/bin:/sbin:/usr/bin:/usr/sbin
export LIBRARY_PATH=$NIXUSER_PROFILE/lib
export CPATH=$NIXUSER_PROFILE/include
export ACLOCAL_PATH=$NIXUSER_PROFILE/share/aclocal
export PKG_CONFIG_PATH=$NIXUSER_PROFILE/lib/pkgconfig
export FONTCONFIG_FILE=$NIXUSER_PROFILE/etc/fonts/fonts.conf
export CMAKE_PREFIX_PATH=$NIXUSER_PROFILE
export
PYTHONPATH=${PYTHONPATH:+$PYTHONPATH:}$NIXUSER_PROFILE/lib/python2.7/site-packa
ges
export SSL_CERT_FILE=/etc/pki/tls/certs/ca-bundle.crt # impure!
```

Catches most searches for EasyBuild builds.
Best to not have any -devel RPMs installed.

Example: $EBROOTICC/bin/icc

```
$ # this binary software was patchelf'ed.
$ echo $EBROOTICC
/cvmfs/soft.computecanada.ca/easybuild/software/2017/gener
ic/Core/icc/2017.1.132
$ patchelf --print-rpath $EBROOTICC/bin/icc
$ORIGIN:/cvmfs/soft.computecanada.ca/nix/var/nix/profiles/
16.09/lib
$ echo $NIXUSER_PROFILE
/cvmfs/soft.computecanada.ca/nix/var/nix/profiles/16.09
$ ls -l $NIXUSER_PROFILE/lib/libm.so.6
lrwxrwxrwx 1 root root 96 Jan  1  1970
/cvmfs/soft.computecanada.ca/nix/var/nix/profiles/16.09/li
b/libm.so.6 ->
/cvmfs/soft.computecanada.ca/nix/store/sbry0lyf8ckk9hq5avg
qh038rwk9sw86-glibc-2.24/lib/libm.so.6
```

compute | calcul
canada | canada

EasyBuild wrapping nix

Idea: Nix can install software in a custom profile, e.g. a "bare" Python:

```
.../nix/var/nix/profiles/python-2.7.13 ->
.../nix/var/nix/profiles/python-2.7.13-6-link ->
.../nix/store/iw112pbd7r8rmczbz5kjbspchi43mbsx-user-environment ->
(symlinks into nix store),
```

using:

```
sudo -u nixuser -i nix-env -iA python27Full python27Packages.virtualenv \
   python27Packages.pip python27Packages.wheel -p \
   .../nix/var/nix/profiles/python-2.7.13
```

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

```
installdir=$EBROOTPYTHON=.../nix/var/nix/profiles/python-2.7.13
```

We do similar things for GCC (but no GCCcore)

compute canada | calcul canada

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).

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, ...