HPC user software environment management with Nix

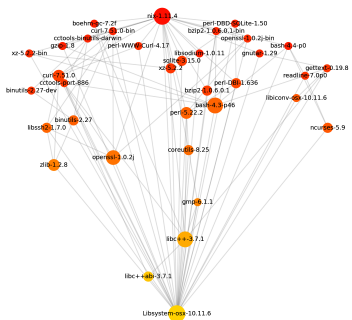
Bruno Bzeznik, Valentin Reis

3rd Easybuild User Group, January 29, 2018

Overview

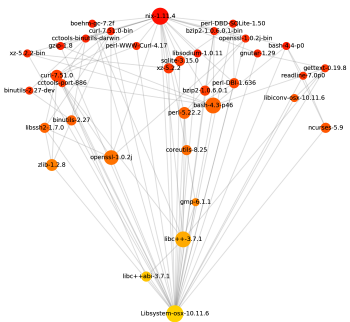
Software Packaging in HPC

Maintenance Reproducibility Portability



Software Packaging in HPC

Maintenance Reproducibility Portability



HPC Specific needs..

- Advanced rights management (user envirenments, community-specific packages, licenses)
- Automated development workflow: custom packages, automated rebuilds, HPC machine vs laptop
- Advanced build option access
- OS independent

Existing solutions

Package-based solutions


Nix

Container-based solutions


Singularity

udocker

Charliecloud

Shifter

The Jungle

	easybuild	Spack	/ GuixHPC
Build Reproducibility	up to system libs	up to system libs	almost binary
User Environments	module-based	module-based	pure, isolated
Runtime env	module/os-based	module/os-based	static deps
Portability	no	no	yes
Packaging Language	Python/Easyconfig/tcl	Python	Nix
Multiple versions	yes	yes	rebuild/grafts
Binary packages	no	yes	yes
Isolated build env.	no	no	yes
External/Legacy Modules	yes	?	no
Who packages	anyone?	no	anyone
Who builds	admin and user?	admin or user	admin and user on head (daemon+store)
Community/Doc	HPC	HPC	large
Custom Packages	?	?	source-based

The Nix Ecosystem. https://nixos.org

Nix Ecosystem

Nix - The Expression Language

Nix - The Nix package manager

Hydra - Nix-based continuous build system

NixOS - The Purely Functional Linux Distribution

NixOps - The NixOS Deployment Tool

Nix - The Expression Language

What?

- Functional, Turing complete language.
- Typed. int, bool, path, string, set, list, lambda.
- Large built-in and standard lib. `stdenv`, `fetchTarball`, `fromJson`, `fromGitHub`, `assert`, `test`..

Nix - The Expression Language

Why?

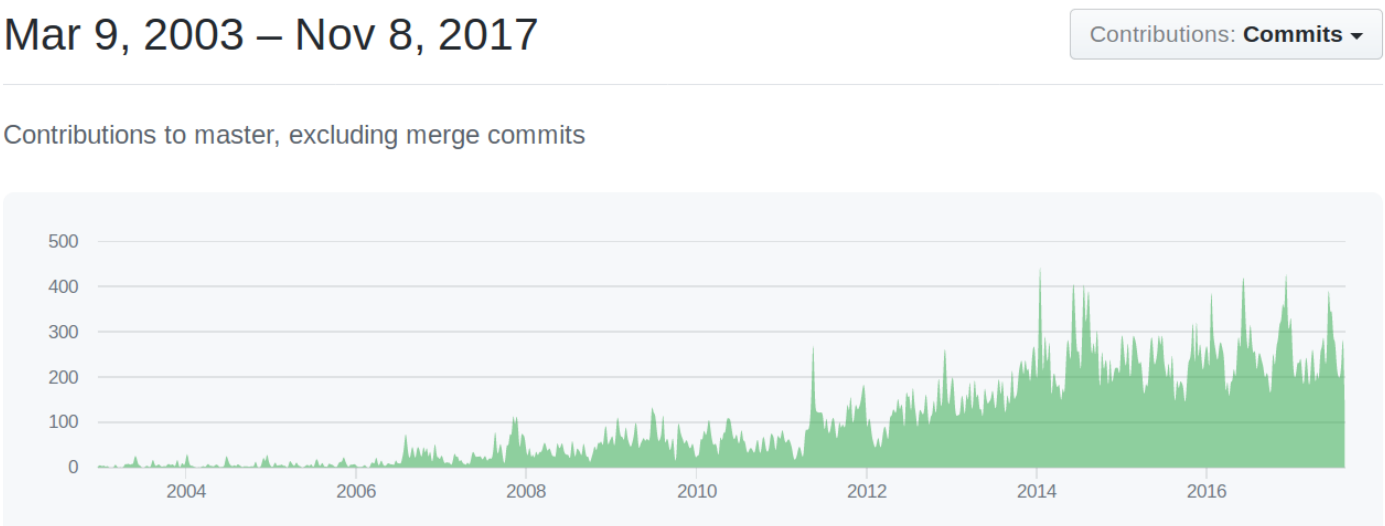
- Packaging is complex.
- Abstraction layers.
- Better reusability, factorization.
- (Readable and Mantainable)

Nix - The Package Manager

- Packages are defined in Nix expressions
- Independent of the system.
- Atomic upgrades and rollbacks
- Several version of the same package on the same system
- Unprivileged package installation
- Provides isolated build & runtime environments
- Reproducible build from source
- Binary Cache
- Garbage collection
- Declarative & Imperative use.

Nixpkgs - https://github.com/NixOS/nixpkgs

- >**10 000** packages
- >**1 400** contributors
- >**110 000** commits



Derivations

```
  { stdenv, fetchurl, cmake, fftw, openmpi
  singlePrec ? true,
  mpiEnabled ? false,
  }:
  stdenv.mkDerivation {
    name = "gromacs-4.6.7";
  src = fetchurl {
    url = "ftp://ftp.gromacs.org/pub/gromacs/gromacs-4.6.7.tar.gz";
  sha256 = "6afb1837e363192043de34b188ca3cf83db6bd189601f2001a1fc5b0b2a214d9";
};
  buildInputs = [cmake fftw]
  ++ (stdenv.lib.optionals mpiEnabled [ openmpi ]);
  cmakeFlags = ''
  ${if singlePrec then "-DGMX_DOUBLE=OFF" else "-DGMX_DOUBLE=ON -DGMX_DEFAULT_SUFFIX=OFF"}
  ${if mpiEnabled then "-DGMX_MPI:BOOL=TRUE
  -DGMX_CPU_ACCELERATION:STRING=SSE4.1
  -DGMX_OPENMP:BOOL=TRUE
  -DGMX_THREAD_MPI:BOOL=FALSE"
  else "-DGMX_MPI:BOOL=FALSE" }
  '';
  meta = ...
}
```

Store

Packages are stored in the nix store: `/nix/store`.
They are identified with a sha-256 hash of the source nix file and its "inputs".
`/nix/store/an9dli66ng2jzvqf13b2i230mm9fq7qk-cdo-1.7.2`
changing a flag alters the inputs produce new packages:
`/nix/store/srf6grrfy9vkc9fsplk8xk292lm8jvz5-cdo-1.7.2`

Profiles

Users have profiles. Profiles :

- are independent.
- are unlimited.
- can be rolled back.

Channels, binary caches

Channels:

- `nixpkgs-unstable`
- `nixos-YY.MM` (NixOS-users)
- `ciment-channel` (The GRICAD channel)

Example: `openmpi` installation

```
$ nix-env -i -A nixpkgs-unstable.openmpi
```

Channels, binary caches

Channels:

- `nixpkgs-unstable`
- `nixos-YY.MM` (NixOS-users)
- `ciment-channel` (The GRICAD channel)

Example: `openmpi` installation

```
$ nix-env -i -A nixpkgs-unstable.openmpi
```

Binary caches

Installing a package:

1. Check if exists in the `binary-cache`
2. Else, building from sources & deps

Attributes

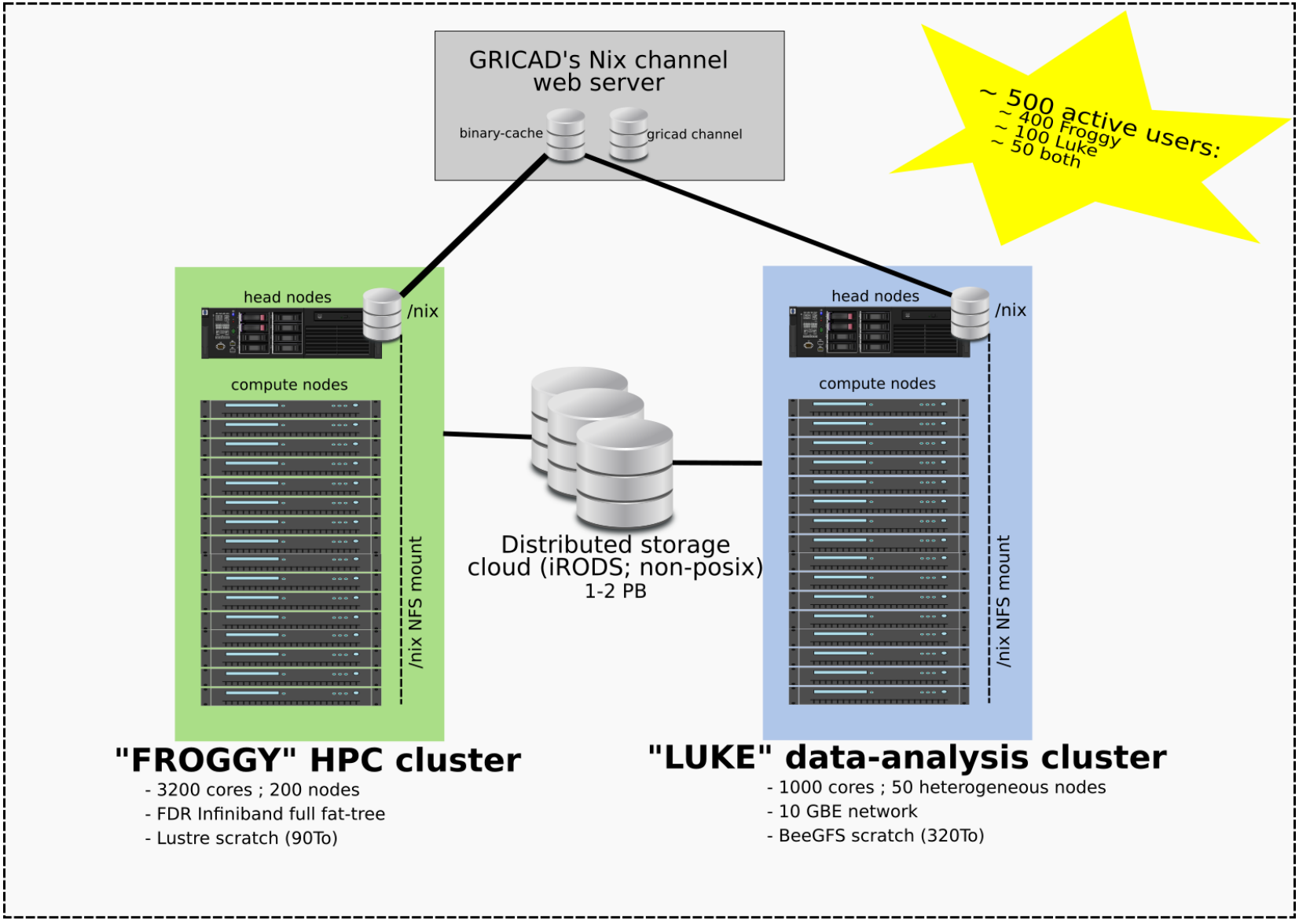
Installation of a `gromacs` application version

```
$ nox gromacs Refreshing cache
1 gromacs-4.6.7 (ciment-channel.gromacs)
Molecular dynamics software package
2 gromacs-4.6.7 (ciment-channel.gromacsDouble)
Molecular dynamics software package
3 gromacs-4.6.7 (ciment-channel.gromacsDoubleMpi)
Molecular dynamics software package
4 gromacs-4.6.7 (ciment-channel.gromacsMpi)
Molecular dynamics software package
```

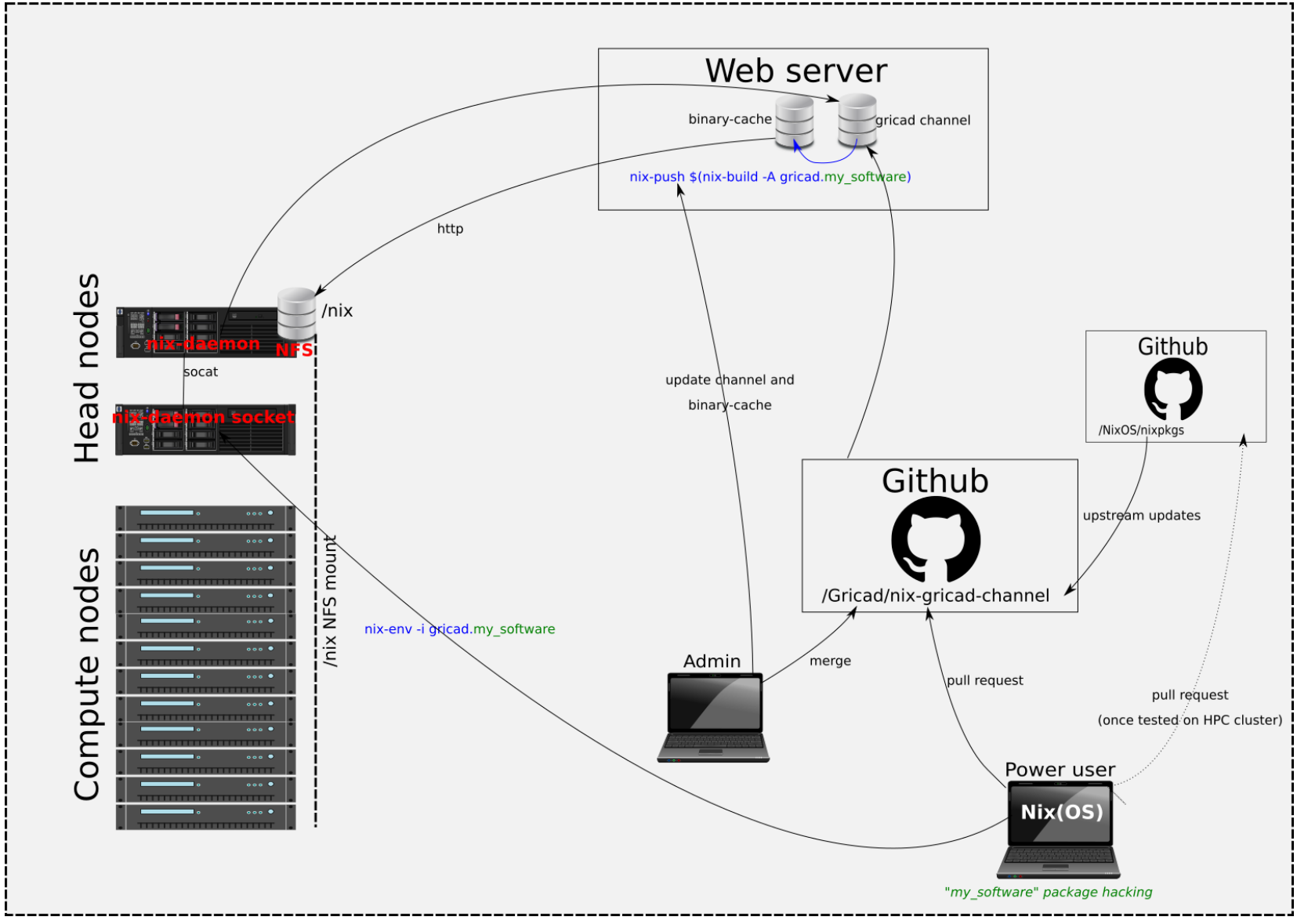
Legend: `attribute` → Multiple concurrent versions of one particular application 

Nix at the GRICAD HPC center

The GRICAD HPC center



The Nix setup



User feedback

12 month experiment. Nix at GRICAD has:

- $>$ 50 users
- $>$ 100gb `/nix/store`
- $>$ 20k derivations
- $>$ 1200 generations

References

Official links

Nix: `https://nixos.org/nix/`
NixOS: `https://nixos.org/`
Nixpkgs: `https://nixos.org/nixpkgs/`

Our documentation

Blog: `https://gricad.github.io/calcul/`
Channel: `https://github.com/Gricad/nix-ciment-channel`

Thanks. Discussion Time..

The Jungle

	easybuild	Spack	/ GuixHPC
Build Reproducibility	up to system libs	up to system libs	almost binary
User Environments	module-based	module-based	pure/nix-shell
Runtime env	module/os-based	module/os-based	static deps
Portability	no	no	yes
Packaging Language	Python/Easyconfig/tcl	Python	Nix
Multiple versions	yes	yes	rebuild/grafts
Binary packages	no	yes	yes
Isolated build env.	no	no	yes
External/Legacy Modules	yes	?	no
Who packages	anyone?	no	anyone
Who builds	admin and user?	admin or user	admin and user on head (daemon+store)
Community/Doc	HPC	HPC	large
Custom Packages	?	?	source-based