ANF UST4HPC
Fréjus

NIX & HPC

GRICAD - Pôle Calcul
Bruno Bzeznik & Laure Tavard

May 17, 2018

Sommaire

Nix comme environnement de calcul à haute performance

- **Part. I] Parlons Nix**
  - Nix - Vue d'ensemble et concepts
  - Nix Advanced - Concepts avancés
  - Les liens utiles
- Part. II] Travaux pratiques [Bruno]

NIX - VUE D'ENSEMBLE ET CONCEPTS

Nix - Vue d'ensemble

The Purely Functional Package Manager

Nix

- Functional package manager
- Fiable & reproductible
- Dispo sur Linux & MAC OS
- Users peuvent créer/installer un paquet sans passer `root`

Nixpkgs

- 6,500 paquets
- Pur: pas de deps en dehors du "Nix store"

NixOS

The Purely Functional Linux Distribution

Nix - Vue d'ensemble
The Purely Functional Package Manager

Nix

- Functional package manager → Pas d'effet de bord
- Fiable & reproductible (Expérimentation, recherche)
- Dispo sur Linux & MAC OS → Un même paquet pour Tier0/1/2/3
- Users peuvent créer/installer un paquet sans passer `root` → Facilité de la personnalisation d'env.

Nixpkgs

- 6,500 paquets (Communauté)
- Pur: pas de deps en dehors du "Nix store"

NixOS

The Purely Functional Linux Distribution → Pour aller plus loin

Stockage des paquets

Unique répertoire `/nix/store`

→ Pas de pollution de l'arborescence du système

Stockage des paquets

Unique répertoire `/nix/store`

→ Pas de pollution de l'arborescence du système

Identification des paquets via hash

Construction du paquet → unique sous-répertoire

`/nix/store/an9dli66ng2jzvqf13b2i230mm9fq7qk-cdo-1.7.2`

$$Hash = mix(sources + deps + flags, ...)$$

Compilation + nouvelle option → nouveau hash
`/nix/store/srf6grrfy9vkc9fsplk8xk292lm8jvz5-cdo-1.7.2`

→ Conservation de l'arbre des dépendances,
→ Unicité du paquet

EN RÉSUMÉ

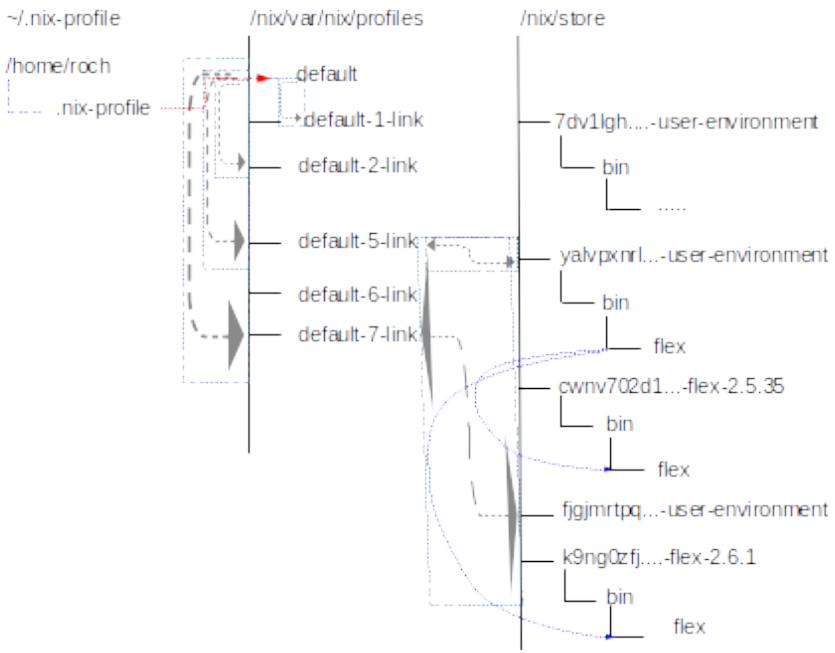
- Installation d'un paquet = Création de liens dans le profil de l'utilisateur stocké dans son home
- `PATH=~/.nix-profile/bin:$PATH`

```
~$ nix-env -i hello

~$ readlink ‘which hello‘
/nix/store/3dlqv87hrrfjynj0brbn4h71g4g4g89z-hello-2.10/bin/hello

~$ ldd /nix/store/3dlqv87hrrfjynj0brbn4h71g4g4g89z-hello-2.10/bin/hello
        linux-vdso.so.1 (0x00007ffe5d1b6000)
        libc.so.6 =>
/nix/store/q3wx1gab2ysnk5nyvyyg56ana2v4r2ar-glibc-2.24/lib/libc.so.6
(0x00007f3d90bc9000)
/nix/store/q3wx1gab2ysnk5nyvyyg56ana2v4r2ar-glibc-2.24/lib/ld-linux-x86-64.so.2
(0x00007f3d90f67000)
```

NIX ADVANCED - CONCEPTS AVANCÉS

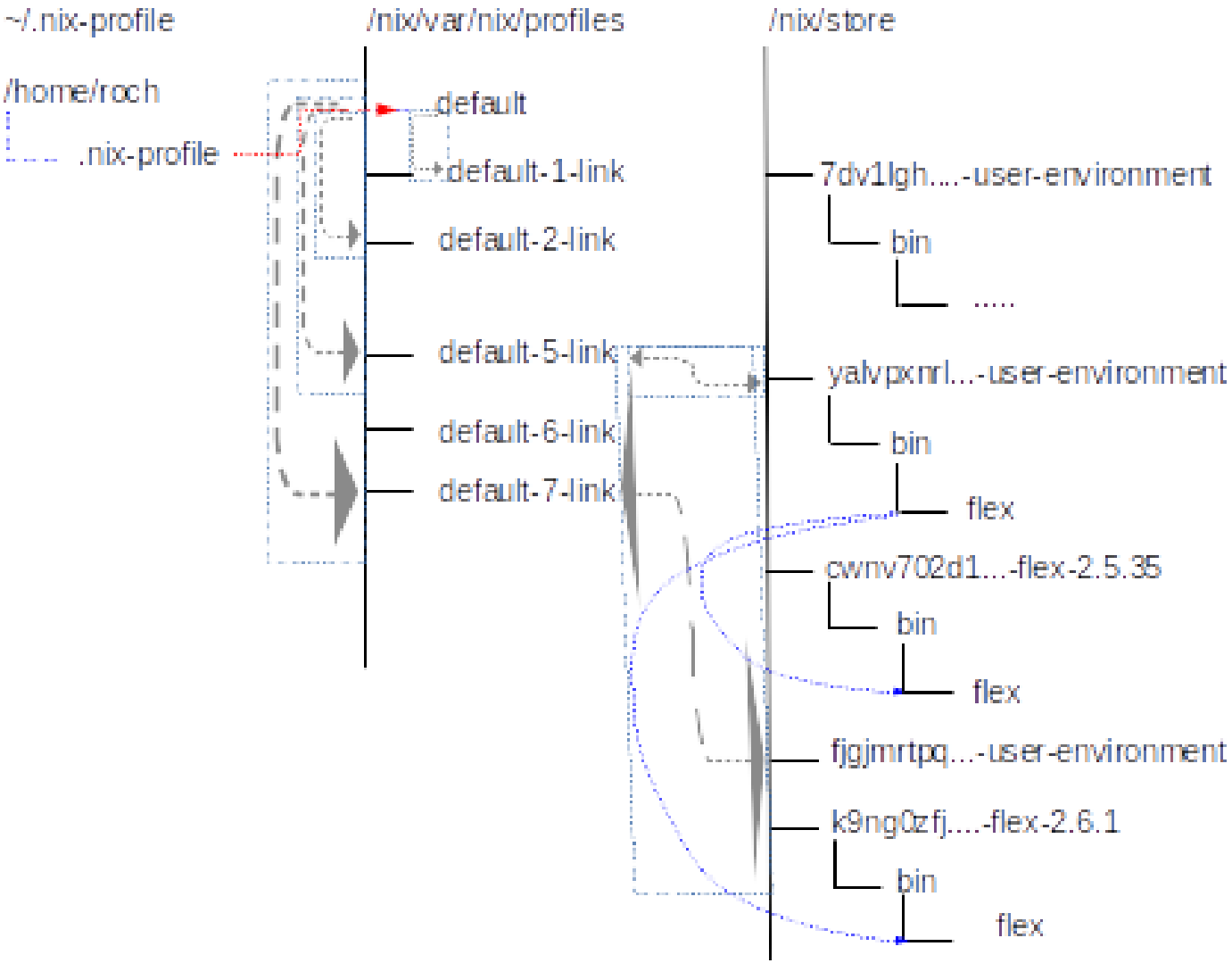
Profile

Environnement défini par un **ensemble de liens symboliques** dans le répertoire personnel de l'utilisateur, vers les paquets utilisés dans le store.



- **Nombre illimité et cohabitation** de profiles
- **Suppression, mise à jour du profile**
  => aucun effet pour le reste
- **Rollbacks** et historique des générations

Les concepts
Profile



Nix - Les concepts
Channels et binary-cache

`channel`

tgz du snapshot de Nixpkgs

Ex. de channels:

- `nixpkgs-unstable`
- `nixos-YY.MM` (NixOS-users)
- `ciment-channel` (CIMENT users :))

Exemple: Installation du paquet `openmpi`

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

Nix - Les concepts
Channels et binary-cache

`channel`

tgz du snapshot de Nixpkgs

Ex. de channels:

- `nixpkgs-unstable`
- `nixos-YY.MM` (NixOS-users)
- `ciment-channel` (CIMENT users :))

Exemple: Installation du paquet `openmpi`

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

`binary cache`

Pour installation d'un paquet:

1. Parcours du `binary-cache`
2. Si besoin, construction depuis les sources et des deps

Pour faire court

- Langage fonctionnel (*Dérivé d'Haskell*) → pas d'effet de bord
- Pas d'affectation de variable → pur
- Argument de fonction utilisé **au besoin** → évaluation paresseuse
- Types: entiers, opérateurs, caractères, listes, ensembles, ...

Essayer le langage avec la commande `nix-repl`:

```
$ nix-env -i nix-repl
$ nix-repl
nix-repl> 1+3
4
nix-repl> builtins.div 6 3
2
```

source: http://lethalman.blogspot.fr/2014/07/nix-pill-4-basics-of-language.html

Installation d'une variante de `gromacs`

```
$ 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
```

Légende: `attribut`

Nix - Les concepts
Les dérivations

Dérivation

fonction d'action de construction du langage qui prend en paramètre des arguments de la construction

```
{ stdenv, fetchurl, cmake,
  singlePrec ? true,
  mpiEnabled ? false,
  fftw,
  openmpi
}:
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 = with stdenv.lib; {
    homepage    = "http://www.gromacs.org";
    license     = licenses.gpl2;
    description = "Molecular dynamics software package";
    platforms = platforms.unix;
  };
}
```

REFS & BIBLIO

Les liens utiles

Biblio officielle

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

Références GRICAD

Blog: `https://gricad.github.io/calcul/`
Channel: `https://github.com/Gricad/nix-ciment-channel`
PosterJDEV2017: `Contributing to the Nix HPC packages collection`

Autres

Blog: `http://lethalman.blogspot.fr/2014/07/nix-pill-1-why-you-should-give-it-try.html`

TRAVAUX PRATIQUES

Déroulement du TP

Pour ce TP, nous allons:

- Utiliser une machine de calcul réelle: **Luke**, à Grenoble
- Apprendre à utiliser NIX en suivant le tutoriel des JDEV 2017
- Réaliser 2 exemples de jobs de calcul (R et mpi-ring.c)

Connexion à Luke

- **ssh -Y access-ciment.imag.fr -l stagiaireXX**
- **ssh -Y luke**
- optionnellement, vous pouvez mettre en place une config de proxy-commands:

```
Host *.ciment
 User yourlogin
 ProxyCommand ssh -q yourlogin@access-ciment.imag.fr "nc -w 60 `basename %h .ciment` %p"
```

Soumission de jobs sur Luke

- Soumission d'un job interactif sur 2 noeuds, avec 1 core par noeud:

```
oarsub --project ust4hpc -l /nodes=2/core=1 -I
```

- Exemple de lancement d'un job MPI avec OAR:

```
mpirun -np `cat $OAR_FILE_NODES|wc -l` \
       --machinefile $OAR_NODE_FILE \
       -mca plm_rsh_agent "oarsh" \
       ./application
```

Chargement de l'environnement NIX sur Luke

- Pour utiliser NIX, il faut sourcer un fichier d'environnement

```
source /applis/site/nix.sh
```

Tutoriel NIX

Suivre le tutoriel des JDEV 2017 en commençant directement à la section **NIX basics**:

```
https:
//gricad.github.io/calcul/nix/tuto/2017/07/04/nix-tutorial.html
```

- Lisez le début, il vous permettera d'installer Nix sur votre propre machine, mais vous n'avez pas à le faire puisque Nix est déja installé, en mode multi-users sur la machine Luke que vous allez utiliser.
- Vous aurez certainement à adapter quelques petites choses puisque le tutoriel est basé sur une installation locale. A vous de découvrir!
- En fonction du temps disponible, vous pouvez vous arreter juste avant la partie **How to add a package to nixpkgs**

Exemple d'utilisation avec R

A l'aide de cette documentation:

```
http://calcul-scientifique-ljk.imag.fr/serveurs-de-calcul/
utilisation/article/packages-r-avec-nix
```

- Installez R et les modules **raster** et **dismo**
- Nous allons maintenant essayer de refaire ce qui est fait sur cette page `https://www.r-bloggers.com/simulating-animal-movements-and-habitat-use/`. Le but n'étant pas de tout refaire/comprendre, mais simplement de relancer le code dans un job.
- Le code R à éxécuter dans un job: `ftp://ngsisem.mbb.univ-montp2.fr/mbbteam/datadir/mangouste_vs_cerf.R` avec la commande R CMD BATCH `-slave mangouste_vs_cerf.R`
- Vous aurez probablement à installer un soft de visualisation PDF (ex: `evince`) afin d'afficher le résultat qui se trouve dans le fichier `density_distrib.pdf`

Exemple d'utilisation avec un code MPI

Créez un nouveau profile NIX et installez y Openmpi 3.0.0 afin de faire tourner le code suivant sur 4 cores répartis sur 2 noeuds de calcul:

mpi-ping.c

`http://gitlab.mbb.univ-montp2.fr/lxdsingu/atelier-lxdsingu/raw/master/mpi-ping.c`

TIP: lisez la partie **Appendix** du tutoriel NIX...

Exemple d'utilisation avec un code MPI
Solution 1: nix-shell

Sur la frontale, on compile dans un **nix-shell**:

```
nix-shell -p openmpi -p gcc
mpicc -o mpi-ping mpi-ping.c
exit
nix-env -i openmpi # pour le runtime
```

Lancement du job en interactif:

```
oarsub -I -l /nodes=2/core=2 --project ust4hpc
source /applis/site/nix.sh
mpirun -np `cat $OAR_FILE_NODES|wc -l` \
       --machinefile $OAR_NODE_FILE \
       -mca plm_rsh_agent "oarsh" \
       --prefix $HOME/.nix-profile \
       ./mpi-ping 1000000
```

Exemple d'utilisation avec un code MPI
Solution 2: Créer un paquet local 1/2

Créer un **builder**:

```
cat > mpi-ping_builder.sh <<EOF
export PATH="\$coreutils/bin:\$gcc/bin:\$openmpi/bin"
mkdir \$out
mpicc -o \$out/mpi-ping \$src
EOF
```

Créer une **dérivation**:

```
cat > mpi-ping.nix <<EOF
with (import <nixpkgs> {});
derivation {
  name = "mpi-ping";
  builder = "\${bash}/bin/bash";
  args = [ ./mpi-ping_builder.sh ];
  inherit coreutils gcc openmpi;
  src = ./mpi-ping.c;
  system = builtins.currentSystem;
}
EOF
```

Exemple d'utilisation avec un code MPI
Solution 2: Créer un paquet local 2/2

Compiler:

```
nix-build ./mpi-ping.nix
```

Lancement du job en interactif:

```
oarsub -I -l /nodes=2/core=2 --project ust4hpc
source /applis/site/nix.sh
mpirun -np ‘cat $OAR_FILE_NODES|wc -l‘ \
       --machinefile $OAR_NODE_FILE \
       -mca plm_rsh_agent "oarsh" \
       --prefix $HOME/.nix-profile \
       ./result/mpi-ping 1000000
```

Exemple d'utilisation avec un code MPI
Solution 3: Créer un paquet officiel

Cela pourrait ressembler à:
`https://github.com/Gricad/nix-ciment-channel/tree/master/ciment/mpi-ping`