GRICAD

Présentation et retour d'expérience sur les outils d'installation de codes

Pierre-Antoine Bouttier, CNRS
Violaine Louvet, CNRS

Journée des utilisateurs, 27 novembre 2018

Introduction

Un petit rappel

- GRICAD propose (entre autres) **plusieurs** machines de calculs aux caractéristiques différentes...
- ...à destination de **l'ensemble** de la communauté **recherche** grenobloise

Diversité de besoins et de ressources : contraintes en terme d'environnements logiciels

Hétérogénéités à tous les étages

Hétérogénéité des machines

- Caractéristiques différentes (parfois au sein d'un même cluster)
- OS différents
- Objectifs différents (HPC vs analyse de données)

Hétérogénéité des besoins de calculs

- Cas d'utilisations (e.g. batch, MPI, tests, portabilité)
- Différentes attentes : performances, tests, reproductibilité, portabilité, etc.
- Différentes expertises en calcul scientifique/info

La problématique de l'environnement logiciel

Comment gérer l'environnement logiciel de façon à répondre à ces hétérogénéités ?

L'outil historique : module

Quelques remarques sur l'outil :

- Largement utilisé dans les centres de calcul (Tier-X)
- Lourd travail côté adminsys...
- ...et utilisateur.
- Pas de portabilité et de reproductibilité garantie (et la grille, dans tout ça ?)
- etc.

Il doit exister des outils correspondants mieux à nos besoins.

Les gestionnaires de paquets et les solutions de conteneurisation

Gestionnaires de paquets VS conteneurs
En théorie

Gestionnaires de paquets

- Portabilité et reproductibilité
- Travail utilisateur simplifié
- Participation à une communauté (paquets existants, docs, aides)

Conteneurs

- Portabilité et reproducibilité
- Isolation forte de l'applicatif
- Large périmètre d'utilisations (infra et applicatif)

Les gestionnaires de paquets en pratique

Easybuild, Spack :

- Orientés utilisateurs
- Écrits en Python
- Génération automatique de modules
- Dépendant du système (compilation)
- Communauté calcul et HPC

Nix/GUIX :

- Orienté adminsys
- Écrits en langage fonctionnel
- Self-contained (glibc embarquées)
- Très grande reproductibilité
- Partage optimisé des ressources (cache)

Nix, la solution actuelle de GriCAD - Focus à suivre

Les solutions de conteneurisation en pratique

Docker :

- Le plus gros !
- Largement utilisé pour des applications diverses
- Nécessite le root : incompatible avec le HPC.

Les alternatives HPC :

- Singularity, Shifter, Charliecloud, udocker, etc.
- Pas de droits root requis pour exécuter un conteneur
- Peuvent exécuter des images docker
- Développés pour le HPC
- Différences mineures côté utilisateur

Charliecloud et singularity disponibles sur Luke et Dahu - Focus à suivre

Survol de Nix

Nix en tant que gestionnaire de paquet
Principales caractéristiques

- Purement fonctionnel
- Fiable et reproductible (expérimentation, recherche et production)
- Linux et OS X (du desktop au Tier-0 pour un même paquet)
- Création/installation sans les privilèges root
- Pas de dépendances externes
- Environ 6500 paquets existants ; possibilité de créer son propre dépôt

Côté utilisateur

Source de l'environnement Nix

```
[~]
bouttier@luke $ source /applis/site/nix.sh
```

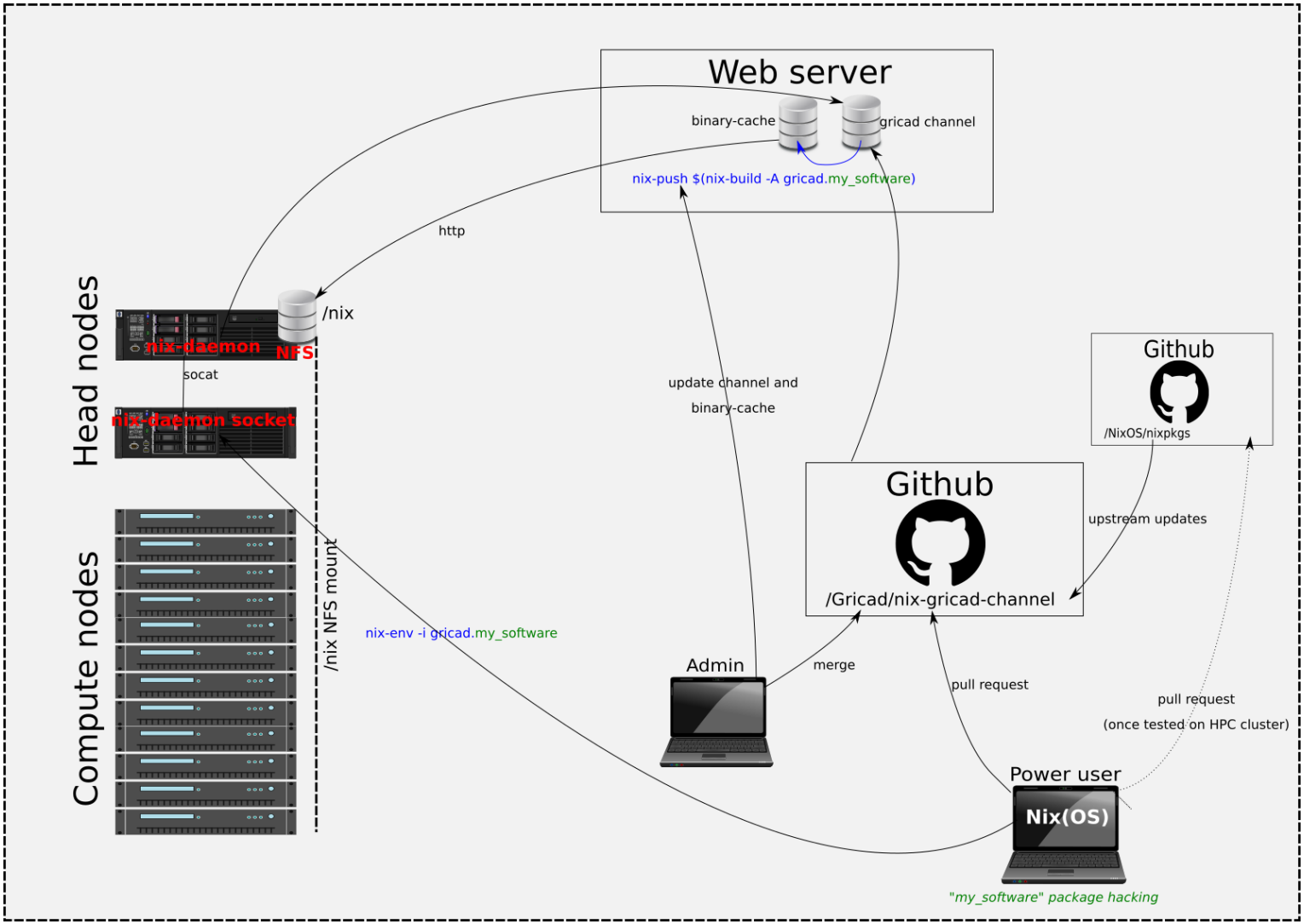
Rechercher un paquet

```
[~]
bouttier@luke $ nix search hdf5
warning: using cached results; pass '-u' to update the cache
* nixpkgs.h5utils (h5utils-1.13.1)
  A set of utilities for visualization and conversion of scientific data in the free, portable HDF5 format

* nixpkgs.hdf5 (hdf5)
  Data model, library, and file format for storing and managing data

* nixpkgs.hdf5-cpp (hdf5-cpp)
  Data model, library, and file format for storing and managing data

* nixpkgs.hdf5-fortran (hdf5-fortran)
  Data model, library, and file format for storing and managing data

* nixpkgs.hdf5-mpi (hdf5-mpi)
  Data model, library, and file format for storing and managing data
```

Installer un paquet

```
[~]
bouttier@luke $ nix-env -iA ciment-channel.hdf5
installing 'hdf5-1.10.3'
building '/nix/store/06jdfvcbqmvw8vbq83krwk8159b1bhvr-user-environment.drv'...
created 308 symlinks in user environment
```

Côté Admin


Web server
binary-cache
gricad channel
nix-push $(nix-build -A gricad.my_software)
http
Head nodes
/nix
nix-daemon
NFS
socat
nix-daemon socket
Compute nodes
/nix NFS mount
update channel and
binary-cache
Github
/NixOS/nixpkgs
Github
/Gricad/nix-gricad-channel
upstream updates
nix-env -i gricad.my_software
Admin
merge
pull request
pull request
(once tested on HPC cluster)
Power user
Nix(OS)
"my_software" package hacking

Avantages et inconvénients

Ce qui nous ravit

- Utilisateur peut créer/installer n'importe quel paquet
- Fiabilité, reproductibilité, portabilité
- Participation à une communauté

Ce qui nous chagrine

- Courbe d'apprentissage ardue pour créer un paquet...
- ...et peu de paquets calcul scientifique et HPC dispos.

Survol de CharlieCloud (et de Singularity)

Charliecloud ?

Charliecloud

- Charliecloud est un système de container léger dédié au HPC
- Développé à Los Alamos depuis 2014
- Sous licence Apache License, Version 2.0

- Repository github : `https://github.com/hpc/charliecloud`
- Documentation : `https://hpc.github.io/charliecloud/`
- Cette présentation s'inspire fortement de la présentation de Michael Jennings à la conférence Swiss HPC 2018 (`https://www.youtube.com/watch?time_continue=2&v=ESsZgcaP-ZQ`)

Comment ça marche ?

- Charliecloud utilise les linux user namespaces pour exécuter des containers sans privilège particulier, sans démon.
- Les problématiques de sécurité sont reportées sur le noyau linux
- Les images des containers peuvent être construite avec docker ou tout autre système qui génère une arborescence linux standard.
- C'est un système de container léger : généralement ne fournit que le support d'excution, et s'appuie sur une arborescence de répertoire existante.
- Charliecloud peut donc exécuter des containers Docker.
- Charliecloud s'appuie sur Docker pour la création de container.
- Charliecloud ne représente que 1 000 lignes de code contre 19 000 pour Shifter, 15 000 pour Singularity et 160 000 pour Docker !

Construction de l'image

- On peut partir d'une image Docker déjà réalisée.
- On peut construire une image :
  - Cela suppose d'avoir installé charliecloud + docker-ce sur une machine sur laquelle on a les droits root
  - Avec un noyau linux raisonnablement récent
  - On écrit un fichier Dockerfile standard
  - On construit l'image avec les commandes charliecloud:
    - Construction :

      ```
      ch-build -t nom_img ./
      ```

    - Compression pour transfert :

      ```
      ch-docker2tar nom_img ./.
      ```

    - On obtient un fichier tar.gz qu'on peut copier sur les machines de calcul.

Installation et utilisation du container

Installation du container

```
ch-tar2dir ../nom_img.tar.gz ./.
```

Pour des questions de performances, le mieux est de le faire dans un espace de stockage local.

Exécution du container

```
ch-run -b /home/:/home -w ./nom_img -- nom_exe
```

Il suffit d'intégrer cette ligne dans un script de soumission de job.

Cas d'un code parallèle

Exécution du container, 1 noeud, plusieurs processus

Deux possibilités :

- L'hôte gère les processus MPI : 1 container par processus MPI

  ```
  mpirun -np 4 ch-run -b /home/:/home -w ./nom_img -- nom_exe
  ```

- Les processus MPI sont gérés par le container

  ```
  ch-run -b /home/:/home -w ./nom_img -- mpirun -np 4 nom_exe
  ```

  Ca permet d'être complètement indépendant de l'hôte, mais cela ne s'utilise qu'en mono-noeud.

Exécution du container, plusieurs noeuds

- Beaucoup moins évident : il faut une cohérence dans les versions mpi de l'hôte et du conteneur.
- Des tests sont en cours pour résoudre les blocages

Singularity ?

- Sans doute le plus utilisé dans le monde HPC
- D'utilisation très semblable à charliecloud, mais de conception différente : en particulier utilisation de setuid.
- Propose un système de build alors que charliecloud s'appuie complètement sur Docker
- Mêmes limites que charliecloud en ce qui concerne l'exécution en distribué
- Son développeur a fondé une start'up (Syslabs) pour le développement, le licensing et le support de Singularity Pro (version entreprise de Singularity) ...
- Site web : `https://www.sylabs.io/singularity/`

Conclusions et perspectives

Conclusions

- Charliecloud est un système de container prometteur : léger, installable en espace utilisateur, permettant de faire tourner des images docker.
- Singularity est un système de container déjà bien installé. Incertitude du fait de son intégration à Syslabs.
- Les deux sont installés sur Luke et Dahu.
- N'hésitez pas à les utiliser et à nous faire des retours !!

Perspectives

- Comprendre le fonctionnement en multi-noeuds.
- Tester sur du GPU, en particulier les images Docker fournies par Nvidia en Deep Learning.
- Faire une documentation, partager des images, faire connaître ces outils.

Conclusion générale/Discussion

GRICAD
GRENOBLE ALPES RECHERCHE
INFRASTRUCTURE DE
CALCUL INTENSIF
ET DE DONNÉES

cnrs
UNIVERSITÉ Grenoble Alpes
Grenoble INP
Informatiques mathématiques Inria

Etats des lieux et questionnements

Nix comme remplaçant à module

- Des avantages indéniables (gestionnaire de paquets) côté utilisateurs et côté admin
- Homogénéité de l'environnement logiciel sur les clusters
- Difficultés de répondre à la demande dans la création de nouveaux paquets
- Spack en espace utilisateur ? (plus forte communauté HPC, plus accessible)
- Lien avec les autres centres de calcul

Etats des lieux et questionnements

Conteneurs

- Singularity et CharlieCloud dispo sur Luke et Dahu (kernel trop ancien côté Froggy)
- Expérimentation en cours de notre côté
- Documentation et accompagnement à suivre
- Lien avec l'intégration continue proposée par la forge GitLab GriCAD

Merci de votre attention.
Questions ?