Singularity, un conteneur pour le HPC

mésocentre de calcul de franche-comté

Cédric Clerget

Journée informatique scientifique - Besançon

2 décembre 2016

Gestion logiciel sur un cluster

Le quotidien :

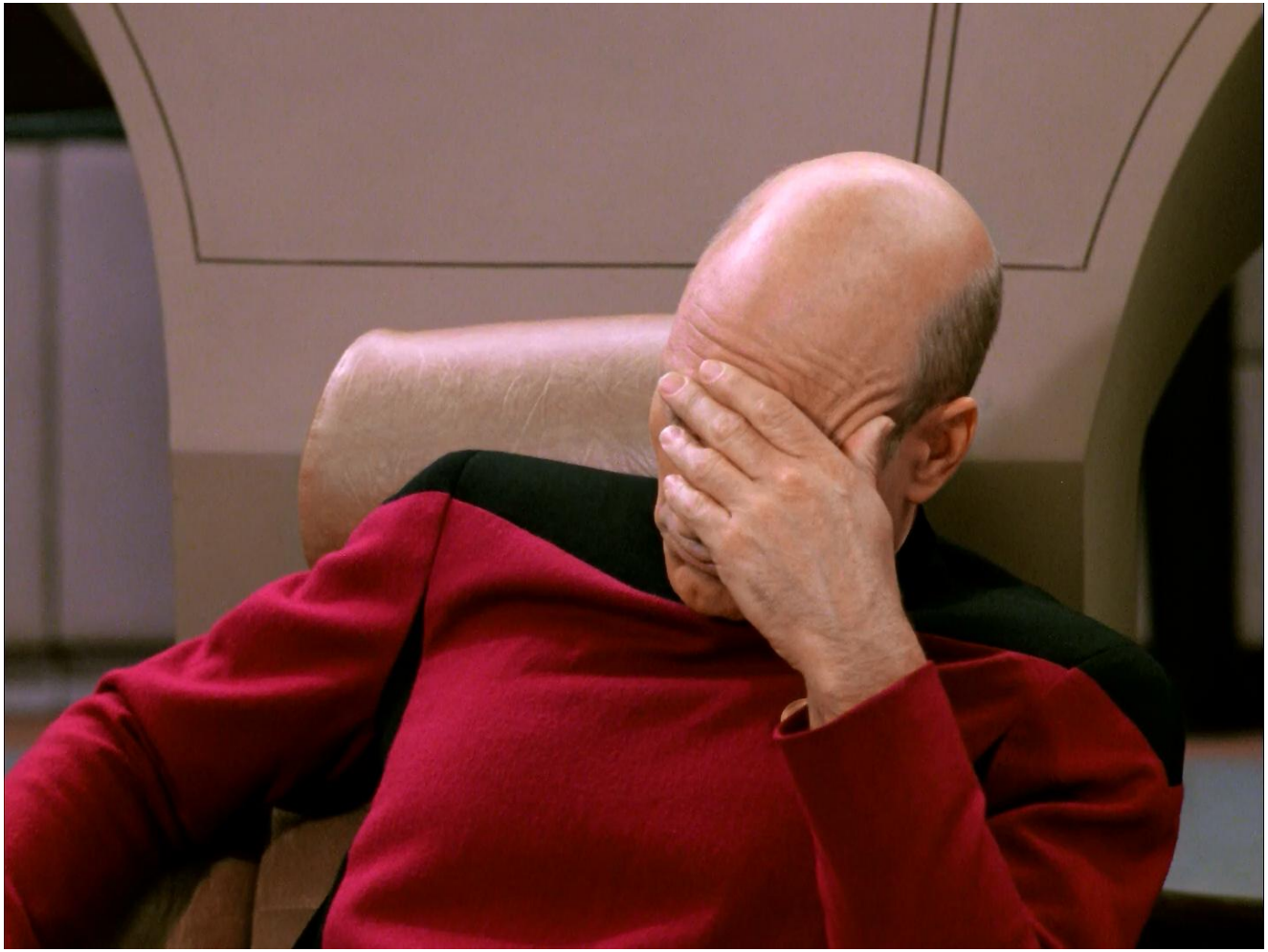
- Compilation et installation
- Mise à jour des logiciels
- Utilisation de modules pour faciliter leur utilisation

Les problématiques rencontrées :

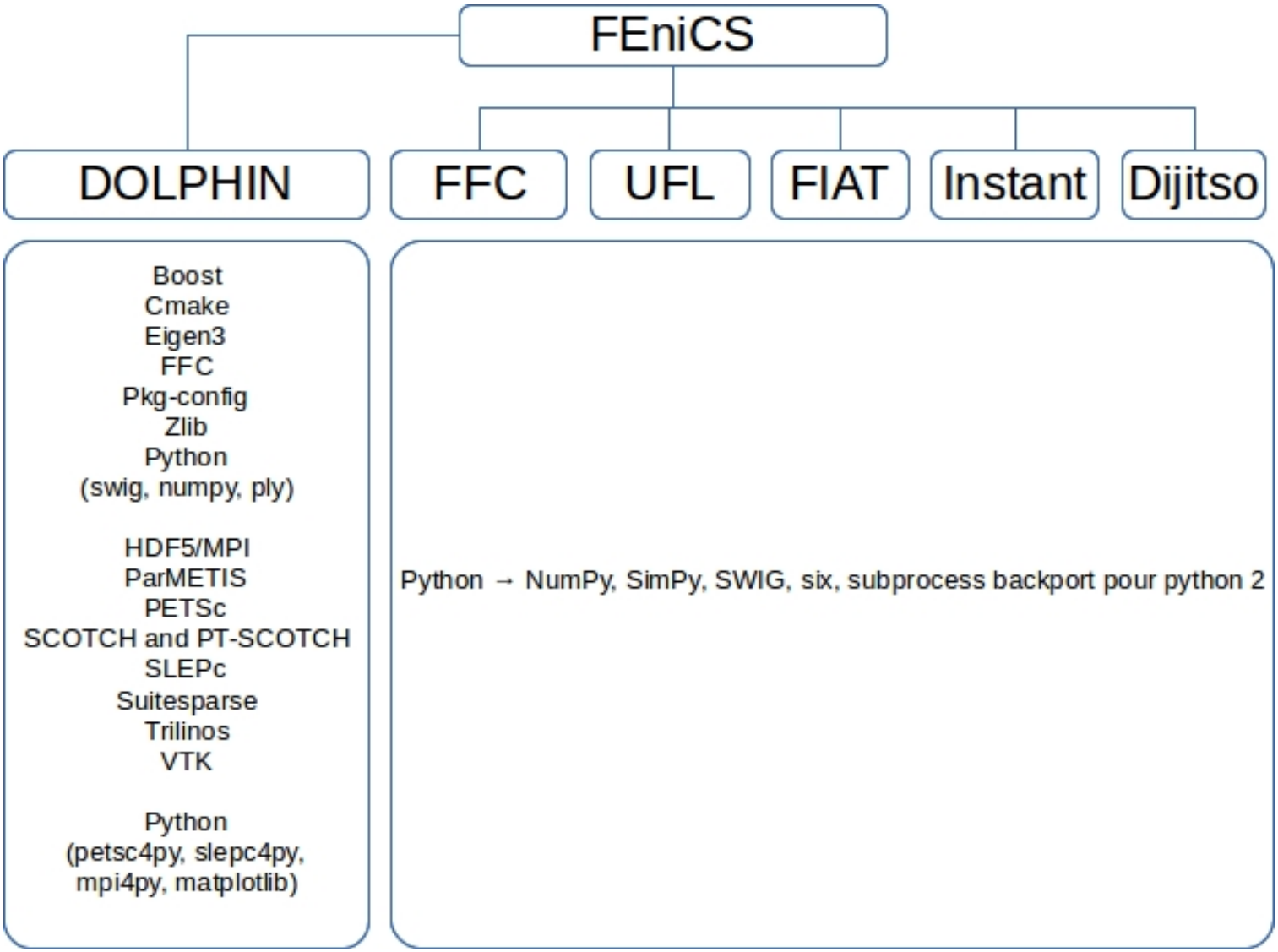
- Compilations complexes avec beaucoup de **dépendances**
- Compilations **non reproductibles**
- Vieilles distributions **VS** nouveaux logiciels

Compilations et dépendances

Dans la famille complexe, je demande FEniCS !



FEniCS et ses dépendances



Non reproductibilité

Solution manuelle :

- Ecrire un script de compilation pour chaque logiciel

Inconvénients

- Aucune garantie entre les versions de logiciels
- Nécessite une écriture pour chaque version

Solutions automatiques :

- Easybuild, automatisation de compilation avec génération de modules (orienté HPC)
- Nix / Guix, gestionnaires de package multi-user et multi-version complétement indépendant du système hôte

Inconvénients

- Ne fonctionne pas forcément avec tous les logiciels

Les vieilles distributions

GLIBC : l'origine du problème

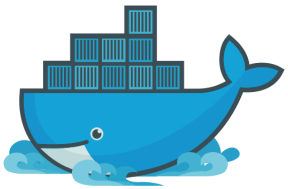
Impact

- Exécution impossible pour les logiciels propriétaires
- Utilisation de hack avec LD_LIBRARY_PATH et une glibc récente
- Des logiciels open source requiert une version récente (tensorflow)

Solutions possibles :

- Utiliser Nix / Guix (apprentissage pour les admins)
- Mettre à jour le système (maintenance lourde)
- Virtualisation (maintenance lourde, impacte les performances)
- Conteneur

Les conteneurs d'applications



Docker	https://www.docker.com/
Shifter	https://github.com/NERSC/shifter
Singularity	http://singularity.lbl.gov/

Docker

Avantages

- Communauté
- Dépôt central de plus de 100 000 images

Inconvénients

- Accès root dans le conteneur
- Forte isolation, pas d'accès à l'Infiniband et aux partages réseaux
- Pas d'accès à l'affichage donc pas d'accès GPU

Inadapté pour le HPC

Shifter

Avantages

- Accès au dépôt central de docker via un serveur d'image local
- Accès Infiniband, partages réseaux
- Pas d'accès root
- Provisioning et lancement d'image depuis SGE

Inconvénients

- Nécessite une intégration avec SGE (epilog, prolog)
- Pour le moment pas de support GPU (shifter-gpu)

Dédié HPC, sans support GPU

Singularity

Avantages

- Accès au dépôt central de docker pour la création d'image
- Accès Infiniband, partages réseaux et GPU
- Pas d'accès root
- Utilisable avec les modules et SGE comme un logiciel classique

Inconvénients

- Quelques bugs pour certaines images Docker, facilement corrigeable
- Intégration MPI pas totalement mature, utilisation de OpenMPI 2

Testé et approuvé

Singularity

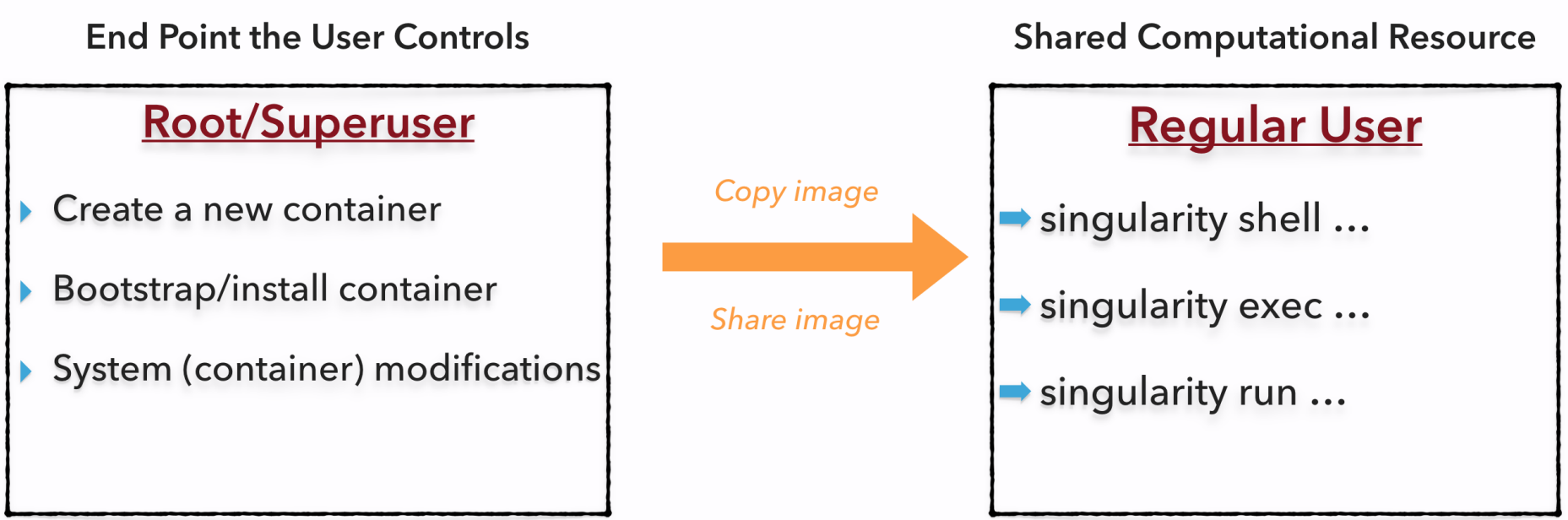
Développer au laboratoire Lawrence Berkeley par le créateur de CentOS pour remplir 3 critères :

- **Portabilité** entre environnements Linux
- **Reproductibilité** des résultats
- **Mobilité** entre cluster

Fonctionalités :

- Encapsulation de l'environnement utilisateur
- Conteneur à base d'image
- Droits utilisateurs identiques dans et hors conteneur
- Montage automatique du répertoire utlisateur et des partages réseaux

Aperçu


End Point the User Controls
Root/Superuser
Create a new container
Bootstrap/install container
System (container) modifications
Copy image
Share image
Shared Computational Resource
Regular User
singularity shell ...
singularity exec ...
singularity run ...

Création d'un conteneur Ubuntu

Création de l'image :

```
$ singularity create -s 4096 ubuntu.img
```

Fichier de définition ubuntu.def :

```
BootStrap: docker
From: ubuntu:16.04
IncludeCmd: yes
```

Bootstrap de l'image :

```
$ singularity bootstrap ubuntu.img ubuntu.def
```

Attention

Commandes à effectuer en tant que root

Bootstrap : le fichier de définition

Bootstrap :

- yum
- debootstrap
- docker

Exécution de commandes :

- **%setup**, hors conteneur durant le bootstrap
- **%post**, dans le conteneur durant le bootstrap
- **%runscript**, à chaque "run" du conteneur
- **%test**, à la fin de la phase de bootstrap

```
BootStrap: docker
From: ubuntu:latest

%post
  apt-get update && apt-get -y install vim
  mkdir /Work /Home
```

Lancement du conteneur

- shell

```
$ singularity shell ubuntu.img
Singularity: Invoking an interactive shell within container...

Singularity.ubuntu.img>
```

- exec

```
$ singularity exec ubuntu.img python
Python 2.7.12 (default, Jul  1 2016, 15:12:24)
>>>
```

- run

```
$ singularity run ubuntu.img
This is what happens when you run the container...
$
```

Possibilité de lancer l'image comme un exécutable :

```
$ ./ubuntu.img
This is what happens when you run the container...
```

FEniCS avec Singularity


```
$ singularity create -s 4096 fenics.img
```

```
BootStrap: docker
From: fenicsproject/stable:latest
IncludeCmd: yes

%post
  mkdir /Work /Home
```

```
$ singularity bootstrap fenics.img fenics.def
```

Quelques minutes plus tard pour l'utilisateur :

```
$ singularity exec fenics.img python /usr/local/share/dolfin/demo/
    documented/poisson/python/demo_poisson.py
```

Singularity for the win

En résumé, Singularity c'est :

- Simple d'utilisation
- La possibilité d'accèder aux images Docker
- Pas de restriction d'accès aux ressources matérielles

Tests de performances

- Linpack -> aucune différence
- Entrée/Sortie -> aucune différence

Problèmes rencontrés :

- Lancement intéractif avec SSH X11 Forwarding
- Variables d'environnements (ex : PATH non standard)

A venir :

- Dépôt de définition
- Conversion de Dockerfile en fichier de définition
- Démarrer un conteneur pour le stopper ultérieurement (start/stop)

Questions ?