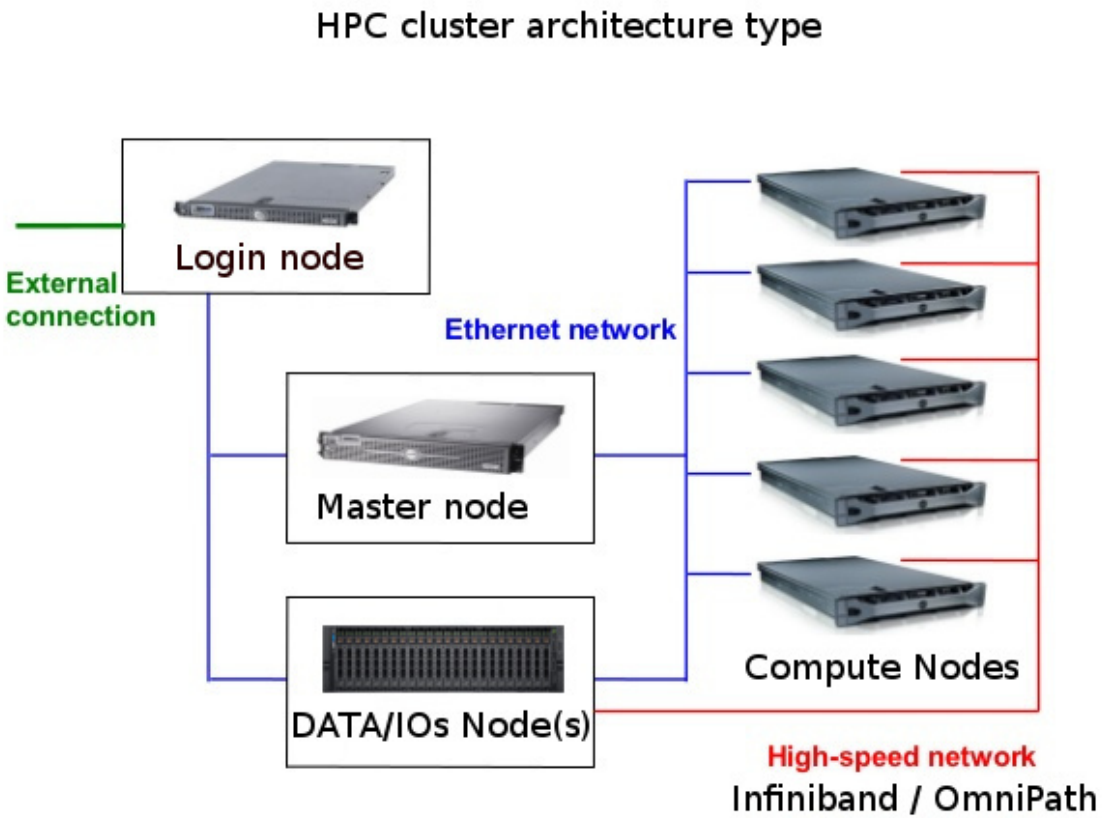
Les conteneurs sur HPC
Docker et Singularity

Thursday
15e March, 2018

Rémy Dernat

CNRS
Institut des Sciences de l'Évolution Montpellier
Plateforme MBB - LabEx CeMEB

Architecture HPC



Les problématiques HPC

- Orchestration avec un Job Scheduler,
- Besoin de **performances** et d'optimisations au plus proche du matériel.
  $\rightarrow$ Parfois spécifique : réseau faible latence/large bande : Infiniband/OmniPath, cartes accélératrices : Xeon Phi, Nvidia Teslas, (...). Architecture manycore...
- Des jobs qui tournent des semaines, voir des mois ; la maintenance doit être planifiée longtemps en avance,

Les problématiques HPC

- Applications scientifiques parfois complexes à compiler, avec de nombreuses dépendances,
- Des noyaux souvent plus vieux que d'ordinaire (nombreux soucis avec la glibc)...,
- Environnement cluster = multi-utilisateurs.
  Support : "J'ai besoin du logiciel X..."

Solutions - Installation de logiciels

- Création d'un script d'install pour chaque logiciel
  - Soucis pour les mises à jour de versions : interopérabilité entre logiciels et entre versions ? Reécriture du script d'install ?
  - Manipulation de variables d'environnement PATH/LD_PRELOAD/LD_LIBRARY_PATH et de variables de compilation CFLAGS/CXXFLAGS/LDFLAGS pour définir des chemins d'install plus récents + compilations en statique.

Solutions - Installation de logiciels

- Solutions de packaging : rpm/deb,
  Snap, FlatPack, AppImage, conda,
  EasyBuild, Spack, Nix, Guix... (propre mais peut s'avérer fastidieux)
- Virtualisation (impact sur les performances) ou conteneurs.

Solutions - Glibc

- Mettre à jour le système (Plannification - Maintenance lourde)
- hack précédent : manipulation des variables d'environnement, LD_PRELOAD, etc... (lourd à gérer ; attention aux effets de bord).
- Utilisation de Nix / Guix / Conda...
- Virtualisation (impact sur les performances) ou conteneurs.

Solutions - Installation de logiciels

Fichiers modules - *modulefiles*. A dé/charger avec

```
module load R-3.2.0
module rm R-3.2.0
```

```
#%Module############################
#
# R-3.2.0 modulefile
#
proc ModulesHelp { } {
puts stderr "This␣modulefile␣defines␣the␣library␣paths␣and"
puts stderr "include␣paths␣needed␣to␣use␣R-3.2.0"
}
set is_module_rm [module-info mode remove]
set R_LEVEL 3.2.0
set R_CURPATH /share/apps/bin/R/R-$R_LEVEL
prepend-path PATH        $R_CURPATH/bin/
prepend-path LD_LIBRARY_PATH   $R_CURPATH/lib64/R/lib
prepend-path MANPATH       $R_CURPATH/share/man
append-path  PE_PRODUCT_LIST  R-$R_LEVEL
setenv  R_LIBS     $R_CURPATH/lib64/R/library
```

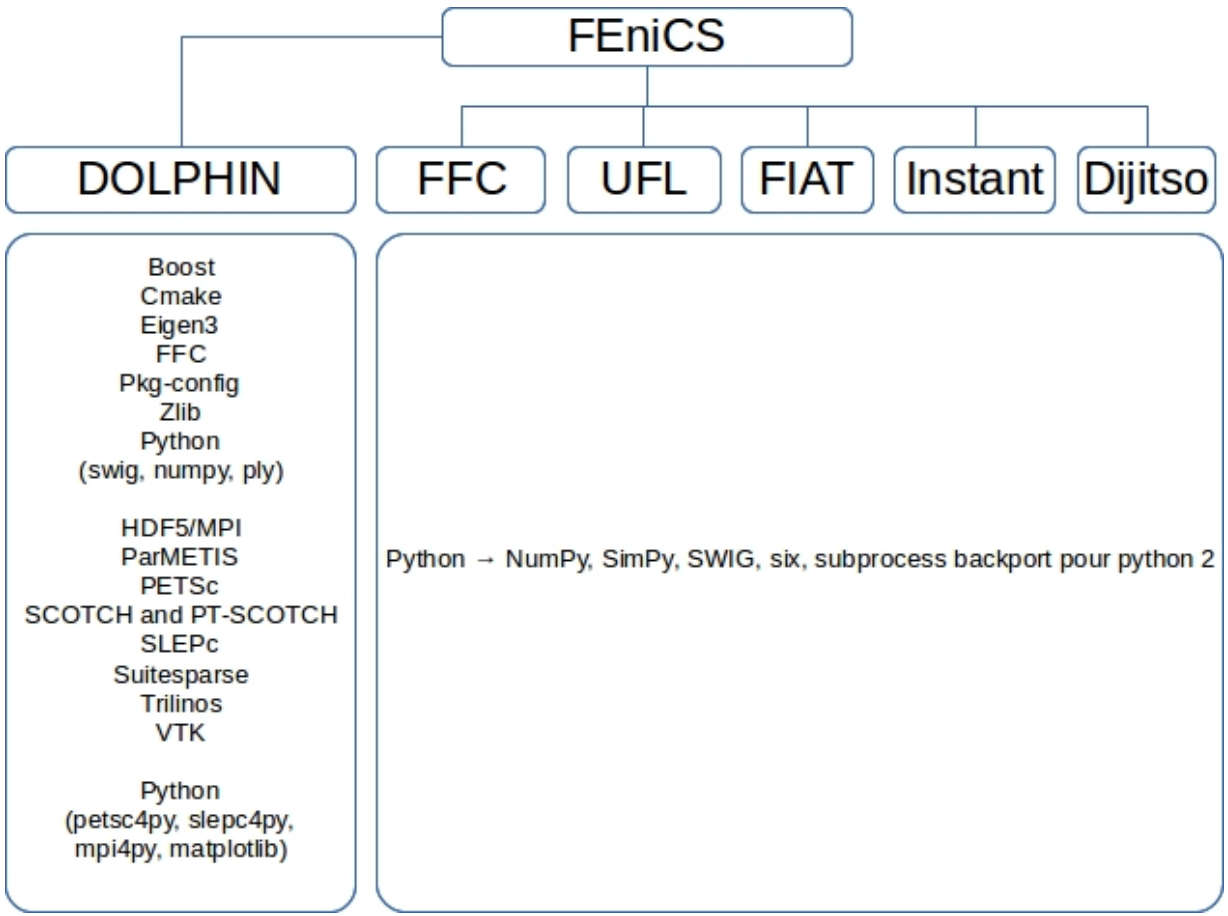
Un exemple simple d'installation logiciel sur cluster

Installation du logiciel Stacks

```
# connexion sur noeud peu ou pas charge
ssh node-0-1
cd /tmp
wget http://catchenlab.life.illinois.edu/stacks/source/stacks-2.0Beta8c.tar.gz
# qqs verifications
tar -tvf stacks-2.0Beta8c.tar.gz && file stacks-2.0Beta8c.tar.gz && \
  sha256sum stacks-2.0Beta8c.tar.gz
tar -xvf stacks-2.0Beta8c.tar.gz
mkdir -p /share/apps/bin/stacks/2.0_Beta
# on charge qqs modules essentiels
module load gcc5.3 libz-1.2.8 libc-2.14
cd stacks-2.0Beta8c
./configure LDFLAGS="-L/usr/local/lib␣-L/share/apps/bin/samtools-0.1.19/" \
  CFLAGS="-I/share/apps/bin/samtools-0.1.19/" --enable-sparsehash --enable-bam \
  --prefix=/share/apps/bin/stacks/2.0_Beta  \
  --with-sparsehash-include-path=/usr/include/sparsehash
make -j($nprocs)
su -
make install
# puis creation du modulefile associe
```

Un exemple plus complexe : FEniCS

L'application FEniCS. @crédits Cédric Clerget <mésocentre de franche-comté / Sylabs.io>



Panorama conteneurs / HPC

1. Solutions orientés HPC : charliecloud, udocker, shifter, singularity
2. autres : docker, rkt, lxc, img, atomic project, smith, LinuxKit, OpenVZ, systemd-nspawn...

Solutions - FEniCS

```
docker run -ti -p 127.0.0.1:8000:8000 -v $(pwd):/home/fenics/shared \
 -w /home/fenics/shared quay.io/fenicsproject/stable:current
```

Mais...

Docker dans tout ça ?

"I want to run Docker containers under SGE." No, you don't - really; or at least your clued-up system manager doesn't want that. It just isn't sane on an HPC system, or probably more generally under resource managers similar to SGE.
https://arc.liv.ac.uk/SGE/howto/sge-container.html

1. Démon root ? Droits élevés en local ?
2. Origine et contenu des images ?
3. Network namespace et compatibilité matériel réseau pour cluster (Intel OmniPath, Infiniband...). Problématique **MPI**.

Orchestrateur vs Job Scheduler

1. Approche microservice vs Job (avec heure de début - fin)
2. Job Scheduler plus avancé sur : mécanismes de files d'attente/de mises en attente ou pause (checkpointing), de priorités / Fair sharing, droits/accès et partages des ressources physiques.
3. Intégration dans l'environnement cluster existant

Intérêt de Singularity

- "J'ai besoin du logiciel X..." -> Ok, faites un conteneur ou récupérez l'existant !
- Dans Singularity, pas de démon, vu comme une application standard par le Job Scheduler. Compatible avec les vieux noyaux, pas de problèmes de sécurité (ou rarement). Pas de 'cgroups' ; les limites sont fixées par le Job Scheduler.

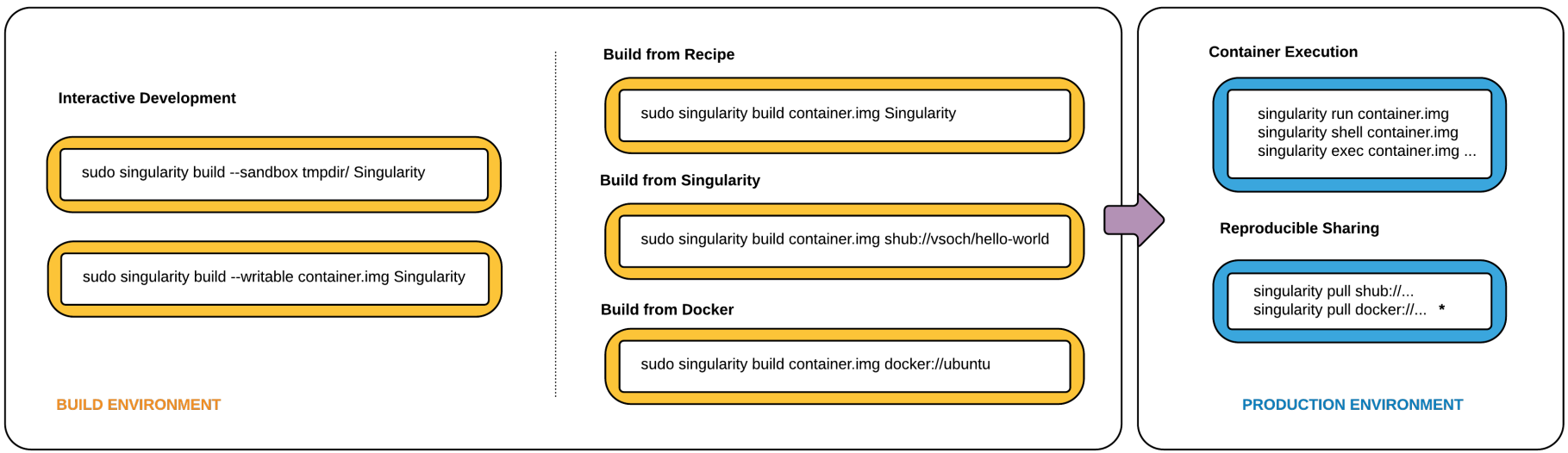
Docker et singularity - OmniPath

Livre blanc Intel : 'Building Containers for Intel®Omni-Path Fabrics using Docker* and Singularity*' Oct. 2017

```
[RHEL7.3 myhost]# docker run --device=/dev/infiniband/rdma_cm \
  --device=/dev/infiniband/uverbs1 \
  --device=/dev/infiniband/ucm1 \
  --device=/dev/infiniband/issm1 \
  --device=/dev/infiniband/umad1 \
  --device=/dev/hfi1_1 \
  --net=host --ulimit memlock=-1 \
  --rm -ti centos73ifs104
```

```
[user@phsmpriv04 singularityimages]$ ./singtest.img
```

Le workflow singularity



* Docker construction from layers not guaranteed to replicate between pulls

Construction d'une image / iozone - Docker

```
FROM ubuntu:16.04
MAINTAINER Eduardo Arango <carlos.arango.gutierrez@correounivalle.edu.co>
ENV IOZONE_VER 3.465
RUN apt-get update
RUN apt-get install -y bash wget build-essential gcc time hdparm
WORKDIR /tmp/
RUN wget 'http://www.iozone.org/src/current/iozone3_465.tar'
RUN tar -xf iozone3_465.tar
WORKDIR /tmp/iozone3_465/src/current/
RUN make linux-AMD64
RUN cp iozone /bin/
ENTRYPOINT ["iozone"]
```

Construction d'une image / iozone - Singularity

```
BootStrap: docker
From: ubuntu:16.04
%environment
  IOZONE_VER=3.465
  export IOZONE_VER
%labels
  Maintainer  Eduardo Arango <carlos.arango.gutierrez@correounivalle.edu.co>
%post
  apt-get update
  apt-get install -y bash wget build-essential gcc time hdparm
  cd /tmp/
  wget 'http://www.iozone.org/src/current/iozone3_465.tar'
  tar -xf iozone3_465.tar
  cd /tmp/iozone3_465/src/current/
  make linux-AMD64
  cp iozone /bin/
%runscript
  iozone "$@"
%test
  iozone -help
```

Commandes Docker - Singularity

Commandes docker	**Commandes singularity**
docker build	sudo singularity build
docker run	singularity run
docker exec	singularity exec
docker run -d	singularity instance.start
docker stop	singularity instance.stop *ou* kill -9 <pid>
docker pull	singularity pull {shub,docker} ://
docker run -ti --rm myimage bash	singularity shell myimage
docker ps	singularity instance.list *ou* lsns
docker inspect	singularity inpect

Autre cas d'utilisation

- Machine GPU très performante (ex : Nvidia DGX-1, autres machines assemblées...) dans un cadre de Deep Learning.
- Accessible sous réservation. Pas d'environnement multi-utilisateurs.
- Solution : 'nvidia-docker' (== solution du support officiel Nvidia) ou 'singularity --nv'

Autre cas d'utilisation

- Images Docker officielles : cuda, digits, tensorflow, caffee...
- Lancement du conteneur tensorflow avec nvidia-docker et le user namespace :

```
nvidia-docker run -d --name $HOSTNAME"_ctmbb" --hostname \
$HOSTNAME"_ctmbb" -t -p 4422:22 -p 8888:8888 -p 6006:6006 \
 -v /data:/home/data -v /data-ssd/sda1/:/home/ssd1:ro \
 -v /data-ssd/sdb1/:/home/ssd2:ro -m 125g --memory-swap 125g \
gpu_tf_sshd
```

Nvidia-docker : Usage GPU/tensorflow avec réservation

```
FROM gcr.io/tensorflow/tensorflow:1.6.0
RUN apt-get update && apt-get install -y nano vim gcc make \
  mlocate git apt-file aptitude libc-bin sudo htop wget \
  iputils-ping dnsutils curl net-tools openssh-server \
  automake autoconf cmake autotools-dev libc6-dev python python3
RUN mkdir /var/run/sshd
RUN mkdir /home/data
RUN mkdir -p /tmp/tensorflow/mnist/logs/
RUN sed -i 's/AcceptEnv LANG LC/#AcceptEnv LANG LC/' /etc/ssh/sshd_config
RUN sed 's@session\s*required\s*pam_loginuid.so@session optional pam_loginuid.so@g'\
 -i /etc/pam.d/sshd
ENV PATH /usr/local/nvidia/bin:/usr/local/cuda/bin:${PATH}
ENV LD_LIBRARY_PATH /usr/local/cuda/extras/CUPTI/lib64:${LD_LIBRARY_PATH}
ENV LD_LIBRARY_PATH /usr/local/nvidia/lib:/usr/local/nvidia/lib64:${LD_LIBRARY_PATH}
ENV LIBRARY_PATH /usr/local/cuda/lib64/stubs
ENV NVIDIA_DRIVER_CAPABILITIES compute,utility
ENV CUDA_PKG_VERSION 8.0.61-1
ENV CUDA_VERSION 8.0.61
EXPOSE 22
EXPOSE 8888
EXPOSE 6006
CMD ["/usr/sbin/sshd", "-D"]
RUN useradd -M -s /bin/bash -d /home/data -g sudo gpuuser
WORKDIR /home/data
RUN echo 'gpuuser:secret' | chpasswd
```

Contacts

- remy.dernat@umontpellier.fr