Charlie Cloud

Violaine Louvet, CNRS

ComputeOps 13 novembre 2018

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 !

Installation

- L'installation est simple et ne nécessite pas de privilège particulier, sauf si on veut le coupler avec Docker (version Docker-CE) pour créer les images
- En espace utilisateur : git clone + make. Donc pas de problème sur les machines de calcul
- La seule contrainte : nécessite un noyau récent (supportant les user namespaces et quelques autres fonctionnalités)
- L'installation de Docker n'est donc pas obligatoire

Test sur un code MPI simple

Description du code

Ce code permet de calculer les distances génétiques entre toutes les paires de reads (découpage de la séquence en taille plus petite) d'un échantillon issu d'un séquenceur nouvelle génération. C'est la première étape pour aller vers du clustering de reads d'un échantillon environnemental (distance entre espèces).
Développé par Alain Franc, Jean-Marc Frigerio, Philippe Chaumeil (INRA).

Deux versions

- Version séquentielle : programme C permettant de calculer la matrice de distance entre plusieurs milliers de reads.
- Version parallèle : parallélisation avec MPI qui permet le calcul de matrice de disctances jusqu'à plus de 150 000 reads (a tourné sur la Blue Gene de l'IDRIS), très scalable.

Dépendances

- Version séquentielle : compilateur C
- Version parallèle : MPI + HDF5

Test sur un code MPI simple (2)

- La construction des deux versions est standard via CMake.
- L'archive des codes se présentent comme un tar.gz mais on peut aussi le récupérer via git.

Dockerfile

```
FROM ubuntu:18.04

RUN apt-get update && apt-get install -y \
    cmake build-essential openmpi-common \
    openmpi-bin libopenmpi-dev libopenmpi2 libhdf5-mpi-dev \
    libhdf5-openmpi-100 libhdf5-openmpi-dev \
    && rm -rf /var/lib/apt/lists/*

# Set the working directory
WORKDIR /disseq

# Copy the current directory contents into the container at /app
COPY disseq_src-*.tar.gz /disseq/

RUN tar xvfz disseq_src-*.tar.gz && mkdir build && cd build && \
    cmake -DDISSEQ_PAR=ON ../D*/. && make
```

Test sur un code MPI simple (3)

- Création de l'image

  ```
  ch-build -t disseq ./
  ```

  Nécessite d'être root / sudoers sur sa machine car c'est un wrapper de la commande *sudo docker build*.
- L'image n'est pas visible car stockée dans le registry local de Docker.
- Pour transférer l'image, il faut la compresser en fichier tar.gz

  ```
  ch-docker2tar disseq ./.
  ```

  A nouveau, demande du mot de passe.
- On peut alors transférer l'image compressée sur la machine dont on veut se servir pour lancer l'exécution.

Test sur un code MPI simple (4)

Installation du container

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

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

Exécution du container, code version séquentielle

```
ch-run -b /home/:/home -w ./disseq -- /disseq/build/disseq/disseq \
-ref /home/louvet/datasets/10V-RbcL_S74_sample.fas \
-query /home/louvet/datasets/TDB_rbcL_300-320_180822.fas \
-out /home/louvet/res.txt
```

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

Test sur un code MPI simple (5)

Exécution du container, code version parallèle, 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 ./disseq -- \
/disseq/build/mpidisseq/mpidisseqAll_full \
-ref /home/louvet/datasets/10V-RbcL_S74_sample.fas \
-query /home/louvet/datasets/TDB_rbcL_300-320_180822.fas \
-out /home/louvet/res.txt
```

- Les processus MPI sont gérés par le container

```
ch-run -b /home/:/home -w ./disseq -- \
mpirun -np 4 /disseq/build/mpidisseq/mpidisseqAll_full \
-ref /home/louvet/datasets/10V-RbcL_S74_sample.fas \
-query /home/louvet/datasets/TDB_rbcL_300-320_180822.fas \
-out /home/louvet/res.txt
```

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

Test sur un code MPI simple (6)

Exécution du container, code version parallèle, plusieurs noeuds

Exécution en dehors du container :

```
mpirun -np 4 ch-run -b /home/:/home -w ./disseq -- \
/disseq/build/mpidisseq/mpidisseqAll_full \
-ref /home/louvet/datasets/10V-RbcL_S74_sample.fas \
-query /home/louvet/datasets/TDB_rbcL_300-320_180822.fas \
-out /home/louvet/res.txt
```

- Il semble qu'il y ait des problématiques de compatibilité mpi
- Tests en cours

Test sur un code python + MPI + HDF5 (1)

Description du code

Ce code permet de pré-traiter des données issues de stations sismologiques réparties sur le territoire, afin qu'elles soient utilisables pour analyse. Il va télécharger des fichiers de données sur le centre de données de sismologie RESIF.
Développé par Albanne Lecointre (ISTERRE, UGA), Pierre-Antoine Bouttier (GRICAD) dans le cadre du projet IWorms.

- Code en python
- Utilise la bibliothèque obspy (traitement de données sismologiques)
- MPI et HDF5-parallel
- Installation via conda

Test sur un code python + MPI + HDF5 (2)

Dockerfile

```
FROM continuumio/anaconda3

RUN /opt/conda/bin/conda config --add channels conda-forge &&  \
    /opt/conda/bin/conda install numpy mpi4py -y && \
    /opt/conda/bin/conda install obspy && \
    /opt/conda/bin/conda config --add channels spectralDNS && \
    /opt/conda/bin/conda install h5py-parallel

# Set the working directory
WORKDIR /iworms

# Copy the current directory contents into the container
COPY preprocess_refact.py /iworms/
COPY whitenWhisper.py /iworms/
```

Test sur un code python + MPI + HDF5 (3)

Cheminement

- Même démarche que pour le premier code : on crée l'image sur son portable, on en fait une archive que l'on remote copie sur la machine de calcul.
- Sur la machine de calcul, on peut utiliser une instance locale de charliecloud pour exécuter l'image.
- Même conclusion que pour le premier code : tout fonctionne bien en mono-noeud, des problèmes non encore résolus en multi-noeuds

Conclusions et Perspectives

Conclusions

- Charliecloud est un système de container prometteur : léger, installable en espace utilisateur, permettant de faire tourner des images docker.
- Simple à installer et à utiliser.
- Pas mal de questionnement sur son utilisation avec mpi en mode multi-noeuds

Perspectives

- Comprendre le fonctionnement en multi-noeuds : les développeurs sont très réactifs, rencontre prévue demain à SC18.
- Tester sur du GPU, en particulier les images Docker fournies par Nvidia en Deep Learning.
- Selon les conclusions, déployer à plus grande échelle sur nos machines et former nos utilisateurs.