Conteneurisation pour GPU

Docker et Singularity

-

Workshop ComputeOps v2

Pourquoi ?

- Isolation des frameworks de DeepLearning et de leur environnement conséquent → les conteneurs sont maintenus à jour sur le DockerHub par les principaux protagonistes (nvidia, google, ...)
- Isolation de ce qui sera fait dedans avec un utilisateur qui aura – ou non - des droits élevés dedans (utilisation du *user namespace*).
- Pratiquement pas de perte de performance.

Avec Quoi ? Comment ?

- Les drivers sont un prérequis sur l'hôte. Le code d'installation cuda *NVIDIA-Linux-<arch>-"$NVID_VER".run"* de nvidia installe tout le nécessaire.
- Le noyau est partagé entre l'hôte et le conteneur ; les drivers kernel nvidia sont donc présents des deux côtés.
- Il ne reste plus qu'à partager les devices, libraries et binaires cuda dans les bons namespaces/volumes (il faut notamment penser aux librairies du pilote au niveau utilisateur).
- Avoir une version récente de Singularity ou Docker.

Docker pour GPU sans nvidia-docker

- Avant docker 0.9 : *runtime = LXC.* Autorisation des *devices* dans les *cgroups* (mode privilégié, options lxc-conf).
- Puis, d'après *StackOverflow :*

```
sudo docker run -ti \
    --device /dev/nvidia0:/dev/nvidia0 \
    --device /dev/nvidiactl:/dev/nvidiactl \
    --device /dev/nvidia-uvm:/dev/nvidia-uvm
    tleyden5iwx/ubuntu-cuda /bin/bash
```

- Enfin, installation de cuda dans le conteneur (la version dans le conteneur doit correspondre à celle sur l'hôte).

Singularity avant l'option `--nv`

- Code gpu4singularity de David Godlove : https://github.com/NIH-HPC/gpu4singularity
- Installation de cuda dans le conteneur avec définition des variables d'environnements (PATH, LD_LIBRARY_PATH).
- La version dans le conteneur doit être la même que sur l'hôte, pour que le conteneur puisse dialoguer avec les bons drivers.

nvidia-docker

- Nvidia-docker : https://github.com/NVIDIA/nvidia-docker
- Wrapper, sous forme de package et service à installer sur l'hôte qui facilite le montage des *devices*, la définition des variables d'environnement, librairies…
- Les commandes se lancent avec *nvidia-docker* au lieu de *docker*
- Améliore la portabilité des conteneurs ; les drivers ne sont pas réinstallés dans le conteneur.

Singularity option `--nv`

- Plus simple car l’environnement utilisateur est le même.
- Chaque fichier nécessaire est bind mounté en *ro* dans le conteneur (chaque librairie, chaque binaire)

  singularity exec --nv docker://nvidia/cuda:9.0-devel-ubuntu16.04 "mount |wc -l"

  - → 67 résultats.

Impacts sur les performances ?

- Docker/nvidia-docker : autour de 1 %

  Cf. : FAQ github nvidia-docker et
  preprint arxiv septembre 2017 (*disclaimer* : je suis l'un des auteurs)

- Singularity : autour de 1 % également (même papier)

Fonctionnement du service GPU - MBB

- 2 machines type « *Nvidia DevBox* » avec respectivement 3 et 4 cartes *TitanXp*,
- Fonctionnement avec réservation par *GRR* qui met à jour un calendrier *caldav*.

  Le *caldav* est analysé par une machine tous les jours *via* un *cron*. Si une réservation démarre, un conteneur est relancé avec utilisation de **nvidia-docker** et du ***user namespace***.

Fonctionnement du service GPU - MBB

- Plusieurs templates de conteneurs disponibles (tensorflow, digits...) avec 2 versions de cuda (8 et 9).
- Surcouche locale appliquée au template pour :
  - Accès SSH dans le conteneur avec utilisateur sudoer.
  - Bind mount de dossiers d'entrée (→ sur disques SSD) et de sorties accessibles par FTP sur des ports différents (un autre conteneur docker).

Environnement cluster ?

- Docker impossible car démon *root*
- Pas de soucis sous Singularity ? A priori, pas simple, mais faisable ; cf : site linuxcluster https://linuxcluster.files.wordpress.com/2018/12/20181204-gpu-accelerated-multi-node-hp-cworkloads-with-singularity1.pdf

contact

- remy.dernat@umontpellier.fr