Ecole informatique IN2P3 2014 : Maîtriser le Cloud

TP Dev : portage d'applications sur le Cloud

TP3 : Tests de performance des MV

Cécile Cavet
cecile.cavet@apc.univ-paris7.fr
Centre François Arago (FACe),
Laboratoire AstroParticule et Cosmologie (APC), LabEx UnivEarthS
APC, Univ. Paris Diderot, CNRS/IN2P3,
CEA/lrfu, Obs. de Paris, Sorbonne Paris Cité, France

3 Juillet 2014

Table des matières

0.1 Introduction

Afin de vérifier les performances des MV pour le calcul scientifique, vous allez réaliser des tests classiques. Vous utiliserez pour cela le paquet « High Performance Computing Challenge (HPCC) benchmark » [1] qui regroupe des tests portant sur les performances des CPU, des accès mémoires et du réseau d'interconnection. Ces tests seront réalisés sur une MV ayant des ressources importantes (8 CPU et 16 GB de RAM) afin de se placer dans les conditions d'une application scientifique de type HPC. Les différentes mesures de performance seront ensuite normalisées par celles effectuées sur le Cluster Arago [2] qui est le Cluster de calcul local du Centre François Arago (FACe) [3]. La comparaison sera relative car le Cluster Arago est dédié au calcul parallèle et donc optimisé dans ce sens. Les mesures relatives seront finalement comparées à une étude sur les environnements virtualisés.

Note : pour pouvoir comparer précisément toutes les caractéristiques du Cluster Arago au Cloud, particulièrement la performance du réseau d'interconnexion des noeuds, il faudrait créer un Cluster virtuel c'est-à-dire un ensemble de MV avec un système de fichier partagé et un gestionnaire de soumission de job (Torque/Maui).

0.2 Configuration de la MV

Vous devez d'abord configurer la MV sur laquelle vous allez effectuer les tests. Comme dans le TP2, vous utiliserez une image disque customisée qui contient des modules Python et les librairies MPI (Open MPI) et BLAS (ATLAS).

0.2.1 Lancement de la MV sur StratusLab

Lancez une MV avec comme ressources 8 CPU, 16 GB de RAM et 2 GB de SWAP, l'option `--context-file` et l'image `Marketplace_ID = HCgdrkS8ZtBQivK9i3xvrcEAvIJ`. Connectez-vous à la MV en tant qu'utilisateur `cloud-user`.

0.2.2 Lancement de la MV sur OpenStack

Lancez une MV avec comme ressources 8 CPU et 16 GB de RAM et l'image `Glance_ID = SL6.5_cloud_school_TP3`. Connectez-vous à la MV en tant qu'utilisateur `cloud-user`.

0.2.3 Compilation

Cette étape consiste à écrire un script de construction qui reflète les caractéristiques de la MV et qui permet de compiler les tests. Il est nécessaire pour cela de connaître l'emplacement du Makefile et des librairies Open MPI et Atlas. Placez-vous dans le répertoire `~/hpcc-1.4.3` afin de compléter le Makefile `hpl/Make.Linux_x86_64` en remplaçant les caractères `[...]` par les informations suivantes :

```
TOPdir       = $(HOME)/hpcc-1.4.3/hpl
MPdir        = /usr/lib64/openmpi
LAdir        = /usr/lib64/atlas
```

Compilez le script complété en utilisant la commande :

```
$ make arch=Linux_x86_64
```

Cette opération crée l'exécutable `~/hpcc-1.4.3/hpcc` qui va être utilisée pour réaliser les tests de la section suivante.
Si besoin, vous pouvez supprimer les fichiers créés par Make avec :

```
$ make clean arch=Linux_x86_64
```

0.2.4 Fichier de configuration

Le fichier de configuration `~/hpcc-1.4.3/hpccinf.txt` permet de sélectionner les paramètres pour les tests High Performance Linpack (HPL) et parallel matrix transpose (PTRANS). Par défaut, les paramètres `n` et `Ns`, que vous devrez modifier pour le test de performance HPL, sont :

```
1            # of problems sizes (N) => n, nombre de problèmes Ns
1000         Ns                    => taille des matrices
```

0.2.5 Exécution

À l'aide de la commande `mpirun` et de l'exécutable `hpcc`, exécutez un job sur 8 coeurs en effectuant :

```
$ mpirun -np 8 hpcc
```

Lors de l'exécution de `hpcc`, vous pouvez vérifier les caractéristiques et contrôler dynamiquement l'utilisation des ressources avec les outils suivants (à éviter pendant les mesures de performance) :

- CPU : `$ cat /proc/cpuinfo` ; `$ mpstat -P ALL 5`
- Mémoire : `$ cat /proc/meminfo` ; `$ free -m -s 2`
- Disque : `$ df -h`

0.2.6 Résultats

Le résultat des différents tests est écrit dans le fichier `~/hpcc-1.4.3/hpccoutf.txt` qui sera utilisé dans la section suivante. Pour voir les résultats spécifiques au test HPL, exécutez :

```
$ grep "WR11C2R4" hpccoutf.txt
```

Vous obtiendrez la deuxième ligne du tableau suivant :

```
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR11C2R4        1000    80     2     2               0.06              1.113e+01
--------------------------------------------------------------------------------
```

Les informations qui vous intéressent pour la mesure des performances sont `N`, la taille des matrices (correspondant au `Ns` du fichier `hpccinf.txt`) et `Gflops`, le nombre d'opérations à virgule flottante (l'unité de mesure du système).
Si vous exécutez plusieurs fois HPCC, les derniers résultats seront mis à la suite des précédents.

0.3 Tests de performance

Les tests inclus dans le paquet HPCC sont au nombre de 4 au niveau noyau (matrix-matrix multiply, STREAM, RandomAccess et FFT) et au nombre de 4 au niveau du système (High Performance Linpack (HPL), parallel matrix transpose (PTRANS), RandomAccess et FFT).
Vous vous concentrerez seulement sur le test HPL (éventuellement aussi sur MPIRandomAccess) car c'est un test mondialement utilisé comme vous le verrez dans la Section 0.3.2.

0.3.1 Méthodologie

Afin de pouvoir quantifier la performance sans trop d'erreur de mesure, il est nécessaire de suivre la méthodologie suivante.

Mesure

Pour effectuer des mesures de performance, vous devez :

1. reproduire l'exécution des tests au moins 5 fois.
2. re-booter la MV entre chaque mesures afin de vider le cache de celle-ci :
   - directement depuis une MV StratusLab : `root@vm$ reboot`
   - depuis le client Nova pour une MV OpenStack : `$ nova reboot MV`
3. faire la moyenne des mesures et calculer l'erreur associée (voir le fichier `~/hpl_compare.py`).

Si vous n'avez pas le temps, vous pourrez vous contenter de 2 mesures (exécution des tests 2 fois) pour ce TP.

Mesure relative

Les mesures seront ensuite comparées à celles effectuées sur le Cluster Arago qui servira de référence pour cette étude. Le Cluster Arago qui est un Cluster MPI a les caractéristiques suivantes :

- 11 noeuds * 16 coeurs (8 coeurs non hyper-threadé).
- 48 GB de RAM par noeud.
- 1 TB de disque.
- inter-connexion des noeuds à 10 GB/s.

Comme c'est un Cluster de production, il est dans les mêmes conditions d'utilisation que les MV du Cloud.
Pour la comparaison des différentes infrastructures, vous utiliserez la formule suivante :

$$\frac{performance_{MV}}{performance_{Arago}} \times 100\% \quad (1)$$

Les caractéristiques (type et mémoire du cache des CPU...) disponible sur le Cluster Arago étant plus performantes que celle auxquelles vous aurez accès sur les MV du Cloud, les mesures de performance seront en faveur du Cluster MPI.

Comparaison

Vous pourrez comparer les mesures relatives trouvées avec celles réalisées par Luszczek et al. [4] dans le cadre d'une étude sur les performances des hyperviseurs qui permettent la création d'environnement virtualisés. Dans cette étude, ils comparent un environnement non virtualisé à une MV créée par différents hyperviseurs (voir Figure 0.3.2 pour l'hyperviseur KVM). Ils utilisent à la place d'un Cluster de calcul comme référence, un environnement « bare metal » qui correspond aux machines physiques du Cloud (sans la virtualisation). Dans ce cas, la comparaison met en évidence uniquement les effets de la virtualisation.
Dans le cas présent de votre étude, il y a deux limitations à cette approche. Tout d'abord, l'accès aux machines physiques n'étant pas possible en général (sur OpenStack, la commande `nova baremetal` le permet), il est plus simple méthodologiquement de comparer les mesures à un environnement extérieur mais cela rend l'analyse des résultats plus compliquée. De plus, le

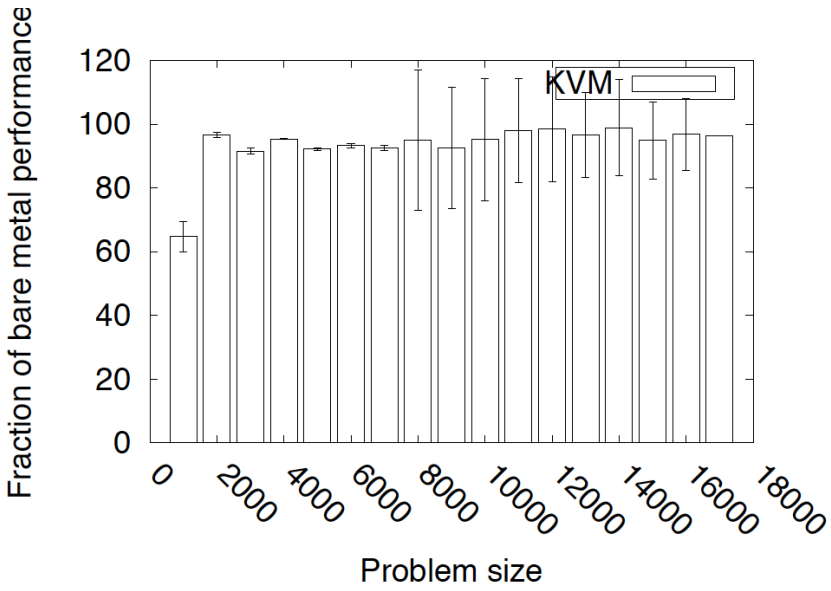

FIGURE 1 – Test HPL : comparaison d'une machine « bare metal » et d'une machine virtuelle instanciée par KVM pour différentes valeurs de N.

Cloud produit certes un environnement virtualisé mais, en phase de production, les ressources y sont partagées entre plusieurs utilisateurs ce qui peut diminuer les performances de celui-ci. La comparaison des mesures relatives des deux études sera donc qualitative.

0.3.2 High-Performance Linpack benchmark (HPL)

Le test HPL est utilisé pour classer les supercalculateurs « High Perfomance Computing » (HPC) mondiaux dans un Top500 [5]. Ce test mesure le nombre d'opérations à virgule flottante lors de l'exécution de la résolution d'un système linéaire dense d'équations.
Sur le Cluster Arago, les résultats obtenus pour le test HPL sont fournis et visibles sur la Figure 2 (voir aussi les fichiers dans `~/Perf_arago`).

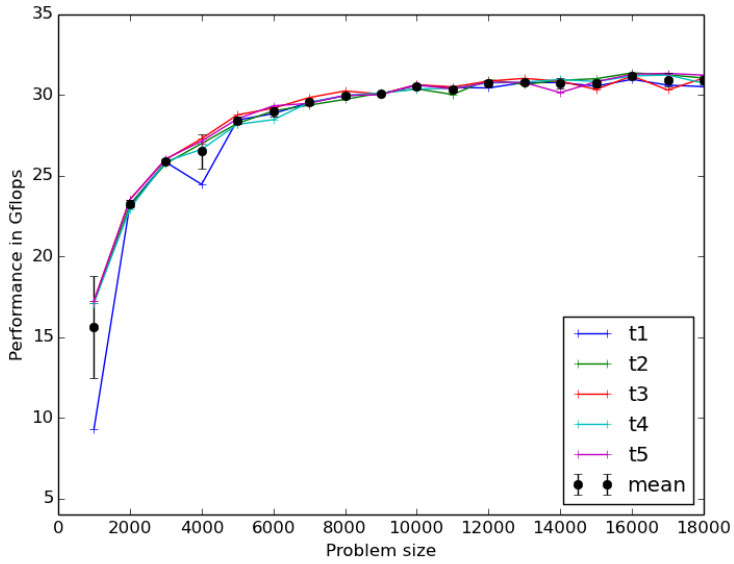

FIGURE 2 – Test HPL sur le Cluster Arago : performance pour 5 mesures et différentes valeurs de N. La moyenne et l'erreur sur les mesures est visible en noir.

Mesure et mesure relative

Afin de réaliser les mesures de performance, vous devez :

- Changez le nombre de problèmes `n` et la taille des matrices `Ns` (c'est-à-dire `N`) dans le fichier `~/hpcc-1.4.3/hpccinf.txt` (voir la Section 0.2.4) de la manière suivante :

```
5            # of problems sizes (N)
1000 2000 4000 8000 16000         Ns
```

- Exécutez le calcul sur 8 coeurs (∼7 min pour 5 problèmes) pour les MV StratusLab et OpenStack. Apres chaque exécution, changez le nom de `hpccoutf.txt` par `hpccoutf_t*.txt` et rebooter (voir Section 0.3.1).

- Déplacez les fichiers `hpccoutf_t*.txt` dans `~/Perf_vm`. Comparez vos résultats au Cluster Arago en utilisant d'abord le script `~/Perf_vm/transfo.sh` pour récupérer les données du test HPL puis le fichier `~/hpl_compare.py` pour le calcul de la moyenne et de l'erreur des mesures (changez `n_time = 5` comme vous n'avez que 5 valeurs de `N`). Le fichier Python va chercher les données dans les deux répertoires `Perf_arago` et `Perf_vm` afin d'effectuer les comparaisons.

- Interprétez les résultats. Vous pouvez vous aider pour cela de la description des caractéristiques des CPU :
  - **Cluster Arago** : Intel Xeon CPU, E5640, fréquence @ 2.67 GHz, cache L3 de 12 MB.
  - **MV OpenStack** : Westmere (Nehalem-C), E56xx/L56xx/X56xx, fréquence @ 3.07 GHz, cache L2 de 4 MB.
  - **MV StratusLab** : QEMU Virtual CPU version (cpu64-rhel6), fréquence @ 2.67 GHz, cache L2 de 4 MB.

Comparaison

Vous pouvez à présent comparer vos résultats à la Figure 1.

0.3.3 MPIRandomAccess

S'il vous reste du temps, vous pouvez regarder les résultats de performance pour le test MPI-RandomAccess et les comparer à la figure 3. Cela implique la modification de `hpl_compare.py`.

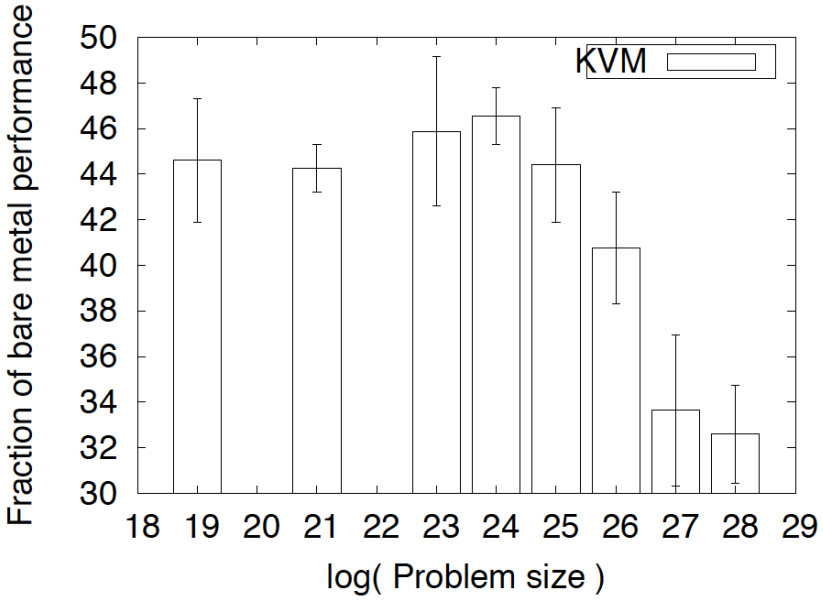

FIGURE 3 – Test MPI : comparaison d'une machine « bare metal » et d'une machine virtuelle instanciée par KVM.

0.4 Références

[1] « HPC Challenge (HPCC) benchmark » : `http://icl.cs.utk.edu/hpcc/index.html`

[2] Cluster Arago : `https://www.apc.univ-paris7.fr/FACeWiki/pmwiki.php?n=Face-cluster.Face-cluster`

[3] FACe : `http://www.apc.univ-paris7.fr/FACe/`

[4] Luszczek et al., « Evaluation of the HPC Challenge Benchmarks in Virtualized Environments », 2011 : `http://icl.cs.utk.edu/news_pub/submissions/vhpc2011_ehbve.pdf`

[5] Top500 : `http://www.top500.org/lists/2013/11/#.U4XhCpR_uWE`