TP1 Grid & Cloud Computing $3^{ème}$ année F2

Cluster de calcul

1 Découverte de l'environnement

IP Frontal : 192.168.5.6
Login : comptes ENT

1.1 Les nœuds

Avant de soumettre des jobs, vous devez connaître l'environnement que vous utilisez. Pour étudier la configuration des nœuds sur lesquels vos jobs vont s'exécuter, exécutez la commande `sinfo -N`.
Les nœuds référencés sont tous accessibles via le cluster. Étudiez les différentes tags qui leur sont associés pour découvrir les ressources du cluster (nombre de nœuds, mémoire, ...).

1.2 Les queues

Lorsque vous soumettez un job, il est affecté à une queue d'ordonnancement. Celles-ci peuvent rassembler les jobs affichant une durée d'exécution proche ou exploitant les mêmes ressources.
Listez les partitions disponibles depuis le master à l'aide de la commande `sinfo -l`. Un appel à `man sinfo` vous guidera vers au moins deux options possibles.
L'état de l'ensemble des jobs soumis sur un cluster peut aussi être présenté par la commande `squeue`.

2 Soumission de jobs

2.1 Exécutable

Nous allons à présent soumettre un premier job. De manière à utiliser tous les moyens de production de sorties disponibles pour un job, ce dernier tentera d'écrire le message *Hello World !* sur la sortie standard, dans un fichier de votre répertoire, dans un fichier à la racine du système de fichier (message d'erreur, espérons...).
Créez un script shell (ou un programme dans le langage de votre choix) qui réalisera les actions évoquées précédemment. Dans un souci de simplicité, votre code n'attendra pas d'option particulière.

2.2 Script de soumission

En vous aidant du cours, proposez un script pour soumettre votre job sur le cluster. Vous utiliserez la commande suivante pour soumettre votre job :

```
sbatch monScript.sh
```

Cette commande retourne l'identifiant du job soumis, à conserver pour les manipulations futures sur ce job (suppression : `scancel`, état : `squeue`, ...).

Indications Vous trouverez les options permettant de modifier les sorties d'erreur et standard du job dans la page de `man sbatch`. Ces options peuvent être définies soit sur la ligne de commande, soit directement dans le script, à la manière du *nom* ou du *walltime* comme le montre l'exemple de votre cours.

2.3 Groupe de jobs

Il n'est pas rare d'avoir à lancer un groupe de jobs se différenciant uniquement par un paramètre d'entrée. Lorsque ce paramètre varie sur une plage d'entiers contiguë, *SLURM* fournit un moyen simple de soumettre de tels ensembles.
Consultez la page de man de `sbatch` pour découvrir la dite option. Pour valider votre compréhension, soumettez le script suivant sur 5 jobs (attention de ne pas en soumettre plus pour éviter de saturer la file d'attente) :

```
#!/bin/sh

echo "Job #${SLURM_ARRAY_TASK_ID}"
```

2.4 Monitoring de jobs

Une fois votre job lancé, vous pouvez vérifier son état à l'aide de la commande `squeue`.

3 Exercice

Cette partie consiste à vous faire construire un script dans son intégralité, afin de paralléliser l'exécution du même algorithme selon plusieurs paramètres.
L'executable que l'on souhaite faire tourner est `stress`, qui permet de charger un système (pour le stresser). L'option `-c` de la commande `stress` permet de définir le nombre de workers (*fork*) qui vont générer de l'activité CPU. On souhaite le faire tourner en parallèle sur 4 CPU pendant 5 min.

Consignes

- Réfléchissez à la manière de fournir le paramètre aux jobs (plusieurs méthodes possibles)
- **Vous devez** ajoutez des contraintes de durée d'exécution (*walltime*), que nous fixerons à 10 minutes, et réserver 4 coeurs pour vos jobs

4 Jobs et Processus...

Il est important de bien distinguer les notions de job et de processus. Combien de **processus** sont exécutés si vous soumettez via SLURM un **job** se contentant d'appeler la commande suivante :

```
srun ./hostname
```

Proposez un diagramme de séquences décrivant ce qui se passe depuis la soumission du job.