TP1 Grille de Calcul $3^{ème}$ année F2

Exploitation d'un cluster de calcul avec Torque/Maui

1 Connexion sur le master et les nœuds

Avant toute chose, nous allons accélérer une étape pouvant être pénible lors de l'accès aux machines de calcul : la connexion distante. Lorsque vous utilisez *ssh*, la machine distante vous demande de lui fournir un mot de passe à chacune de vos connexions. Cette étape a pour but de supprimer la phase de saisie. Cette méthode utilise la techniques des paires de clés de cryptage, que nous espérons connue de tous.

1.1 Génération d'une paire de clés

La commande `ssh-keygen` permet de générer une paire de clés utilisant l'algorithme de cryptage de votre choix.

> Générez une paire de clés publique/privée utilisant l'algorithme que vous voulez, dans le doute consultez `man ssh-keygen`.

1.2 Installation des clés

Vos comptes sont localisés sur le serveur `hpc1.clermont-universite.fr`. Vous devez donc transférer la clé publique sur le serveur distant, avant de l'**ajouter** (`cat »`) dans le fichier prévu à cet effet par *ssh*.

Vérifiez la réussite de votre manipulation en vous connectant sur le serveur `hpc1.clermont-universite.fr`.

2 Découverte de l'environnement

Certaines des commandes utilisées par la suite sont issues de l'ordonnanceur Maui.

2.1 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 `pbsnodes`.

Les nœuds référencés sont tous accessibles via le cluster. Étudiez les différentes tags qui leur sont associés pour découvrir les ressources du cluster du CRRI (nombre de nœuds, mémoire, ...).

2.2 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 queues disponibles depuis le serveur *hpc1* à l'aide de la commande `qstat`. Un appel à `man qstat` vous guidera vers au moins deux options possibles. Notez que pour la suite du TP, vous ne pourrez soumettre des jobs que la queue **practice** de **hpc1**.

> L'état de l'ensemble des queues et des jobs soumis sur un cluster peut aussi être présenté par la commande `showq`.

3 Soumission de jobs

3.1 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 cluster (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.

3.2 Script de soumission

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

```
qsub monScriptPBS.sh
```

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

Indications Vous trouverez les options permettant de modifier les sorties d'erreur et standard du job dans la page de `man qsub`. 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.

3.3 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, *PBS* fournit un moyen simple de soumettre de tels ensembles.

> Consultez la page de man de `qsub` pour découvrir la dite option. Pour valider votre compréhension, soumettez le script disponible à cette adresse.

4 Monitoring de jobs

4.1 Pendant l'exécution

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

> Une autre commande permettant de surveiller l'état de vos jobs vous a été présentée en cours. De **quelle commande** s'agit-il ? De **quel ordonnanceur** dépend-elle ? Testez la sortie de cette commande sur l'un de votre job.

4.2 Après l'exécution

Pour obtenir des informations complètes sur les étapes qu'a suivies votre job tout au long de sa vie, vous pouvez utilisez la commande `tracejob`, qui fournit un log détaillé. Cette commande est aussi utilisable pendant la vie du job et permet de donner des précisions sur l'état de celui-ci.

4.3 Jobs interactifs

> La commande `qsub` vous permet de soumettre un job et de prendre le contrôle du shell qui lui est affectée lors de son exécution. Trouvez l'option vous permettant de soumettre un job interactif et utilisez la pour consulter les variables d'environnement définies par PBS (`$PBS_xxx`).

5 Exercice

Cette partie consiste à vous faire construire un script PBS dans son intégralité, afin de paralléliser l'exécution du même algorithme selon plusieurs paramètres.

Récupérerez l'exécutable situé à cette URL : `http://fc.isima.fr/~passerat/enseignement/grille/tp1/mtdc_3F2`. Il s'agit d'un générateur de nombres pseudo-aléatoires paramétrique, prenant en entrée son paramètre et affichant sur la sortie standard 100 nombres pseudo-aléatoires. Votre travail consistera à lancer 4 exécutions parallèles de ce générateur en lui fournissant des paramètres différents, de sorte que les flux stochastiques produits soient indépendants (propriété intrinsèque de l'algorithme si les paramètres sont différents).

Consignes

- Prenez connaissance de l'exécutable (temps d'exécution, sortie générée, ...)
- 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 nœuds pour vos jobs
- Le script terminé **et testé**, envoyez le en pièce-jointe à `passerat@isima.fr` avant le prochain TP.
- Le nom de votre fichier doit se présenter sous la forme : `script_tp1_nomBinome1_nomBinome2_3f2.pbs`

6 Exercices avancés

6.1 Rapatriement automatique de résultats

Nous souhaitons désormais récupérer les résultats des jobs issus de l'exercice précédent dans un fichier séparé de la sortie standard. Modifier votre script pour que les résultats soient rapatriés automatiquement dans un fichier *results.log* **sans ajouter de commande supplémentaire**. Quel objets doivent être déployé(**e**)s pour que cela fonctionne ?

6.2 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 PBS un **job** se contentant d'appeler la commande suivante :

```
mpiexec -n2 ./random_walk
```

Proposez un diagramme de séquences décrivant ce qui se passe depuis la soumission du job jusqu'à la récupération de ces résultats.

(Vous trouverez l'exécutable `random_walk` *à l'emplacement suivant* `http://fc.isima.fr/~passerat/enseignement/grille/tp1/random_walk`*)*