https://ccub.u-bourgogne.fr/dnum-ccub/spip.php?article379

Introduction à Grid Engine (logiciel de batch)

- Site Public - Calcul -

Date de mise en ligne : lundi 13 février 2012

Introduction à Grid Engine

Cas particuliers : pour avoir une aide sur chaque logiciel, taper la commande "module help nom_du_programme" , par exemple "module help maple".

1- Introduction

SGE remplace le système de batch LSF.

Sun Grid Engine (GE ou SGE) est un système batch avec répartition automatique de charge sur un cluster de calcul. SGE est un logiciel d'origine SUN mis à la disposition de la communauté Open Source, la licence étant reconnue par la Free Software Foundation et l'Open Source Initiative ; Pour en savoir plus :
http://gridengine.sunsource.net

Le but de ce document est de présenter à l'utilisateur les principales commandes et options. L'utilisateur saura soumettre des jobs, suivre leur exécution, et les arrêter éventuellement. Il n'est pas exhaustif et doit être considéré comme une première aide à la mise en oeuvre.

2- Documentation

- Sun Grid Engine 6.2u5 Administration and User's Guide : http://gridengine.sunsource.net/documentation.html
- Aide en ligne du CCUB (commande man sge).
- Man Grid Engine (commande man qsub).
- Accès à la doc du centre : http://www.u-bourgogne.fr/dnum-ccub

3- Concepts et ressources

SGE peut être utilisé dans le cas présent comme un portail d'entrée de soumission de travaux et de répartition de charge sur le cluster du centre, en fonction des ressources demandées par l'utilisateur et disponibles sur les éléments de la grille de calcul ; par exemple : parallélisme sur x processeurs (MPI, OpenMP)...

Le répertoire courant de chaque utilisateur est vu de manière identique par l'ensemble des noeuds constituant les clusters.

Important : le job de soumission doit être en principe un fichier script exécutable (`chmod u+x mon-script`) (qui lancera le binaire ou l'application).

Nous vous rappelons que l'exécution d'une commande en mode interactif ne peut dépasser 3 heures cpu sur les machines opteron du centre ; seul le système batch (ici SGE) permet d'exécuter des travaux dépassant 3 heures de calcul.

4- Quelques définitions

- noeud : plus petit constituant d'un cluster permettant le traitement autonome de données (ou calculateur à un ou plusieurs processeurs).
- cluster : ensemble constitué de noeuds en général identiques et interchangeables administré d'une manière unique et centralisée. Grappe ou ferme en français.
- machine : terme générique pour noeud, voir cluster
- serveur de données : machine dédiée au stockage des données et programmes d'un utilisateur ou d'une application.
- job interactif : job nécessitant l'action de l'utilisateur via clavier, écran et souris
- job batch : job pouvant être exécuté sans aucune action de l'utilisateur pendant son déroulement les données étant lues et écrites sur disque
- machine de soumission : machine sur laquelle l'utilisateur demande une exécution de ses jobs
- mémoire partagée : concerne les travaux parallèles s'exécutant sur un seul noeud de calcul multi-processeur (modèle de programmation OpenMP par exemple).
- mémoire distribuée : concerne les travaux parallèles s'exécutant sur plusieurs noeuds de calcul multi-processeur, interconnectés par un réseau rapide (ici Infiniband) et utilisant le modèle de programmation MPI.

5- Les options par défaut

- envoi d'un message à la fin d'un travail à l'adresse :
- login@u-bourgogne.fr
- le répertoire de soumission est le répertoire de travail
- l'environnement et le PATH sont les mêmes qu'en interactif
- fichier d'entrée (par défaut /dev/null) : l'option -i permet de spécifier un chemin d'accès
- fichier de sortie (par défaut : job_name.ojob_id : l'option -o permet de spécifier un autre chemin d'accès pour ce fichier (standard ouput)

Ces options par défaut peuvent être modifiées par l'utilisateur dans une certaine mesure.

5.1- Ressources définies

Elles peuvent être de différentes natures : architecture (type de processeur), séquentielles ou parallèles, machines réservées par certains laboratoires, taille mémoire...

5.2- Les différentes files d'attente

Nom	Description	Durée max	Nombre de slots max
3d	travaux nécessitant un rendu graphique par GPU (lancer avec `3dsub`)	12 heures	2
batch	travaux séquentiels ou parallèles (mémoire partagée ou distribuée)	21 jours (sauf part000-part091 : limite à 48 heures)	600
gpu	travaux séquentiels ou parallèles (mémoire partagée uniquement) nécessitant un GPU	7 jours (sauf webern07 : limite à 2 heures)	2 (sauf webern07 : limite à 1)
intera	travaux intéractifs (lancer avec `qlogin -q interactif`)	4 heures	4
transf	copie de données entre l'`/archive` et le `/work`	7 jours	4
mv	travaux séquentiels ou parallèles nécessitant des ressources de mémoire importante	21 jours	92

5.3- Environnement d'exécution parallèle ou séquentielle

- **dmp** : environnement parallèle en MPI (distributed memory parallel)
- **smp** : environnement parallèle en mémoire partagée (shared memory parallel)

5.4- Autres ressources

- machines appartenant à un laboratoire pour son usage exclusif
- taille mémoire
- ...

6- Les principales commandes

Ce chapitre donne la liste des commandes les plus fréquemment utilisées, ainsi que leurs principales options.

- **qdel** : suppression d'un job
- **qhost** : état des machines
- **qmon** : lancement de l'interface graphique
- **qstat** : état des jobs
- **status** : état des jobs (qstat amélioré)
- **qsub** : soumission d'un job
- **qqueues** : liste des différentes files d'attente
- **qgroups** : liste des différents groupes et leurs membres

6.1- Soumission de jobs : qsub

Pour soumettre un job dans le cas particulier de notre cluster, on doit se connecter sur une machine interactive Linux (par exemple krenekxx).

Important : le job à exécuter doit être en principe un fichier script ! Ne pas oublier le :

```
chmod u+x mon-script
```

6.1.1- Soumission en mode commande : qsub

Cas général

Le premier argument trouvé par *qsub* qui n'est pas une option est considéré comme le nom de la commande à soumettre, et tout le reste de la ligne comme des arguments de cette commande.

Si aucune commande à exécuter n'est spécifiée sur la ligne de commande, *qsub* lit son entrée standard. Les fonctionnalités de *qsub* sont accessibles en interface graphique à travers *qmon*.

Syntaxe :

```
qsub [options] [scriptfile]
```

Exemples

En géneral on aura :

```
qsub -q file-attente mon_job
```

- soumission du script ; ce job s'exécutera sur une machine quelconque ; il faudra être sûr que toutes les ressources nécessaires sont disponibles quelque soit toutes les machines. Il n'y a pas de file d'attente par défaut.

Cas particuliers : pour avoir une aide sur chaque logiciel, taper la commande "module help nom_du_programme" , par exemple "module help maple".

Cas particulier de Gaussian parallèle

C'est une exécution en mémoire partagée ; nous n'avons pas la possibilité à ce jour de faire tourner Gaussian en mémoire distribuée (via Linda). Le fichier Default.route doit contenir -P- 2 ou -P- 4

```
qsub -q batch -pe smp N mon_job.exe
```

En exigeant N processeurs le job s'executera sur une machine ayant 2 ou 4 processeurs ; (N= 2 ou 4, doit être cohérent avec la valeur du fichier Default.route). Le fichier **mon_job.exe** contient le nom du fichier de données Gaussian (suffixé par .com par défaut) : attention aux confusions.

Une particularité : le job, s'il trouve les ressources nécessaires (au moins 2 processeurs libres sur 1 noeud de calcul) s'exécutera sur la partie séquentielle ou sur la partie parallèle du cluster sans distinction.

Cas particulier de MPI et Infiniband

Exécution en mémoire distribuée via le réseau Infiniband et la bibliothèque MPI.

```
qsub -q batch -pe dmp* 8 -i input-data /usr/ccub/bin/mpiib mon-prog
```

input-data est un fichier d'entrée de données et **mon-prog.com** est un script qui contient les commandes de lancement du binaire (la version compilée du programme).

Rappel : pour compiler un programme MPI la commande suivante est à utiliser : `mpif90` ou `mpif77 (options) -o mon-prog mon-prog.f`

Cas particulier de Jaguar séquentiel

Le fichier **jag.com** contient ce qui suit :

```
 #!/bin/ksh
export SCHRODINGER=/usr/local/schrod7
$SCHRODINGER/jaguar run -WAIT jag.in
```

jag.in est le fichier contenant votre procédure Jaguar ; le mot clef **-WAIT** est important pour l'exécution en batch.

La commande de lancement en batch sera :

```
qsub -q batch jag.com
```

Cas particulier de Gamess séquentiel

La commande est :

```
qsub -q batch /usr/ccub/bin/gmsrun toto.inp
```

Cas particulier de Gamess parallèle

La commande est :

```
qsub -q batch -pe dmp* 4 /usr/ccub/bin/gmsrun toto.inp
```

Cas particulier de Mathematica

On se constitue un fichier **factx.txt** contenant les directives Mathematica par exemple :

```
votre code mathematica....
quit
```

et un fichier **mathem.com** contenant :

```
hostname
time bmathematica
```

La soumission se fait depuis une machine krenekXX par :

```
qsub -q batch mathem.com
```

Attention : il faut utiliser **bmathematica** pour le lancement de Mathematica dans le cas du batch !
Attention : le fichier factx.txt doit impérativement finir par la commande **quit** sinon le job ne se termine pas !

6.1.2- Soumission de jobs par script

C'est une façon un peu différente de soumettre ses travaux. On peut se créer un script contenant les différents paramètres et soumettre ce script par la commande :

```
qsub mon-script
```

Le fichier mon-script pourrait avoir le contenu suivant à titre d'exemple :

```
#!/bin/ksh
#$ -q batch
#$ -o result.out
#$ -J zirconium

module load openmpi/intel/13.1.3

/usr/ccub/bin/mpiib mon-prog
```

Dans l'exemple ci-dessus, on demande l'exécution de mon-prog, les résultats de la sortie standard étant placé dans le fichier **result.out**, l'identificateur de job étant **zirconium**. Bien d'autres paramètres peuvent être utilisé. Attention par contre à la modification de mon-script pendant l'exécution du job ; le fichier **result.out** est en principe écrit en "append". Des arguments peuvent être passés au programme.

6.2- Etat des jobs : qstat

C'est la commande **qstat** ; voir aussi `man qstat`

6.2.1- Quelques options de qstat

- `qstat` : état des jobs en cours
- `qstat -f` : état du cluster
- `qstat -g t` : état complet des jobs parallèles
- `qstat -s z` : historique des jobs

La commande `qstat2` particulière au centre de calcul permet une visualisation triée par user, date, queue ou jobid.

```
qstat2 -u, -d, -q, -j
```

Voir aussi la commande `status` : pour en savoir plus, `status -h`

La commande `qjob2` permet d'avoir la liste des jobs pour un utilisateur ou des informations sur le job parallèle d'un utilisateur ; c'est particulièrement utile pour avoir une liste ordonnée et cohérente des machines utilisées par un job parallèle.

- `qjob2 -u user` : liste des jobs d'un utilisateur
- `qjob2 -j jobid` : détails sur les machines utilisées par un job parallèle

6.2.2- Etats possibles d'un job

Les options de la commande `qstat -s xx` sont les suivantes

- **p** : ?
- **r** : run ; job en cours d'exécution
- **s** : ?
- **ha** : suspendu car soumis à une date d'exécution (`qsub -a`)
- **ho** : suspendu pat l'opérateur
- **hs** : suspendu par le système
- **hu** : suspendu par l'utilisateur
- **hj** : suspendu car en attente de ressources dépendantes de la fin d'un aure job

6.3- Arrêt de jobs : qdel

C'est la commande `qdel jobnum` ; on obtient le **jobnum** via la commande `qstat` ; voir aussi `man qdel`

6.4- Status des machines : qhost

C'est la commande `qhost` ; voir aussi `man qhost`

6.5- Différentes files d'attente : qqueues

C'est la commande `qqueues` avec les paramètres -a, -l, -q queue, -h host.

6.6- Différents groupes : qgroups

C'est la commande `qgroups` avec les paramètres -a, -l, -g group, -h host.

7- Commandes graphiques

7.1- Qmon

`qmon` vous permettra de réaliser en mode graphique toutes les commandes décrites ci-avant.

7.2- Cartographie de la charge

Une cartographie de la charge du cluster de calcul GE est accessible via un navigateur à l'adresse :
http://krenek2000.u-bourgogne.fr/clustermap