Grid Engine au CC-IN2P3

Jobs, ressources, monitoring

Sinikka Loikkanen

Groupe Opération/Support
Centre de Calcul IN2P3
Villeurbanne, France

20 mars 2012

Tutoriel Grid Engine

- Une journée de 9.30 à 17.00
- Objectifs : soumission, monitoring des jobs
- Contenu :
  - I Introduction, architecture, les machines
  - II Commandes de base : `qsub, qstat, qacct, qdel`
    - *Hello World !* — Monitoring — La sortie du job
    - Options de base GE
  - III Ressources informatiques
  - IV Soumission des jobs
    - Job interactif — Job multi-cœur — Array job — Job parallèle
  - V Commandes avancées : `qalter, qhold, qrls`
- Matériel :
  - Indico : `http://indico.in2p3.fr/conferenceDisplay.py?confId=6596`
  - Tutorial : `http://cc.in2p3.fr/docenligne/1007`
  - Scripts : `/afs/in2p3.fr/home/throng/ccin2p3/scripts/tutorial`

Grid Engine au CC-IN2P3

Introduction, architecture, jobs, machines

1. L'ordonnanceur de job, qu'est-ce qu'il fait ?
2. Pour quels types de jobs ?
3. Comment les jobs passent-ils en exécution ?
4. GE au CC-IN2P3
5. Se connecter à une machine d'accueil
6. Positionnement de l'environnement GE

Grid Engine (2/9)

L'ordonnanceur de job, qu'est-ce qu'il fait ?

L'ordonnanceur :

- unique point d'entrée commun à tous les jobs et utilisateurs
- accepte et ordonne des jobs
- exécute les jobs sur le cluster de calcul

GE est un programme d'équilibrage de charge qui alloue des ressources telles que la mémoire, l'espace disque et la CPU à l'exécution d'un job d'une manière optimale

Grid Engine (3/9)
Pour quels types de jobs ?

Grid Engine reconnaît différents types de jobs :

- Jobs séquentiels mono-cœur
- Jobs multi-cœurs – Jobs qui s'exécutent sur plusieurs cœurs, avec une mémoire partagée
- Array jobs – Jobs paramétriques, le même code est exécuté sur des lots de données différents et/ou des jeux de paramètres indépendants
- Jobs parallèles – Ensemble de jobs qui travaillent et modifient le même jeux de paramètres, d'où une exécution simultanée de cet ensemble de jobs
- Jobs interactifs – pour le développement de scripts, compilation

Grid Engine (4/9)

Comment les jobs passent-ils en exécution ?

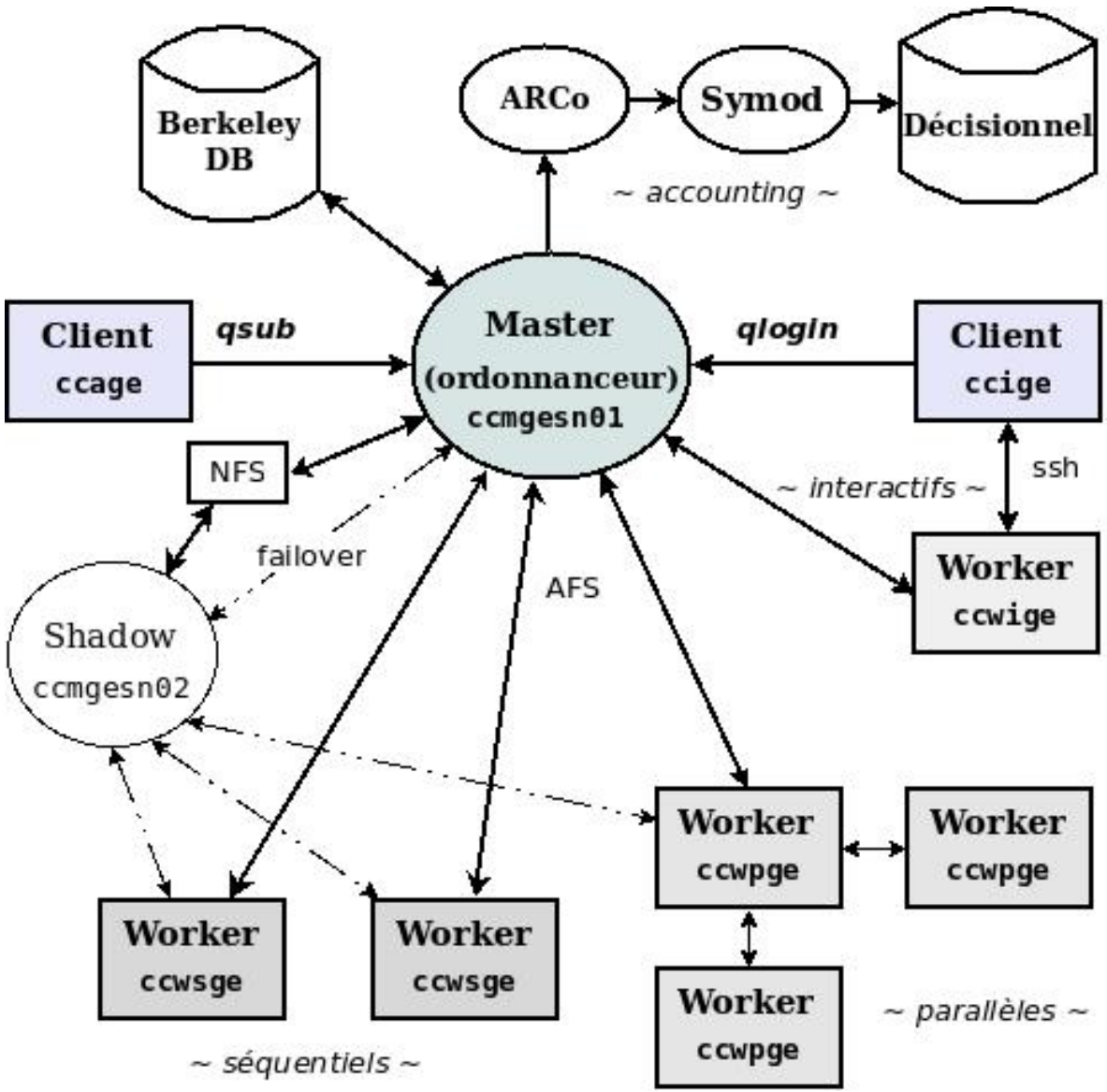
- Un job est toujours soumis à une queue

  Une queue d'exécution correspond à des valeurs par défaut, par example, pour l'espace disque, le temps cpu et la mémoire
- Si l'utilisateur demande une queue
  - Si l'utilisateur est autorisé, l'ordonnanceur vérifie si les exigences matérielles du job demandées correspondent avec les ressources que fournit la queue
    - Si elles correspondent, et s'il y a des ressources disponibles, le job est exécuté
    - Si elles ne correspondent pas, le job reste en queue
  - Si l'utilisateur n'y est pas autorisé, le job reste en queue
- Si l'utilisateur ne demande pas de queue, l'ordonnanceur prend la première queue qui répond aux exigences

Grid Engine

- Sun Grid Engine (SGE) → Oracle Grid Engine (rachat de Sun par Oracle en 2010)
- Version actuelle : version 6.2 update 6 (SGE 6.2u6_12)
- Installation faite en local sur des machines Linux
- Avec gestion du token AFS (token fourni au `qsub`)
- 2 masters (= 1 master + son shadow) qui gèrent la répartition des jobs sur les queues
- ∼1330 serveurs de calcul (`ccwsge`), 64 pour les jobs parallèles (`ccwpge`) et 4 workers interactifs (`ccwige`)

Grid Engine (6/9)
Architecture locale



Grid Engine (7/9)

Se connecter à une machine d'accueil

- Pour soumettre des jobs sur la ferme de calcul, vous devez disposer d'un compte local AFS et vous connecter à une machine d'accueil
  - Pour compiler et tester brièvement vos codes, vous pouvez ouvrir une session interactive en se connectant à une machine interactive **ccige** Jobs interactifs :

```
> ssh [-X] <loginname>@ccige.in2p3.fr
```

  - Pour soumettre des jobs, vous devez vous connecter à une machine d'accueil **ccage** :

```
> ssh [-X] <loginname>@ccage.in2p3.fr
```

- Aujourd'hui, vous avez les droits d'accès à la ferme de test :

```
> ssh [-X] <loginname>@ccage025.in2p3.fr
```

- L'environnement GE est défini automatiquement après le login sur une machine d'accueil/interactive
- Si on n'utilise pas l'environnement par défaut, on doit le définir en exécutant :

```
> source /usr/local/shared/bin/ge_env.sh
# ou
> source /usr/local/shared/bin/ge_env.csh
```

- Si les jobs ont besoin des profiles de shell d'un utilisateur, il lui faut ajouter au début de son script, pour bash :

```
#! /usr/local/bin/bash -l
```

ou utiliser l'option "-S" :

```
> qsub -S "/usr/local/bin/bash -l"
```

TP 1

- Se connecter à la machine interactive, version test
- Positionnement de l'environnement GE si nécessaire
- Afficher votre profil :

  Utilisateur = nom + projet par défaut

```
> qconf -suser <loginname>
```

Commandes de base

`qsub, qstat, qacct, qdel`

1. Soumettre un job : `qsub`
2. Premier script : Hello world !
3. Consulter l'état des jobs en cours : `qstat`
4. Obtenir des informations sur des jobs terminés : `qacct`
5. Supprimer un job : `qdel`
6. La sortie de votre job
7. Options de base GE

Commandes de base (2/13)

Soumettre un batch job : qsub


```
> qsub -P P_<myproject> [ options ] [ scriptfile ]
```

- La commande qsub est utilisée pour soumettre un job
- On soumet toujours un job dans un projet ; le nom du projet est généralement le même que le nom du groupe de l'utilisateur
  – Voir la liste des projets disponibles :

```
> qconf -sprjl
```

```
P_acicmb
P_adonis
P_agape
P_agata
P_alice
...
```

- La queue n'est déclarée que dans les cas particuliers
- Les options sont données dans la ligne de commande ou dans un script

Commandes de base (3/13)

Hello World !

- Premier script : *Hello World !*
- Soumission avec la commande :

```
> qsub -P P_<myproject> hello_world.sh
```

Une fois le job soumis, le système renvoi un numéro d'identifiant pour ce job :

```
Your job 1260 ("hello_world.sh") has been submitted
```

- Avec ce *jobid* :
  - vous trouvez l'information concernant votre job (`qstat`/`qacct`)
  - vous pouvez modifier les demandes de ressources (`qalter`)
  - vous pouvez supprimer votre job (`qdel`)

Commandes de base (4/13)

Consulter l'état des jobs en cours : qstat


```
> qstat [ options ]
```

- Afficher le statut des jobs soumis (en queue/exécution) :

```
> qstat
```

```
job-ID  prior   name       user  state submit/start at     queue slots
-----------------------------------------------------------------------
5947764 0.00000 hello_worl login  qw    01/31/2012 16:22:23         1
```

– Autres statuts ▸ http://cc.in2p3.fr/docenligne/969#JobCheckStatus

- Afficher les jobs se trouvant dans un état spécifique :

```
# qstat -s pending/running/suspended -u <loginname>
> qstat -s prs -u <loginname>
```

- pour le groupe

```
> qstat -s r -u \* -ext |grep <groupname>
```

- Sur un job particulier, s'il est en queue/exécution :

```
> qstat -j <jobid>
```

Commandes de base (5/13)

Obtenir des informations sur des jobs terminés : qacct


```
> qacct [ options ]
```

- Pour obtenir des informations sur un job particulier déjà terminé (pendant la dernière semaine) :

```
> qacct -j <jobid>
```

- Accés aux données des mois précédents, l'option "-f" :

```
> qacct -o <loginname> -j \
    -f /opt/sge/ccin2p3/common/accounting.YYYY.MM
```

- Afficher les informations de derniers 10 jours :

```
> qacct -d 10 -j -o <loginname>
```

- Afficher les informations de comptabilité d'un groupe :

```
> qacct -g <groupname>
```

Commandes de base (6/13)

Obtenir des informations : qacct => exit_status

- Dans la sortie de la commande `qacct -j`, il y a deux lignes utiles : `failed` et `exit_status` pouvant expliquer les raisons pourquoi vos jobs échouent
  Si tous les deux sont "0", votre job a été exécuté et bien terminé pour GE :

```
failed 0
exit_status 0
```

- Si une limite soft est dépassée, les signaux suivants sont envoyés :

exit_status	corresponds to
152 = 24 (SIGXCPU) + 128	SIGXCPU : dépassement de cpu ou mémoire
138 = 10 (SIGUSR1) + 128	SIGUSR1 : pour le temps elapsed
153 = 25 (XFSZ) + 128	XFSZ : pour le disque (fsize)

- Si une limite hard est dépassée, un signal SIGKILL 137 = 9 (SIGKILL) + 128 est envoyé

Commandes de base (7/13)

Obtenir des informations : `qacct` => `failed`

- `failed` indique le problème survenu dans le cas où un job n'a pas pu être démarré sur l'hôte d'exècution

`failed`	explication
1 : assumedly before job	Job could not be started
7 : before prolog	Job could not be started
8 : in prolog	Job could not be started
10 : in pestart	Job could not be started
19 : before writing `exit_status`	
21 : in recognizing job	
25 : rescheduling	Job ran, job will be rescheduled
26 : opening input/output file	Job could not be started, stderr/stdout file could not be opened
28 : changing into working directory	Job could not be started, error changing to start directory
29 : invalid execution state	
37 : qmaster enforced h_rt limit	
100 : assumedly after job	Job ran, job killed by a signal

Commandes de base (8/13)

Supprimer votre job : qdel


```
> qdel <jobid>[,jobid,...]
```

- L'utilisateur peut effacer lui même son job en queue :

```
> qdel 1234
```

- Pour annuler l'ensemble de vos jobs :

```
> qdel -u <loginname>
```

- Pour annuler les ▸ array jobs

TP 2

- Ecrivez/copiez le script suivant :

  /afs/in2p3.fr/home/throng/ccin2p3/scripts/tutorial/hello_world.sh

- Soumettre un job
- Récupérer le numéro du job
- Monitoring du job
- Statut du job
- Annuler le job

Commandes de base (10/13)

La sortie du job

- Par défaut, lors de la soumission d'un batch job, deux fichiers seront créés dans votre $HOME :

```
<jobname>.o<jobid> (défaut standard output)
<jobname>.e<jobid> (défaut standard erreur output)
```

- Lors de la soumission d'un *array job*, il y a aura autant de fichiers qu'il y a de tâches :

```
<jobname>.o<jobid>.<taskid>
<jobname>.e<jobid>.<taskid>
```

- Le nom du job peut être changé avec l'option `"-N"`
- Pour positionner les résultats dans votre répertoire de travail, utilisez l'option `"-cwd"`

Commandes de base (11/13)

Les options de base GE

Option	Description
`-C <prefix>`	permet de changer le préfix par défaut "#$" utilisé dans les fichiers script
`-cwd`	les fichiers output (.e, .o) seront placés dans le dossier de travail
`M <emailaddress>`	envoie un email à cette adresse
`-m a,b,e,n`	un email est sera envoyé : "a" quand le job est tué par GE, "b" quand le job passe en exécution, "e" quand le job est terminé, "n" (par défaut) pas de mail
`-N`	permet de changer le nom du job
`-S [shell path]`	permet de spécifier le shell utilisé lors de la soumission
`-a`	permet de soumettre le job demandant son exécution (RUNNING) à une date précise [[CC]]YY]MMDDhhmm[.SS]
`-e [/path/]file`	permet de changer le nom du fichier stderr
`-o [/path/]file`	permet de changer le nom du fichier stdout
`-l resource=value`	pour définir les ressources à demander pour le job
`-q <queuename>`	
`-pe [type] [num]`	
`-r [y,n]`	

Commandes de base (12/13)

Les options de base GE

Les options peuvent être transférées :

- Via la ligne de commande lors de la soumission du job :

```
> qsub -l sps=1 scriptfile
```

- A l'intérieur de votre fichier script :

```
#$ -l sps=1
```

Dans ce cas, les options doivent être mises au début du fichier script, avant toutes autres lignes correspondant à votre job

- Dans un fichier spécifique `.sge_request` :

```
-l sps=1
```

Ce dernier est une bonne place pour des options par défaut comme le `projet`

TP 3

- Soumettre un script en utilisant les options de base GE
- Positionner les fichiers standard error/output dans votre répertoire de travail
- Changer le nom du job, ...
- Créer un fichier `.sge_request` pour les valeurs par défaut

Ressources informatiques
... calcul, stockage, licences ...

1. Les complexes
2. Resource Quote Set (RQS)
3. Déclarations des ressources de calcul
4. Déclarations des ressources de stockage et de licences

Ressources informatiques (2/7)

Les complexes

- Ressources informatiques partagées : ressources de calcul, de stockage, de licences
- A demander via les commandes `qsub -l` ou `qalter -l`
- Afficher les ressources (= *complex* en langage GE) :

```
> qconf -sc
```

- Les ressources sont de types *requestables* et *consommables* :
  - la colonne `requestable` (`YES`/`NO`) indique si l'utilisateur peut demander la ressource pour son job
  - la colonne `consumable` indique si l'attribut est une ressource consommable (`YES`/`NO`)

```
#name               shortcut       type         relop   requestable consumable default  urgency
#------------------------------------------------------------------------------------------------
arch                a              RESTRING     ==      YES         NO         NONE     0
calendar            c              RESTRING     ==      YES         NO         NONE     0
cpu                 cpu            DOUBLE       >=      YES         NO         0        0
dcache              dcache         INT          <=      YES         YES        0        0
display_win_gui     dwg            BOOL         ==      YES         NO         0        0
excl                excl           BOOL         EXCL    YES         YES        FALSE    0
fsize_used_rate     fsize_rate     DOUBLE       >=      NO          NO         0        0
h_core              h_core         MEMORY       <=      YES         NO         0        0
```

Ressources informatiques (3/7)
Resource Quote Set (RQS)

- RQS : Contrôle des limites sur les ressources
  - http://cctools.in2p3.fr/mrtguser/info_sge_complex.php
- Il n'y a pas de ressources spécifiques par groupe à déclarer
- Mais il y a une limite globale par groupe
  - par service de stockage (hpss, sps, . . .)
  - un nombre de jobs en exécution
- Afficher les RQS fixés pour votre groupe :

```
> qconf -srqs | grep <GROUPNAME>
```

– Par ex.

```
qconf -srqs | grep INDRA
  limit         users @INDRA to srb=500
  limit         users @INDRA to hpss=20
  limit         users @INDRA to sps=70
  limit         users @INDRA to slots=300
```

- Pour savoir qui a utilisé tous les slots de votre groupe :

```
> qstat -u \ -ext -s r | grep <groupname> | awk 'print $5' |
sort | uniq -c ...
```

Ressources informatiques (4/7)

Les ressources de stockage et de licences

- Pour les ressources
  - de stockage : dcache, hpss, irods, mysql, oracle, sps, srb, xrootd
  - de licences : idl, matlab

on note :

```
-l <resourcename>=1
# par exemple :
-l sps=1
-l matlab=1
```

Ressources informatiques (5/7)

Les ressources de calcul : durée

- Pour les ressources de calcul, on note :

```
-l <resourcename>=<value>
```

- Durée :

- pour spécifier la durée *cputime* (= le temps cpu que le job peut utiliser), par ex. pour demander 10 minutes :

```
# durée en format hh:mm:s
-l ct=00:10:00
# durée en seconds
-l ct=600
```

- pour spécifier la durée *realtime* (= le temps réel que le job peut utiliser en tout), par ex. pour définir que le job ne peut durer plus de 4 jours :

```
-l h_rt=96:00:00
```

Ressources informatiques (6/7)

- Mémoire :

- pour spécifier la mémoire :

```
# Les unités à utiliser : B, K, M, G
# La mémoire à demander doit être au minimum 64M
-l vmem=256M
```

- Disque :

- pour spécifier le scratch :

```
# Les unités à utiliser : B, K, M, G
# L'espace disque à demander doit être au minimum 64M
-l fsize=4096M
```

TP 4

- Étudier les RQS de votre groupe
- Ajouter différents types de ressources de calcul à votre script
- Vérifier le status des jobs
- Capter les messages d'erreurs par email

Soumission des jobs au CC-IN2P3

. . . interactifs, multi-cœurs, array jobs, parallèles . . .

1. Jobs & queues
2. Jobs interactifs
3. Batch jobs
4. Job multi-cœurs
5. Array jobs
6. Jobs parallèles

Soumission des jobs (2/13)

Jobs & queues

- GE utilise la notion d'une queue pour distinguer les jobs
- Un job est toujours soumis à une queue
- Une queue d'exécution correspond à des valeurs par défaut : l'espace disque, le temps cpu, la mémoire, ...
- Afficher la liste des queues :

```
> qconf -sql
```

```
                    demon               mc_interactive      pa_long
interactive         huge                mc_long             pa_medium
long                longlasting         mc_longlasting      pa_short
medium              verylong            mc_medium           pa_verylong
short               virtual             mc_short
```

- Afficher les propriétés d'une queue :

```
> qconf -sq <queuename>
```

- On ne spécifie la queue que pour les jobs multi-cœurs et parallèles :

```
> qsub -q <queuename> -pe ...
```

Soumission des jobs (3/13)

Batch jobs : exemples

- Un batch job est un script de shell UNIX pouvant être exécuté sans intervention de l'utilisateur et sans accès à un terminal
- Job avec passage de paramètres :

```
> qsub $HOME/test.sh arg1 arg2 arg3
```

Cette commande permet de passer les arguments `arg1`, `arg2` et `arg3` au niveau du shell script test.sh, `arg1` est récupéré au moyen de la variable `$1`, `arg2` avec `$2` et `arg3` avec `$3`

- Job utilisant une base de données :

```
> qsub -l mysql=1,ct=00:30:00,vmem=1500M,fsize=20GB test.sh
```

Soumission d'un job utilisant la base de données mysql, avec 20 Go de quota sur le scratch du worker, 1.5 Go de mémoire

Soumission des jobs (4/13)

Jobs multi-cœurs


```
> qsub -pe multicores <number_of_cores> -q <queuename> -R y
script.sh
> qlogin -pe multicores <number_of_cores> -q mc_interactive
```

- Utilisation : pour les jobs qui utilisent simultanément plusieurs cœurs avec une mémoire partagée
- Concerne les jobs parallèles s'exécutant sur un seul worker multi-processeur
- L'option `-R y[es]` signifie que une réservation pour ce job devrait être faite
- Afficher les queues disponibles pour les jobs multi-cœurs :

```
> qconf -sql | grep mc_
```

- La demande de temps CPU et de mémoire est exprimée par cœur, et l'espace scratch est global pour le job

Soumission des jobs (5/13)

Jobs interactifs


```
> qlogin -P P_<myproject> [ options ]
```

- Soumet une session de connexion interactive à GE
- Les jobs sont exécutés sur les queue interactives : *interactive* (par défaut), *mc_interactive* (jobs multi-cœurs)
- Utilisation : pour écrire et développer votre script
- Les jobs interactifs ne sont pas mis en queues s'ils ne peuvent pas être exécutés au moment de la soumission
- A utiliser avec les mêmes options que pour `qsub`

Par exemple pour avoir un worker avec une queue offrant 1 heure de limite de temps, avec la mémoire et l'espace disque de 1G :

```
> qlogin -P P_<myproject> -l h_rt=1:00:00,vmem=1G,fsize=1G
```

Soumission des jobs (6/13)

Jobs interactifs

- Vous aurez également votre *jobid* :

```
Your job 355795 ("QLOGIN") has been submitted
```

- Une fois connecté, vous aurez l'accès à un dossier local correspondant à votre *jobid* :

```
> cd /scratch/355795.1.interactive/
```

- Pour finir la session, tapez :

```
> exit
```

TP 5

- Se connecter pour la session interactive
- Soumission du job
  - séquentiel, multi-cœur
  - les options
- Exécuter le code
- Visiter le scratch
- Finir la session

Soumission des jobs (8/13)

Array jobs : qsub -t


```
> qsub -t min[-max[:intervalle]]
```

- Un job qui est constitué de plusieurs tâches
- Utilisation : lorsqu'on veut exécuter un même script avec plusieurs différents ensembles de données
- Les arguments `min`, `max` et `intervalle` sont disponibles par les variables d'environnement `$GE_TASK_FIRST`, `$GE_TASK_LAST` et `$GE_TASK_STEPSIZE`, respectivement
- Pour lancer par ex. un job qui lit les données de 4 différents fichiers (*input[1-4].txt*) et ne veut utiliser que 2 (intervalle 2 : les fichiers 1 et 3) :

```
> qsub -t 1-4:2
```

Soumission des jobs (9/13)

Array jobs : qdel

- *Array jobs* sont aussi supprimés avec la commande `qdel` :
- Pour annuler un ou plusieurs jobs :

```
> qdel <jobid>[,jobid,...]
```

- Pour annuler une tâche d'un job :

```
> qdel <jobid>.<taskid>
```

- Pour annuler plusieurs tâches d'un job :

```
> qdel <jobid>.<taskid_first>-<taskid_last>[:intervalle]
# ou avec l'option ''-t'' :
> qdel <jobid> -t <taskid_first>-<taskid_last>[:intervalle]
```

Par exemple :

```
> qsub -t 1-100 sleeper.sh
> qdel 326431 -t 40-80:5
> qstat
326431 0.00000 sleeper.sh sloikkan qw 03/18/2012 18:12:50
1 1-39:1,41-44:1,46-49:1,51-54:1,56-59:1,61-64:1,66-69:1,
71-74:1,76-79:1,81-100:1
```

TP 6

- Se connecter à une machine interactive
- Soumission du job
  - notions : `min`, `max`, `intervalle`
  - les options
- Vérifier l'état, le statut
- Annulation du job, des tâches

Soumission des jobs (11/13)

Jobs parallèles


```
> qsub -pe <parallelenvironment> <number_of_cores> -q
<queuename> script.sh
```

- Utilisation : pour les jobs parallèles s'exécutant sur plusieurs workers multi-processeur interconnectés et utilisant le modèle de programmation MPI
- Afficher les environnements parallèles disponibles :

```
> qconf -spl
```

- Afficher les queues disponibles pour les jobs parallèles :

```
> qconf -sql | grep pa_
```

La queue est choisie correspondant aux besoins de vos jobs en termes de CPU, la mémoire et l'espace de travail

  - Le cpu et la mémoire sont à calculer par cœur

Soumission des jobs (12/13)

Jobs parallèles

- La sortie du job parallèle est écrite directement dans sa destination finale lors de l'exécution du job, et pas localement sur le worker et transféré à la fin du job vers sa destination finale

- Si vous utilisez les bibliothèques OpenMPI ou MPICH2, vous devez créer un fichier `.mpd.conf` avec les droits 600 (''`-rw -------`'') avec le contenu suivant :

```
# $HOME/.mpd.conf
Secretword = XXXX # où XXXX est un mot secret
```

- ...

Soumission des jobs (13/13)

Jobs parallèles] : TP

TP 7

- ...
-