

Administration et gestion d'un parc informatique:
Répartition de charges
Open Portable Batch System

Danjan Guillaume
Trouillet Thomas
Gobelny Sébastien
Grégory Hagenbourger
Antoine Bantzhaff

Table des matières

1	Introduction	3
1.1	Description du système Pbs	3
1.2	Composants d'un système PBS	3
1.2.1	le "Job Server" : pbs_server	3
1.2.2	l'ordonnanceur : pbs_sched	4
1.2.3	le "Job Executor" : pbs_mom	4
1.2.4	Les commandes utilisateurs	4
1.3	Fonctionnement d'un système pbs sur un cluster	5
1.4	Description d'un traitement	5
1.5	Resources systèmes	6
2	Configuration	8
2.1	Serveur Pbs	8
2.1.1	installation des composants logiciel	8
2.1.2	initialisation	8
2.1.3	configuration des différents composants du serveur pbs_serv	9
2.1.4	configuration des différents composants de l'ordonnan- ceur pbs_sched	11
2.1.5	configuration de pbs_mom sur le serveur	13
2.2	Hôte d'exécution	13
2.2.1	installation des composants logiciel	13
2.2.2	configuration de l'hôte d'exécution : pbs_mom	13
2.3	Client Pbs	14
2.3.1	installation des composants logiciel	14
3	Exécution de jobs sur un cluster	15
3.1	fichier de configuration de pbs_serv	15
3.2	fichier de configuration de pbs_mom	16
3.2.1	machines à 350Mhz	16

3.2.2	machines à 1200Mhz	17
3.3	lancement des deamons	17
3.3.1	sur le serveur	17
3.3.2	sur les hôtes d'exécution	17
3.4	exécution d'un job	17
3.4.1	shell script décrivant le traitement	17
3.4.2	soumission du job	18
3.4.3	Informations et statistiques d'exécution	18
3.4.4	Forcer l'exécution d'un job	18
3.4.5	Supprimer un job	18
3.4.6	Re-exécuter un job	18

Chapitre 1

Introduction

1.1 Description du système Pbs

OpenPbs est un système de gestion de ressource et de travaux. Il permet de soumettre des travaux (sous forme de traitements batch décrit par des shell scripts), de préserver et protéger ces traitements jusqu'à ce qu'ils soient exécutés et finalement de renvoyer le résultat de l'exécution de chaque traitement. Les shell scripts servent à contrôler les différents paramètres d'exécution d'un traitement.

PBS peut-être installé et configuré pour supporter l'exécution de jobs sur un seul système, ou sur une grappe de cluster.

1.2 Composants d'un système PBS

Un système PBS est composé de quatre composants : le "job server", le "job executor", l'ordonnanceur et des commandes utilisateur.

1.2.1 le "Job Server" : pbs_server

Le "Job Server" est le composant central, il offre des opérations basiques comme la réception et la création de traitements, la modification des travaux en cours, la protection des traitements contre les plantages du système, et la mise en exécution des travaux.

Le serveur gère une ou plusieurs files d'attente. Les travaux contenus dans une file d'attente ne sont pas forcément exécutés dans un ordre FIFO. Il existe deux types de files d'attente les files d'attente de routage et celles d'exécution. Les files d'exécution ne contiennent que des travaux en attente

d'exécution (attention : un traitement en cours d'exécution reste dans la file d'attente). Les travaux contenus dans une file de routage sont en attente de déplacement dans une autre file d'attente (sur le même serveur ou sur un serveur différent). Les files de routage permettent par exemple de classer les travaux selon leurs paramètres d'exécution.

1.2.2 l'ordonnanceur : `pbs_sched`

L'ordonnanceur contrôle quel est le prochain traitement à exécuter, et où il doit l'être. L'ordonnanceur communique avec les différents "Job executor" afin de connaître leurs disponibilités et ainsi faire de la répartition de charge.

1.2.3 le "Job Executor" : `pbs_mom`

Le "Job executor" ne sert qu'à exécuter les traitements. Il reçoit du serveur Pbs, une copie des travaux, ouvre une nouvelle session à l'identique de celle de l'utilisateur (si possible). `pbs_mom` a aussi la responsabilité de retourner le résultat de l'exécution du traitement à l'utilisateur. `Pbs_mom` doit être exécuté sur chacun des noeuds de calcul.

1.2.4 Les commandes utilisateurs

PBS offre des outils en ligne de commande et une interface graphique. Ces outils permettent de soumettre, monitorer, modifier et supprimer des travaux. Ces commandes peuvent être installées sur n'importe quel machine supportant Pbs, et ne nécessite pas la présence d'un des autres composants.

Il existe trois types de commandes :

Commandes utilisateur

- `qsub` : soumission d'un traitement
- `qstat` : affichage du status d'un traitement
- `qdel` : suppression d'un travail
- `qselect` : permet de lister les travaux en fonction de certains critères
- `qrerun` : ré-exécution d'un traitement
- `qorder` : echange l'ordre de deux travaux dans une file d'attente
- `qmove` : deplace un traitement d'une file d'attente dans une autre
- `qhold` : permet d'empêcher l'exécution d'un job (le place en attente pour une durée indéfinie)
- `qalter` : modification des attributs d'un traitement
- `qmsg` : envoi d'un message à un job

- qrls : permet de supprimer un point d'attente d'un job (cf qhold)

Commandes opérateur

- qenable : permet à une file d'attente d'accepter des traitements
- qdisable : empêche une file d'attente d'accepter des traitements
- qrun : force le serveur Pbs à exécuter un traitement
- qstart : permet à une file d'attente de traiter les travaux qu'elle contient
- qstop : empêche les travaux d'une file d'attente d'être éligible
- qterm : arrêt du serveur Pbs

Commandes administrateur

- qmgr : console d'administration du system Pbs
- pbsnodes : liste les noeuds de calcul rattachés au serveur Pbs

1.3 Fonctionnement d'un système pbs sur un cluster

Le fonctionnement d'un système Pbs, peut être décomposé en neuf phases (cf figure 1 page 7) :

- phase 0 : soumission d'un traitement,
- phase 1 : évènement temporel de début de cycle d'ordonancement
- phase 2 : le serveur demande à l'ordonnanceur de choisir le prochain job à exécuter,
- phase 3 : l'ordonnanceur demande au "job executor" des informations sur l'état des systèmes hôtes,
- phase 4 : les "job executor" renvoient les informations demandées,
- phase 5 : l'ordonnanceur demande une description des différents travaux,
- phase 6 : le serveur renvoie le status de chacun des travaux,
- phase 7 : l'ordonnanceur informe le serveur du prochain traitement à exécuter,
- phase 8 : le serveur envoie une copie du job au "job executor" choisi par l'ordonnanceur.

1.4 Description d'un traitement

La base du système Pbs est le traitement. Il peut :

- être batch ou interactif

- définir une quantité de ressources systèmes nécessaires,
- définir une priorité,
- définir un temps d'exécution,
- envoyer un mail à l'utilisateur avant, après l'exécution d'un traitement,
- définir des dépendances (after, afterOk, afterNotOk, before, ...),
- être synchronisé avec d'autres traitements.

1.5 Ressources systèmes

Chaque traitement a la possibilité de définir la quantité de ressources (minimum et maximum) nécessaire à son exécution. Le nombre et le type des ressources spécifiées est dépendant de la plateforme d'exécution. En général les critères concernant les ressources nécessaires sont les suivants :

- cput : temps cpu utilisé par tous les process d'un traitement,
- pcpur : temps cpu utilisé par un process d'un traitement,
- mem : quantité de mémoire physique demandée par le traitement
- pmem : quantité de mémoire physique demandée par un des process d'un traitement
- vmem : quantité de mémoire virtuelle utilisée par le traitement
- pvmem : quantité de mémoire virtuelle demandée par un des process d'un traitement
- file : taille de fichier maximum qu'un process peut créer
- host : nom de l'hôte sur lequel le job doit tourner
- nodes : nombre et types de noeuds devant être réservés de manière exclusive pour le job,
- ...

Si l'exécution d'un traitement nécessite plus de ressources systèmes que demandées, ce dernier sera annulé.

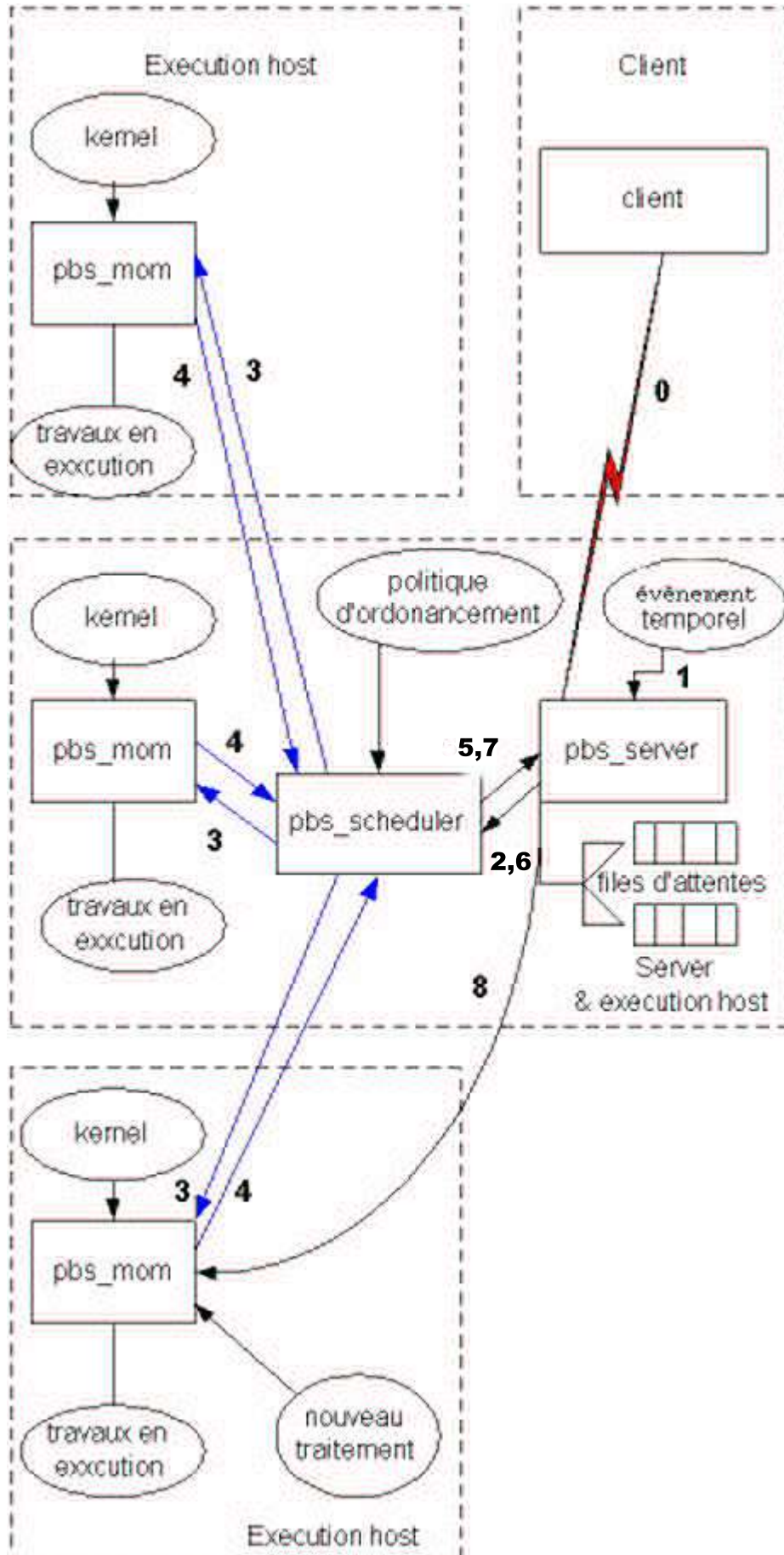


Figure 1 : Schéma de fonctionnement d'un système pbs sur un cluster

Chapitre 2

Configuration

Il est possible de configurer trois types de machines :

- serveur Pbs (pbs_serv + pbs_sched + pbs_mom + commandes utilisateurs) : exécution de jobs, ordonnancements, gestion des files d’attente ...
- hôte d’exécution (pbs_mom + commandes utilisateurs) : exécution de jobs ...
- client Pbs (commandes utilisateurs) : soumission de jobs sur un serveur Pbs

2.1 Serveur Pbs

2.1.1 installation des composants logiciel

1. décompression des sources : `tar -zxvf OpenPBS_2_3_16.tar.gz`
2. `./configure --prefix=$PBS_HOME --enable-server --enable-mom --enable-clients --set-default-server='uname -n' --set-server-home=/etc/PBS/ --set-sched=fifo`
3. `make`
4. `make install`

2.1.2 initialisation

Avant la première exécution il est nécessaire de lancer la commande `pbs_server -t create`.

2.1.3 configuration des différents composants du serveur `pbs_serv`

La configuration du serveur se fait par l'intermédiaire de la commande `qmgr`.

information générale

Configuration de l'envoi automatique de mails (après exécution ou échec d'un traitement) :

`Qmgr : set server mail_from = antoine@www-aius.u-strasbg.fr`

configuration des access control lists

1. liste des hôtes pouvant utiliser le serveur : `Qmgr : set server acl_hosts = *.u-strasbg.fr`
2. prise en compte des ACL : `Qmgr : set server acl_host_enable = true`

configuration des listes d'attentes

Création de trois files d'attentes. Une file pour les processus légers, une pour les processus lourds et une file de routage (routage des jobs dans une des deux autres listes selon les paramètres du traitement)

1. suppression de la file d'attente par défaut : `Qmgr : delete queue workq`
2. création des listes d'attentes :
 - (a) `Qmgr : create queue lowQueue`
 - (b) `Qmgr : create queue highQueue`
 - (c) `Qmgr : create queue routingQueue`

configuration de la file `lowQueue`

1. type de la file d'attente `Qmgr : set queue lowQueue queue_type = execution`
2. temps cpu maximal des traitements de la file d'attente (45 minutes)
`Qmgr : set queue lowQueue resources_max.cput = 00 :45 :00`
3. la file d'attente accepte des jobs `Qmgr : set queue lowQueue enabled = true`
4. la file d'attente est démarrée `Qmgr : s q lowQueue started = true`

configuration de la file *highQueue*

1. type de la file d'attente Qmgr : set queue highQueue queue_type=execution
2. temps cpu minimum demandé par les traitements de la file d'attente (min. 45 minutes) Qmgr : set queue highQueue resources_min.cput=00 :44 :59
3. la file d'attente accepte des jobs Qmgr : set queue highQueue enabled=true
4. la file d'attente est démarrée Qmgr : set queue highQueue started=true

configuration de la file *routingQueue*

1. type de la file d'attente Qmgr : set queue routingQueue queue_type=route
2. la file d'attente destination Qmgr : set queue routingQueue route_destinations="lowQueue,highQueue"
3. la file d'attente accepte des jobs Qmgr : set queue routingQueue enabled=true
4. la file d'attente est démarrée Qmgr : set queue routingQueue started=true

configuration des ressources demandées et de la file d'attente (par défaut)

1. temps cpu nécessaire par défaut Qmgr : set server resources_default.cput=30 :00
2. quantité de mémoire nécessaire par défaut Qmgr : set server resources_default.mem=80mb
3. file d'attente Qmgr : set server default_queue=routingQueue

ajout des noeuds de calcul

noeud de calcul dédié Qmgr : create node nom_noeud

noeud de calcul ne servant pas seulement à exécuter des jobs Pbs

Qmgr : create node nom_noeud=time-shared

remarque Si le noeud a plusieurs processeurs, rajouter np=x (où x est le nombre de processeurs (virtuel ou non) du noeud de calcul). De plus la répartition de charge n'est effectuée que sur les noeuds de type time-shared (les autres noeuds étant considérés comme libre si aucun job Pbs n'est exécuté dessus).

sauvegarde de la configuration

```
qmgr $nom_server -c "print server" > /home/antoine/pbs/conf/server.conf
```

chargement de la configuration

```
qmgr $nom_server < /home/antoine/pbs/conf/server.conf
```

2.1.4 configuration des différents composants de l'ordonnanceur pbs_sched**détail du fichier de configuration de l'ordonnanceur**

Dans le fichier '*\$PBS_HOME/sched_priv/sched_config*', ajout des lignes :

```
# round_robin
#     Exécution d'un job de chaque file d'attente avant d'exécuter
#     le deuxième job de la première file d'attente
round_robin: False      all

# by_queue
#     Si cette variable n'est pas renseigné, l'ordonnanceur va
# voir les traitements comme faisant partie d'une seule et
# même file d'attente, en ne considérant pas les
# files d'attentes définies par l'administrateur
by_queue: True ALL

# strict_fifo
#     exécution des jobs dans un ordre first in first out
#     Si un travail ne peut être exécuté, déplacement dans la
# prochaine file d'attente et n'exécute plus de traitements de cette
# file d'attente, même si certains peuvent l'être !
#     Si cette variable n'est pas assignée cela peut entraîner l'appartition
#     de beaucoup de jobs morts...
strict_fifo: true      ALL

#
# fair_share
#     ordonnance les jobs selon leurs utilisations et leurs partages
fair_share: True      ALL

#
```

```
# help_starving_jobs
#   Les travaux trop longtemps en attente seront considérés comme mort.
#   Une fois qu'un job est considéré comme mort, l'ordonnanceur
#   n'élira aucun job tant que les jobs morts ne pourront être
#   exécutés.
help_starving_jobs:    false    ALL

#
# backfill
#   Cette option en conjonction avec la variable help_starving_job
#   permet au serveur de deviner quand les jobs morts pourront être
#   exécutés et ainsi l'ordonnanceur pourra essayer d'exécuter un
#   maximum de travaux court avant
backfill:              true     ALL

# load_balancing
# Faire de la répartition de charge sur les noeuds de type time-shared,
# sans utiliser la méthode "round-robin"
load_balancing: True     ALL
load_balancing_rr: false    ALL

# sort_queues
# autorise la réorganisation des files d'attentes
sort_queues           true     ALL

# sort_by:
# réorganisation des files d'attentes selon plusieurs critères
# d'abord les processus courts
# puis ceux demandant le moins de mémoire
#   et enfin selon leurs priorités
sort_by: multi_sort    ALL
key:    shortest_job_first
key:    smallest_memory_first
key:    high_priority_first
```

mise en place de l'ordonancement des travaux

Pour pouvoir utiliser l'ordonnanceur (et ainsi faire de la répartition de charge) : Qmgr : set server Scheduling = True

2.1.5 configuration de pbs_mom sur le serveur

voir page `pagerefconfPbsMom` : "configuration de l'hôte d'exécution : pbs_mom"

2.2 Hôte d'exécution

2.2.1 installation des composants logiciel

1. décompression des sources : `tar -zxvf OpenPBS_2_3_16.tar.gz`
2. `/configure -prefix=$PBS_HOME -enable-mom -enable-clients -disable-server -set-sched=no -set-default-server=$NomServer -set-server-home=/etc/PBS/`
3. `make`
4. `make install`

2.2.2 configuration de l'hôte d'exécution : pbs_mom

Le fichier '`$PBS_HOME/mom_priv/config`', doit contenir les lignes suivantes :

```
#accepte les jobs en provenance de ...
$clienthost nom_server

#seul les machines ... peuvent avoir des infos sur
#les jobs en cours (cf qstat)
$restricted *.u-strasbg.fr

#multiplicateur du cpu permettant d'avoir une base
# de comparaison entre les machines du cluster
#méthode de calcul du coeff multiplicateur:
# fréq_machine_référence / fréq_machine
$cputmult x.x

#paramètres nécessaires sur un noeud de type time-shared
# le load va de 0 à 3.0 (si utilisation totale des ressources cpu)
#ideal_load = si le load descend en dessous de la valeur spécifiée
```

```
pbs_mom prévient le serveur qu'il est disponible
  $ideal_load 1.0

#max_load = si le load dépasse load_max alors le serveur sera prévenu
# afin de ne pas lancer d'autres traitements
  $max_load 2.50
```

2.3 Client Pbs

2.3.1 installation des composants logiciel

1. décompression des sources : `tar -zxvf OpenPBS_2_3_16.tar.gz`
2. `/configure --prefix=$PBS_HOME --disable-mom --enable-clients --disable-server --set-sched=no --set-default-server=$NomServer --set-server-home=/etc/PBS/`
3. `make`
4. `make install`

Chapitre 3

Exécution de jobs sur un cluster

3.1 fichier de configuration de pbs_serv

Lors de notre test, la machine serveur était la machine pc3c13.

```
#
# Create queues and set their attributes.
#
#
# Create and define queue highQueue
#
create queue highQueue
set queue highQueue queue_type = Execution
set queue highQueue resources_min.cput = 00:45:00
set queue highQueue enabled = True
set queue highQueue started = True
#
# Create and define queue lowQueue
#
create queue lowQueue
set queue lowQueue queue_type = Execution
set queue lowQueue resources_max.cput = 00:44:59
set queue lowQueue enabled = True
set queue lowQueue started = True
#
# Create and define queue routingQueue
```



```
#
create queue routingQueue
set queue routingQueue queue_type = Route
set queue routingQueue route_destinations = lowQueue
set queue routingQueue route_destinations += highQueue
set queue routingQueue enabled = True
set queue routingQueue started = True
#
# Set server attributes.
#
set server scheduling = True
set server acl_host_enable = True
set server acl_hosts = *.u-strasbg.fr
set server default_queue = routingQueue
set server log_events = 511
set server mail_from = antoine
set server query_other_jobs = True
set server resources_default.cput = 00:30:00
set server resources_default.mem = 96mb
#un tour d'ordonancement toutes les 10 minutes
set server scheduler_iteration = 600
```

3.2 fichier de configuration de pbs_mom

Liste des machines du cluster :

1. pc3c13 : 350Mhz
2. pc3c20 : 350Mhz
3. pc3c16 : 1200Mhz
4. pc3c17 : 1200Mhz
5. pc3c18 : 1200Mhz

La machine de référence est pc3c13.

3.2.1 machines à 350Mhz

```
$clienthost pc3c13
$restricted *.u-strasbg.fr
$ideal_load 1.0
$max_load 2.30
$cpumult 1.0
```

3.2.2 machines à 1200Mhz

```
$clienthost pc3c13
$restricted *.u-strasbg.fr
$ideal_load 1.0
$max_load 2.30
$cpumult 3.42
```

3.3 lancement des deamons

3.3.1 sur le serveur

lancement de *pbs_serv* et *pbs_sched*.

3.3.2 sur les hôtes d'exécution

lancement de *pbs_mom*.

3.4 exécution d'un job

3.4.1 shell script décrivant le traitement

Le programme à exécuter est *'/home/antoine/test/test'*. Les lignes *#PBS -j oe* et *#PBS -o ser_job.out* permettent de mixer la sortie standard et la sortie d'erreur dans un fichier (ici : *'ser_job.out'*). De plus on spécifie des ressources nécessaires pour ce traitement :

- 10 minutes de temps cpu
- 64Mo de mémoire

```
#PBS -j oe
#PBS -o ser_job.out
#PBS -l mem=64mb,cput=00:10:00

echo "Job started at `date` on echo `uname -a`

# déplacement vers sous rep
cd $HOME/pbs/test/

#lancement du prog test
./test
```

```
# Print the ending time.  
echo "Job ended at 'date'"  
  
# Exit  
exit 0
```

3.4.2 soumission du job

```
'qsub mon_shell_script'.
```

3.4.3 Informations et statistiques d'exécution

informations générales sur les files d'attente d'un serveur

```
'qstat -W mon_serveur'.
```

informations détaillées sur tous les jobs d'un serveur

```
'qstat -W mon_serveur -f '.
```

informations détaillées sur un job

```
'qstat -W mon_serveur -f job_id '.
```

3.4.4 Forcer l'exécution d'un job

```
'qrun -H mon_serveur job_id '.
```

3.4.5 Supprimer un job

```
'qdel -W delai_avant_suppression job_id '.
```

3.4.6 Re-exécuter un job

```
'qrerun job_id '.
```