SGE:
Formation utilisateur

Dorin Preda
(Serviware Toulouse)

25 Mars 2010

Plan

- SGE: vue globale
- Concepts SGE
  - Jobs, utilisateurs, calendriers, projets
  - Ressources, environnement spéciaux, quotas
  - Files d’attente
- La soumission
- Commandes
- Accounting

SGE: vue globale

Architecture

Fonctionnement

Composants

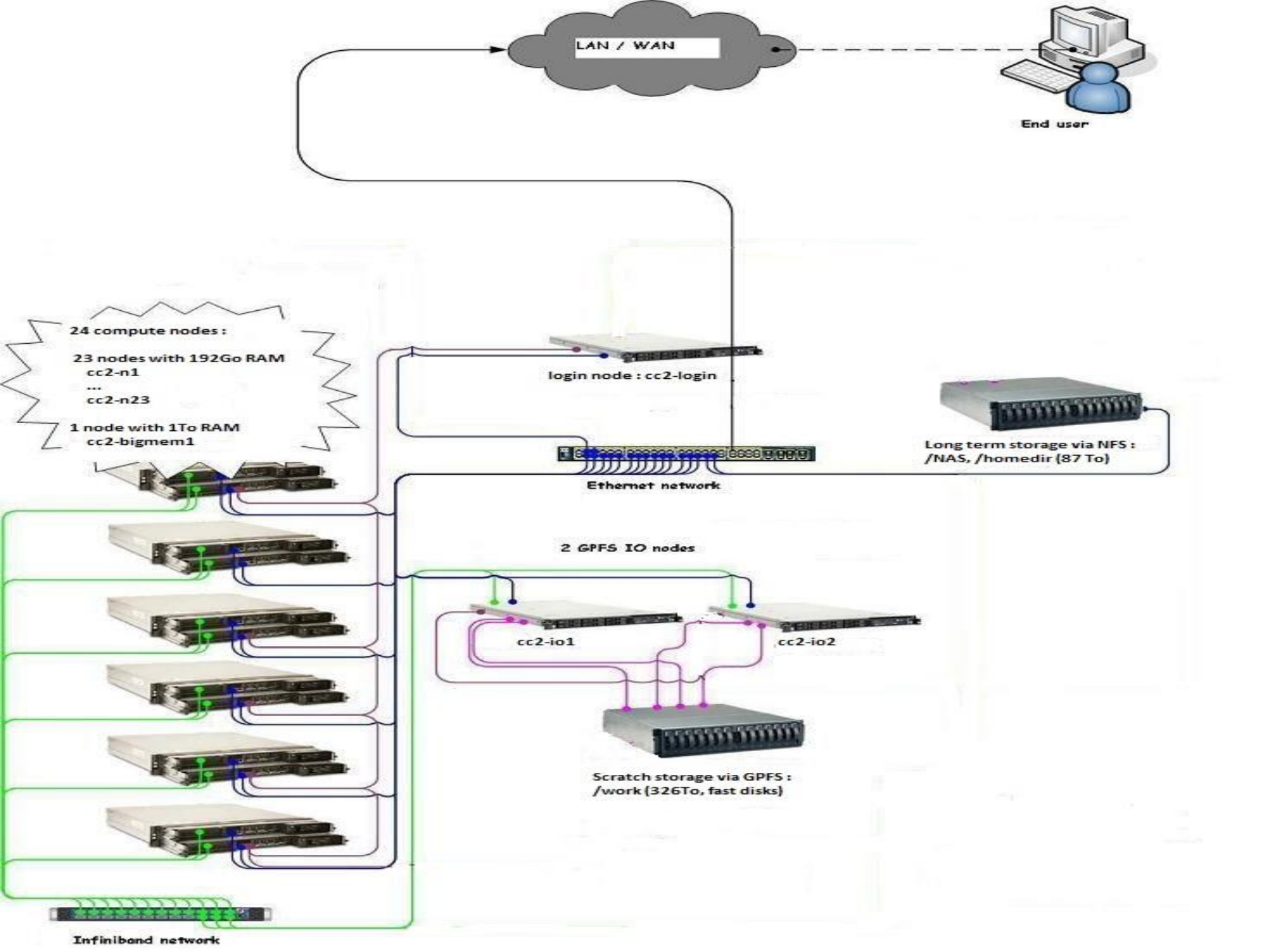
Intérêt

- Gestion des ressources (processeurs, mémoire, scratch, licences, etc.)
- Mise en place des politiques d'utilisation (fairshare, restrictions)
- Gestion des utilisateurs
- Gestion de la coopération
- Présenter une seule machine aux utilisateurs

- *CIRAD:*
  - Gestion des serveurs (processeurs)
  - Gestion de la mémoire
  - Gestion de la coopération et de l'accès concurrent

Rôles des machines

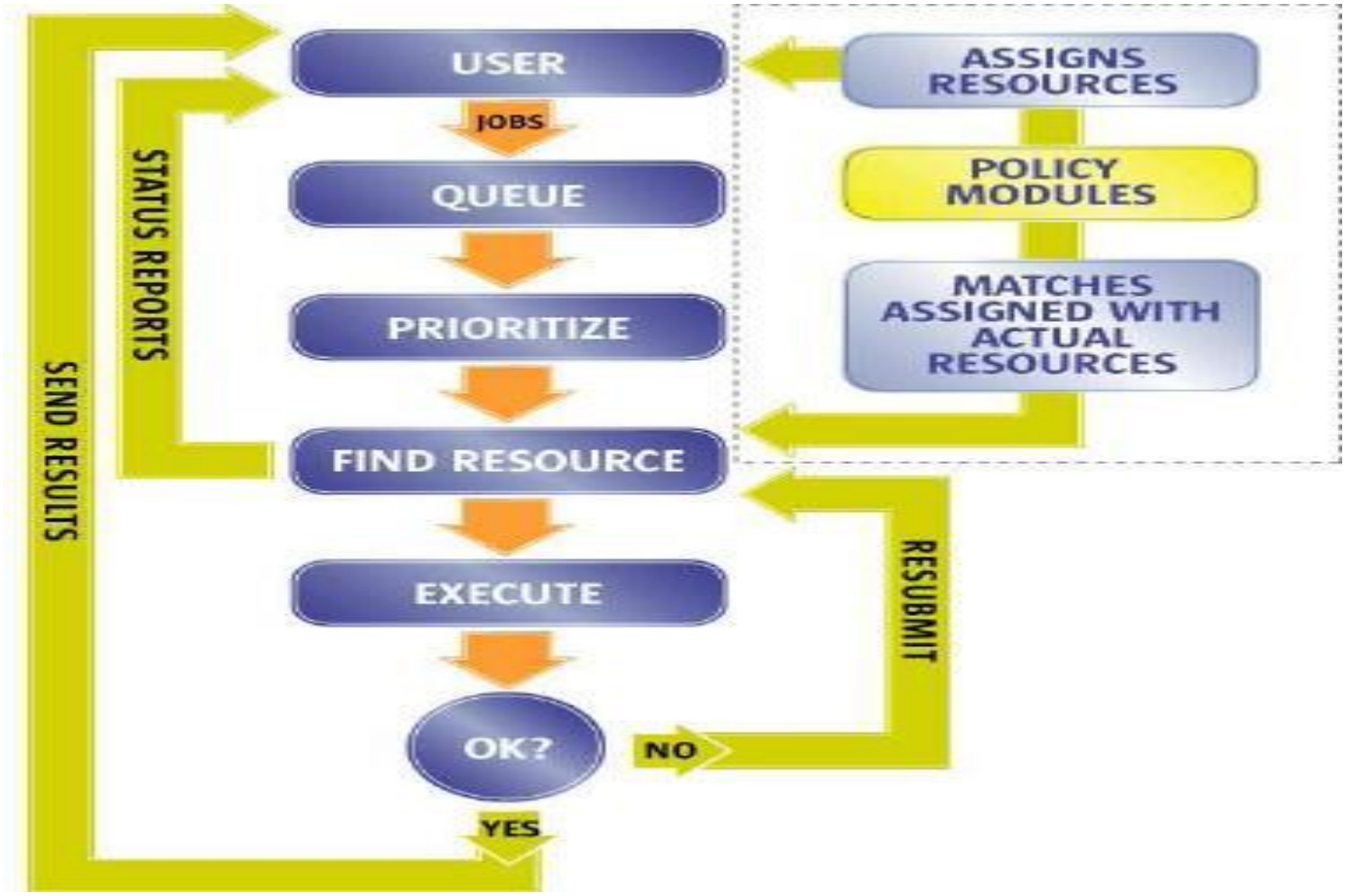
- Administration : gestion sge, scheduler
  - CIRAD : ***cc2-admin***
- Soumission : permettre de soumettre des jobs
  - CIRAD : ***cc2-login***
- Exécution : tourner les jobs
  - CIRAD :
    - ***192Go mémoire : cc2-n1, … , cc2-n23***
    - ***1To mémoire : cc2-bigmem1***


LAN / WAN
End user
24 compute nodes :
23 nodes with 192Go RAM
cc2-n1
...
cc2-n23
1 node with 1To RAM
cc2-bigmem1
login node : cc2-login
Long term storage via NFS :
/NAS, /homedir (87 To)
Ethernet network
2 GPFS IO nodes
cc2-io1
cc2-io2
Scratch storage via GPFS :
/work (326To, fast disks)
Infiniband network

Ecosystème SGE

- SGE est une collection d'objets:
  - Configuration globale (master et SGE)
  - Scheduler
  - *Files d'attente*
  - *Utilisateurs (Projets, départements, listes d'accès)*
  - *Jobs*
  - *Calendriers*
  - Environnements spéciaux (*parallèles*, checkpoint)
  - Contraintes (quotas génériques)
  - Hôtes (administration, exécution, soumission)
  - *Ressources*
- ***… et leurs relations !!!***

Scheduler et jobs



Démarche

- Connexion sur *cc2-login*
- Préparation du calcul
  - Mettre en place les données d'entrée
  - Adaptation/Création des scripts
- Soumission du job
  - Quel type de job: séquentiel, parallèle, tableau de tâches
  - Quelle file d'attente
  - Quelles ressources (mémoire)
- Consultation de l'état du job
- Fin du job: récupération des résultats et analyse

Concepts SGE

Jobs,
Files,
Projets,
Environnements parallèles

Les jobs (I)

Un job est un script qui encapsule une tâche de calcul.

Plusieurs types de jobs:

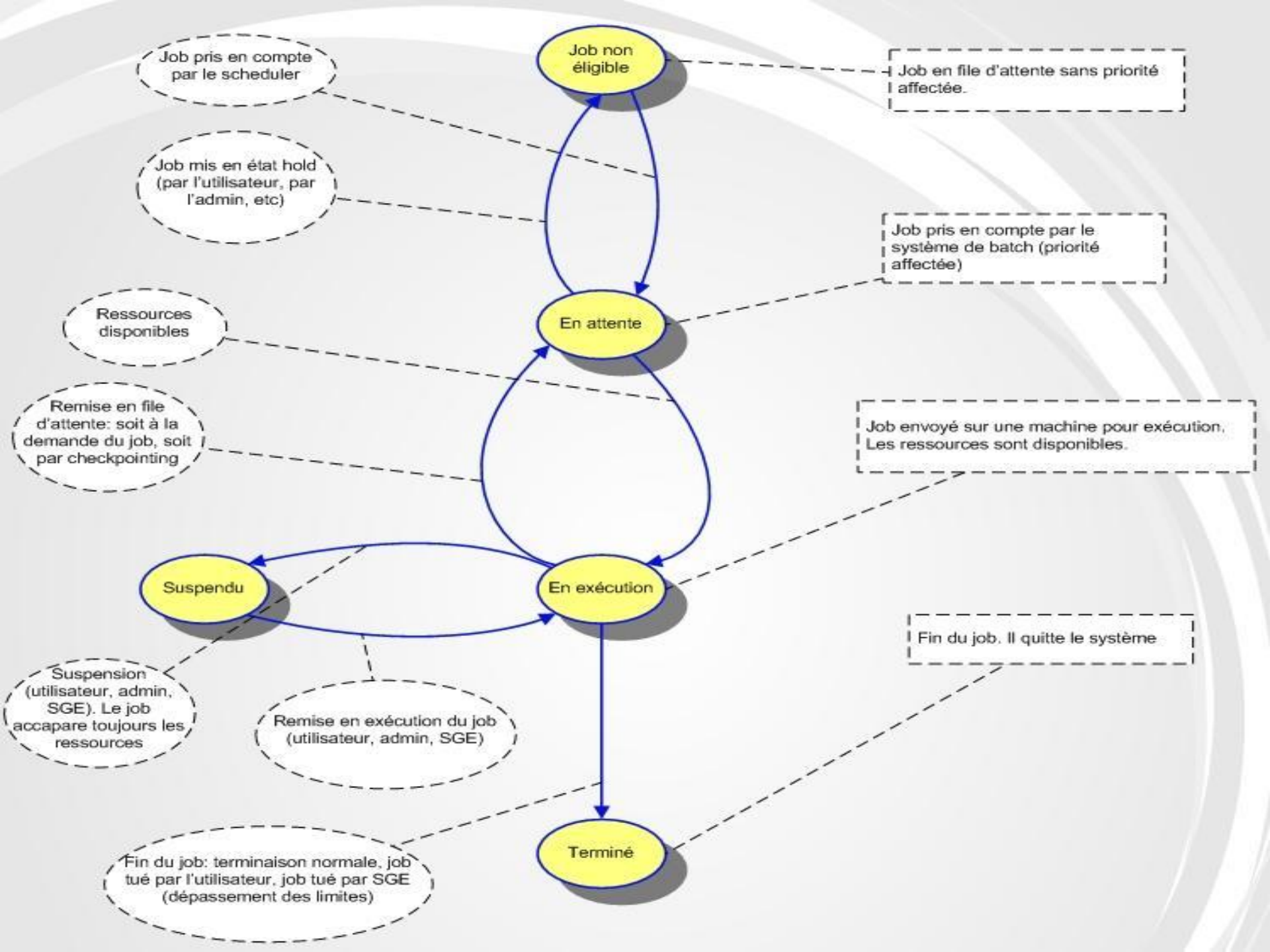
- Batch (***qsub***)
- Interactif (***qrsh***, équivalent de rsh)
  - qsrh <commande>
  - qrsh
- Xterm (***qsh***, équivalent de xterm)

En fonction du nombre des processeurs:

- Séquentiel (qsub … myscript.sh): pas de parallélisation;
- Parallèle (qsub –pe <env_par> 8 … myscript.sh): parallélisation fine;
- JobArray (qsub –t 2-10:2 … myscript.sh): parallélisation gros grain;

Les jobs (II)

- Caractéristiques d'un job:
  - Un *identificateur unique* fourni par SGE (job id);
  - Un *nom* (qsub –N);
  - Un *répertoire de démarrage* (qsub –cwd, par défaut $HOME)
  - Un *projet* (optionnel, mais utile: qsub –P)
  - Un *nombre de slots et un environnement parallèle* (-pe <pe> <N>, défaut 1)
  - Un *environnement* (au sens Unix):
    - Variables d'environnement positionnées par SGE :
      - $JOB_ID, $PE, $NSLOTS, etc
    - Variables d'environnement transmises par l'utilisateur:
      - *qsub –v VAR1=valeur,VAR2=valeur …*
  - Un *fichier de sortie* (-o output, -e error, -j y)
  - Une *file d'attente* et une (ou plusieurs) *machine*(s)
- Un job passe par plusieurs états dans sa vie …


Job non éligible
Job en file d'attente sans priorité affectée.
Job pris en compte par le scheduler
Job mis en état hold (par l'utilisateur, par l'admin, etc)
Job pris en compte par le système de batch (priorité affectée)
En attente
Ressources disponibles
Remise en file d'attente: soit à la demande du job, soit par checkpointing
Job envoyé sur une machine pour exécution. Les ressources sont disponibles.
Suspendu
En exécution
Fin du job. Il quitte le système
Suspension (utilisateur, admin, SGE). Le job accapare toujours les ressources
Remise en exécution du job (utilisateur, admin, SGE)
Terminé
Fin du job: terminaison normale, job tué par l'utilisateur, job tué par SGE (dépassement des limites)

Workflow d'un job

- ***Soumission du job …*** en précisant les ressources …
- Prise en compte du job par SGE …
- Lancement du job par SGE sur une machine …
  - Exécution du prologue du job …
  - Exécution du script de démarrage de l'env. parallèle
  - ***Exécution du script du job …***
  - Exécution du script d'arrêt de l'env. parallèle …
  - Exécution de l'épilogue …
- Nettoyage au niveau SGE …

Jobs: environnement

- Transmettre une variable :
  - ***qsub –v VAR=value,VAR=value …***
- … ou toutes les variables :
  - ***qsub –V …***
- Certains variables sont définies par SGE :
  - ***NSLOTS, NHOSTS, NQUEUES,***
  - ***JOB_ID, TMP, TMPDIR,***
  - ***SGE_TASK_STEPSIZE, SGE_TASK_FIRST, SGE_TASK_LAST, SGE_TASK_ID***
  - ***RESTARTED, QUEUE, SGE_ACCOUNT, JOB_NAME, SGE_O_WORKDIR***
- … soumettre un job /bin/env … : ***qsub –b y … /bin/env***

Jobs: CIRAD

- ***Par défaut*** (déjà mis en place):
  - -cwd : le répertoire courant (sur la machine de soumission) est le répertoire du démarrage du job (sur la machine d'exécution)
  - -j y : par défaut, les sorties du job (standard et erreur) vont dans le même fichier
  - on demande 4G de mémoire (par processeur utilisé)
  - on demande un seul processeur (job séquentiel)
  - le nom du job est le nom du script (!!!)
  - la sortie du job se trouve dans un fichier *<nom_job>.o<job_id>* dans le répertoire courant
- ***Fortement conseillé:***
  - Fournir *un nom* à votre job
  - Choisir la bonne *file d'attente*
  - Préciser la *quantité mémoire* dont vous avez besoin
  - Estimer, si possible, la durée de votre job (ex.: *-l h_rt=02:00:00*, pour *2h*)
  - Préciser si votre job est *parallèle* …

    … et comment *repartir les processeurs sur les machines*

  (pour un job parallèle de plus de 48 slots, ajouter l'option *–R y* à la soumission)

Les files d'attente

- Une file d'attente:
  - Est un conteneur pour les jobs appartenant à la même catégorie
  - Dispose d'un certain nombre de ressources :
    - Machines, slots, environnements parallèles
  - Impose des limitations:
    - Par utilisateur (liste d'accès), par Projets
    - L'accès aux ressources (licences, temps CPU, mémoire)
- Configuration:
    - **qconf –sql** : visualiser les files d'attente
    - **qconf –sq bigmem.q** : visualiser la file **bigmem.q**

CIRAD

- ***bigmem.q***
  - Pour la machine 1To mémoire
  - Accès: tout utilisateur
  - Runtime: infini
  - *Priorité normale*
- ***normal.q***
  - Pour les machines 192Go mémoire
  - Accès: utilisateurs internes
  - Runtime: maximum 48 heures
  - *Priorité normale*
- ***long.q***
  - Pour les jobs longs (192G mémoire)
  - Accès: tout utilisateur
  - Runtime: infini
  - *Faible priorité*
- ***urgent.q***
  - Pour toutes les machines
  - Accès: limité
  - Runtime: maximum 24 heures
  - *La plus haute priorité*
- ***web.q***
  - Pour les machines 192Go mémoire
  - Accès: utilisateurs web
  - Runtime: maximum 36 heures
  - *Haute priorité*

Files d'attente: états

- Ouverte: accepte des jobs en exécution
- Suspendue: l'ensemble des jobs sont suspendus
- Fermée: on n'accepte pas des jobs en exécution mais on laisse finir ceux qui tournent déjà
- Alerte: un paramètre (ex.: charge cpu) a dépassé la limite
- Erreur: un problème (grave) est survenu

Commandes:

- Ouverture, fermeture: ***qmod –d <queue>***, ***qmod –e <queue>***
- Suspension: ***qmod –s <queue>***
- Erreur: ***qmod –cq <queue>***

- Visualisation : **qstat -f**

```
queuename                      qtype used/tot. load_avg arch          states
---------------------------------------------------------------------------------
all.q@anda001                  BIPC  0/2       0.00     lx24-amd64    S
---------------------------------------------------------------------------------
all.q@anda002                  BIPC  0/2       0.00     lx24-amd64    A
---------------------------------------------------------------------------------
liebherr.q@anda001             BIPC  2/2       0.00     lx24-amd64
   2025 0.56000 J_N_1      preda        r     09/27/2007 17:30:20     1
   2028 0.55250 J_N_4      preda        t     09/27/2007 17:30:20     1
---------------------------------------------------------------------------------
liebherr.q@anda002             BIPC  2/2       0.00     lx24-amd64
   2026 0.55500 J_N_2      preda        r     09/27/2007 17:30:20     1
   2027 0.55333 J_N_3      preda        r     09/27/2007 17:30:20     1
---------------------------------------------------------------------------------
liebherr_night.q@anda002       BIP   0/2       0.00     lx24-amd64    Cd
---------------------------------------------------------------------------------
liebherr_p.q@anda002           BIPC  0/2       0.00     lx24-amd64    d
---------------------------------------------------------------------------------
liebherr_u.q@anda002           BIPC  0/2       0.00     lx24-amd64
#################################################################################
 - PENDING JOBS - PENDING JOBS - PENDING JOBS - PENDING JOBS - PENDING JOBS
#################################################################################
   2029 0.55200 J_N_5      preda        qw    09/27/2007 17:30:18     1
```

Ressources

- Objet qui traduit en langage SGE la notion de ressource du monde réel
- Caractéristiques (Attributs):
  - Nom : nom de la ressource
  - Type : MEMORY, TIME, INT, BOOL, DOUBLE, STRING
- Les ressources s'associent:
  - Au niveau du cluster : ressources globales
    - **Exemple:** les licences d'une application App
  - Au niveau d'une machine : ressources per hôte
    - **Exemple:** cette machine a 96 Go mémoire
  - Au niveau d'une file d'attente (restriction)
    - **Exemple:** dans la file web.q je n'ai pas de licence pour l'application App
- ***CIRAD:***
  - La mémoire: *qsub –l mem_free=24G …*
  - Le nombre de processeurs (*voir section env. parallèle*)
  - Restriction Run Time: *qsub –l h_rt=00:00:30 …*

Ressources: commandes utiles

- Visualiser les ressources définies dans SGE
  - ***qconf –sc***
- Quelle est l'état des ressources:
  - ***qhost –F***
  - ***qhost –F mem_free***
- Quelles ressources pour mon job:
  - ***qstat –r***
- Je veux la ressource …
  - ***qsub –l <nom>=<valeur>***
- J'aimerais la ressource …
  - ***qsub –soft –l <nom>=<valeur>***

Exemples

- Soummission :

```
qsub -q normal.q -N JOB_TEST -l lic_abaqus=5 ./sleeper.sh 60
```

- Visualisation des ressources :

```
[preda@anda priority]$ qstat -r
job-ID  prior   name       user         state submit/start at     queue
---------------------------------------------------------------------------------
   2032 0.56000 JOB_TEST   preda        r     09/27/2007 19:16:23 normal.q@s-fr-ap-prod-01
        Full jobname:     JOB_TEST
        Master Queue:     normal.q@s-fr-ap-prod-01
        Hard Resources:   lic_abaqus=5 (0.000000)
        Soft Resources:
        Hard requested queues: normal.q
```

- L'état des ressources :

```
HOSTNAME                ARCH         NCPU  LOAD  MEMTOT  MEMUSE  SWAPTO  SWAPUS
----------------------------------------------------------------------------------
global                  -               -     -       -       -       -       -
   gc:licence_a=4.000000
   gc:licence_b=2.000000
   gf:suspend=0.000000
anda001                 lx24-amd64      2  0.01    2.0G   58.2M    2.0G  144.0K
   gc:licence_a=4.000000
   gc:licence_b=2.000000
   gf:suspend=0.000000
   hl:arch=lx24-amd64
   hl:num_proc=2.000000
   hl:mem_total=1.961G
...
```

- La définition des ressources :

```
[preda@anda priority]$ qconf -sc
#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
display_win_gui     dwg        BOOL        ==    YES         NO         0        0
...
```

Environnements parallèles (I)

- Idée: préparer un job parallèle :
  - En exécutant un script de mise en place (juste avant le job)
  - En exécutant un script de nettoyage (juste après le job)
  - En spécifiant le job parallèle en terme de : politique d'allocation des slots, ACL
- Consultation:
  - qconf –spl, qconf –sp <parenv>
- Utilisation:
  - qsub –pe <parenv> <no. Slots> … myscript.sh
- CIRAD:
  - pour la soumission des jobs parallèles
  - pour préciser comment les slots seront repartis sur les machines

Environnement parallèles (II)

- Politiques d'allocation des slots (allocation_rule):
  - Tous les slots sur la même machine ($pe_slots)
  - N par N (N)
  - Round robin ($round_robin)
  - Remplir les machines, une par une ($fill_up)
- Attention aux pièges :
  - -pe parallel_8 4 : ne va jamais tourner …
  - parallel_rr est différent de parallel_1 !!!
- CIRAD:
  - *parallel_1, parallel_2, parallel_4, parallel_6,parallel_8,parallel_10, parallel_12,…, parallel_40, parallel_42, parallel_44, parallel_46, parallel_48,*
    - N par N avec N=1,2,4,6,8,10,12, … , 40,42,44,46,48
  - *parallel_rr*
    - Allocation de type Round Robin
  - *parallel_fill*
    - Allocation de type remplissage des machines
  - *parallel_smp*
    - Allocation de tous les slots sur la même machine

En pratique …

- **SMP: Shared Memory (OpenMP, pthreads)**
  - **Multi-thread**: plusieurs threads sur la même machine
  - Communication des threads à travers la mémoire partagée
  - **SGE**:
    - **chaque thread utilisera un slot**
    - **utilisez l'environnement parallel_smp**
    - **pas plus de 48 threads (slots)**
- **DMP: Distributed memory (MPI, PVM)**
  - Plusieurs processus sur la même machine ou sur des machines différentes
  - Communication des processus: réseau (cas inter-machine), mémoire partagée (cas intra-machine)
  - **SGE**:
    - **Chaque processus utilisera un slot**
    - **A vous de choisir l'environnement parallèle**
- **Mixte (DMP): MPI+OpenMP**
  - Plusieurs processus (sur des machines différentes), chaque processus ayant plusieurs threads
  - **SGE**:
    - **Nombre de slots: No. Processus x No. Threads**
    - **Environnement parallèle: parallel_N , avec N=nombre de threads**

Les classiques

- **OpenMP**
  - *export OMP_NUM_THREADS=N*
  - *./my_prog* => lancement de N threads
- **Pthreads**
  - Voir la doc de votre logiciel
  - *./my_prog <paramètres>*
- **MPI (OpenMPI)**
  - Hors SGE:
    - *mpirun –np <N> -machinefile <fichier.txt> ./my_prog <paramètres>*
  - SGE+OpenMPI:
    - *mpirun ./my_prog <paramètres>*

… consulter la documentation de votre logiciel …

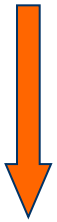
OpenMPI

- CIRAD:
  - Développement (hors SGE)
    - *module load mpi/openmpi/1.6.5*
    - *mpirun –np 2 my_prog*
  - Nœuds (avec SGE, dans un job)
    - *source /etc/profile.d/modules.sh*
    - *module load mpi/openmpi/1.6.5*
    - *mpirun my_prog*

    Car openmpi communique avec SGE pour connaître*:*
    - Le nombre de processeurs
    - Les machines qu'il doit utiliser

Priorités

- *CIRAD*
  - *La priorité d'un job est donnée principalement par la file dans laquelle le job a été soumis*
    - *long.q*
    - *normal.q, bigmem.q*
    - *web.q*
    - *urgent.q*



La priorité croît

  - *Fair-share utilisateur:*
    - *Les jobs des utilisateurs ayant utilisé le moins le cluster son prioritaires (aspect temporel)*
    - *Les jobs des différents utilisateurs soumis dans la même file d'attente s'entrelacent (aspect spatial)*
  - *Les jobs parallèles d'un utilisateur sont plus prioritaires que les jobs séquentiels (pas toujours)*
  - *Les jobs d'un utilisateur passent dans l'ordre FIFO*

Projets

Idée

- Regrouper les jobs d'un point de vue fonctionnel

Intérêt

- Gérer les priorités par projet
- Limiter l'accès (utilisateurs, ressources)
- Retrouver les statistiques

Commandes (consultation)

- **qconf –sprjl, qconf –sprj <projet>**

Utilisation

- Option *–P* de la commande *qsub*

Note

- Seul l'administrateur SGE peut définir des projets

Quotas

- Contraintes génériques évaluées par le scheduler
  - Si la contrainte est violée le job n'est pas éligible à l'exécution
- Commandes
  - qconf –srqsl
  - qquota : pour voir l'état des restrictions
  - qstat –j <job_id>
- CIRAD:
  - Une machines déclarée dans plusieurs files d'attente n'acceptera pas plus de 48 tâches
  - Chaque file d'attente a une limitation quant au nombre total de processeurs
  - La file **web.q** doit toujours disposer de 32 slots

Soumission des jobs

La commande ***qsub*** …

qsub & Co.

- Deux façons de préciser les paramètres :
  - En ligne de commande : qsub <options> (qsub -cwd)
  - Dans le script (job) : #$ <options> (qsub –cwd)
- Piège :

  MY_PROG=prog.sh

  #$MY_PROG …
  - => erreur incompréhensible !!!
- Vous pouvez préciser des paramètres pour le job :
  - ***qsub … my_job.sh fichier_input.dat***

Les indispensables …

- *-q <queue>* : préciser une file d'attente :
  - Le classique
    - qsub –q normal.q …
  - Choisir une machine
    - qsub –q long.q@cc2-n10 …
  - Choisir une machine faisant partie d'un groupe
    - qsub –q urgent.q@@bigmem …
- *-N <nom>* : préciser le nom du job:
  - qsub –N mon_job …
    - génère un fichier de sortie mon_job.o<job_id>
- *-P <projet>* : préciser un projet :
  - qsub –P p1n2 …
    - *CIRAD: pas encore utilisé …*

Environnement

- *-cwd* : démarrer dans le répertoire curant
  - *CIRAD: ceci est le défaut …*
- *-e <fichier>* : fichier de sortie pour les erreurs
- *-o <fichier>* : fichier de sortie standard
  - *CIRAD: le défaut est le <nom du job>.o<job_id>*
- *-j y* : utiliser le même fichier de sortie
  - *CIRAD: ceci est le défaut*
- *-i <fichier>* : fichier en entrée du job
  - *CIRAD: défaut /dev/null*
- *-v <var>=<valeur>* : transmettre une variable
  - qsub **–v VAR1=toto,VAR2=autre** …
- *-V* : transmettre toutes les variables
  - *Un peu barbare, mais ca peut-être utile dans un workflow de jobs*

Contrôle

- *-h* : rendre le job inéligible (annuler avec *qalter –h U*)
- *-hold_jid <liste_jobs>* : dépendance des jobs

- *-terse* : afficher le JOB ID
  - Pour simplifier le workflow des jobs
- *-sync y* : attendre la fin du job
  - On peut réaliser des dépendances des jobs
- *-now y* : essayer d'exécuter immédiatement
  - Si manque de ressources => pas d'exécution
- -r y : redémarrer le job en cas de crash de la machine

Date, parallélisation

Date :

- *-a <date>* : éligible à partir d'une date …
- *-dl <date>* : essayer à tout prix de l'exécuter avant la date (désactivé)

Parallélisation:

- *-pe <pe> N* : exécuter un job parallèle avec l'environnement <pe>
- *-t n-m:s* : exécuter un job array en partant de l'indice n (***SGE_TASK_FIRST***) jusqu'à l'indice m (***SGE_TASK_LAST***) avec un pas de s (***SGE_TASK_STEP***)
    - L'indice est fourni dans l'env. du job à travers ***SGE_TASK_ID***

Ressources

Demander une ressource (*-l nom=valeur*):

- qsub –l lic_A=2,lic_B=1
- Attention!: interprétation per slot …

Utiliser les ressources comme limitations *(-l h_...=valeur*):

- -l h_rt=0:0:40 => tuer le job après 40 secondes …

Je veux, par défaut (*-hard*):

- -hard –l bm=1 : je veux des machines à 92Go (bigmem)

J'aimerais, mais … (*-soft*):

- -soft –l bm=1 : j'aimerais des machines à 92Go (bigmem)

Réserver les ressources (*-R y*):

- qsub –R y –pe parallel_8 80 …

Commandes

Vue d’ensemble …

Commandes usuelles

Soumission:

- **qsub** : soumission des jobs batch
- **qrsh** : soumission des jobs interactifs
- **qsh** : demande de session interactive
- **qalter** : modification des jobs soumis

Contrôle des jobs:

- **qstat:** consultation des jobs et de leurs état
- **qmod**: changement d'état (suspension)
- **qdel**: suppression
- **qhold, qrls**: mettre le job en attente sans possibilité d'exécution

Commandes (suite)

Accounting:

- **qacct**: accounting SGE

Etats des files et des jobs:

- **qstat**: consultation (jobs, files d'attente)

Etat des ressources:

- **qhost:** visualisation

Accounting

Ou historique …

qacct

Une seule commande : qacct

…et plusieurs façons de sélectionner les infos :

- *-j [jobid]* : pour un job
- *-P [projet]* : par projet
- *-q [file] :* par file
- *-m [machine]:* par machine
- *-o [user]* : par utilisateur
- … etc …

Exemples

Quelques templates …

Job simple (I)

- Le job ***sleeper.sh*** (voir slide suivant) exécute un simple sleep.
- Soumission:
  - simple
    - *qsub ./sleeper.sh*
  - soyons plus précis sur la mémoire:
    - *qsub –l mem_free=500M ./sleeper.sh*
  - Changer la file d'attente:
    - *qsub –q bioinfo.q ./sleeper.sh*
- Consultation:
  - Vue rapide de mon job:
    - *qstat*
  - Ressources demandées par mon job:
    - *qstat –r*
  - Informations complètes (scheduling):
    - *qstat –j <job_id>*
- Statistiques du job:
    - *qacct –j <job_id>*

Job simple (II)

```
#!/bin/bash
#

# Infos SGE :

#$ -N sleeper
#$ -q formation.q


time=60
do_echo=1
if [ $# -ge 1 ]; then
   time=$1
fi
if [ $# -ge 2 ]; then
   do_echo=$2
fi

if [ $do_echo != '0' ]; then
   /bin/echo Here I am. Sleeping now at: `date`
fi

sleep $time

if [ $do_echo != '0' ]; then
   echo Now it is: `date`
fi

exit 0
```

Job multi-threads (I)

- Le job suivant (slide suivant) exécute un programme multi-thread
- Le programme demande le format:
  - ***./cpu_threads <nombre threads> <durée du travail>***
- Soumission
  - Simple (le job restera en file d'attente car il demande *12Go x 8=96Go* de mémoire sur une seule machine)
    - ***qsub ./job_threads.sh***
  - En précisant une valeur réaliste pour la mémoire:
    - ***qsub –l mem_free=4G ./job_threads.sh***
- Consultation
  - ***qstat***
  - Raison de l'attente:
    - ***qstat –j <job_id>***
  - Où tourne le job:
    - ***qstat –g t***
- Statistiques
  - ***qstat –j <job_id>***
  - On note que le temps CPU est égale à ***N*** x ***temps elapsed***

Job multi-threads (II)

```
#!/bin/bash

# Infos SGE ...

#$ -q formation.q
#$ -pe parallel_smp 8
#$ -N threads
#$ -l mem_free=12G

# la variable NSLOTS contient le no. de slots alloues
nthreads=$NSLOTS
my_time=60

echo "Starting $nthreads on host $HOSTNAME at $( date )"

# lancement en respectant la ligne de cmde de l'application
./cpu_threads $nthreads ${my_time}

echo "Threads ended at $( date )"

exit 0
```

Job MPI (I)

- Le programme MPI *mpi_pi* utilise calcule la valeur de π
- Chargement de l'environnement OpenMPI
  - *module load mpi/openmpi/1.6.5*
- Compilation
  - *mpicc –Wall –o mpi_pi cpi.c*
- Exécution sur le nœud de dev:
  - *mpirun –np 4 ./mpi_pi*
- Soumission des jobs:
  - *qsub ./mpi_job.sh* (voir listing, env. parallèle SMP à 4 slots)
  - *qsub -pe parallel_8 16 ./mpi_job.sh* (8 slots per machine, 2 machines)
  - *qsub -pe parallel_8 48 ./mpi_job.sh* (8 slots per machine, 6 machines)
- Statistiques:
  - *qacct –j <job_id>*
  - on note que chaque information apparaît autant de fois que le nombre de machines

Job MPI (II)

```
#!/bin/bash

# SGE infos

#$ -q formation.q
#$ -N compute_pi
#$ -l mem_free=500M
#$ -pe parallel_smp 4


# Load module environment
. /etc/profile.d/modules.sh


# Load mpi environment
module load compiler/gcc
module load mpi/openmpi-1.3.4


# Maintenant trouve la commande mpirun dans le PATH.

# Openmpi s'occupe de la communication avec sge ...
# ... pour recuperer le no. de procs et ...
# ... la liste des machines a utiliser.
mpirun ./mpi_pi

exit 0
```