Utilisation du cluster de calcul CIRAD (southgreen)

Nom du cluster

cc2-login.cirad.fr
Système de gestion informatique des ressources du cluster : SGE (Sun Grid Engine)

Votre compte

Pour vous creer un compte, utilisez le formulaire à cette adresse:

http://gohelle.cirad.fr/cluster/compte_cluster.php

Votre login et un mot de passe vous seront communiqué par l'administrateur du cluster.

Connexion

Par SSH (Secure Shell)

Depuis un terminal mac/linux:

```
>ssh login@cc2-login.cirad.fr
```

Puis mot de passe / retour chariot

Depuis windows:

Utiliser un client SSH comme MobaXterm ou Putty.
Configurer la connexion avec l'adresse cc2-login.cirad.fr + login + mot de passe

Attention, en dehors du cirad, il faut passer par un VPN (virtual private network). Voir avec la DSI, infos à cette adresse:
http://intranet-dsi.cirad.fr/reseau-et-securite/acces-au-reseau/acces-depuis-l-exterieur

Après la connexion

Vous êtes connecté sur le noeud maitre du cluster, positionné dans votre home.
La commande

```
>hostname
```

Vous indique sur quel noeud de calcul du cluster vous êtes connecté (cc2-master)
La commande

```
>cwd
```

Vous indique où vous êtes dans l'arborescence.
Votre repertoire contient par defaut les dossiers:

/home/login/save

Pour stocker vos données (200 Mo)

/home/login/work

Pour executer vos programmes et générer les fichiers intermédiaires et les résultats.
Stockage "illimité" mais pas plus de 3 semaines.

Le noeud maitre (cc2-master) n'est pas destiné à lancer des programmes.
Vous devez uniquement l'utiliser pour "reclamer" des ressources au cluster.

Reclamer des ressources:

Les ressources du cluster sont attribuées grâce un système de files d'attentes (queue).
La commande

```
>qconf -sql
```

vous indique le nom des files d'attentes:

`bigmem.q`
: Pour les jobs qui demandent beaucoup de mémoire vive (ex: Assemblage)

`long.q`
: Pour les jobs qui durent plus d'une journée (ex: Mapping d'un gros jeu de données, blast)

`normal.q`
: Pour les jobs qui dure moins d'une journée (a privilégier).

Vous devez toujours préciser une file d'attente lorsque vous réservez des ressources du cluster.

Pour avoir des infos sur le niveau d'utilisation de chaque queue:

```
>qstat -g c
CLUSTER QUEUE                   CQLOAD   USED    RES  AVAIL  TOTAL aoACDS  cdsuE
--------------------------------------------------------------------------------
arcad.q                           0.78      0      0   1104   1104      0      0
bigmem.q                          0.42     44      0     32     80      0     32
gnpannot.q                        0.78      0      0   1104   1104      0      0
greenphyl.q                       0.78      0      0   1104   1104      0      0
long.q                            0.78    574      0    530   1104      0      0
normal.q                          0.78    294      0    810   1104      0      0
urgent.q                          0.78      0      0   1104   1104      0      0
web.q                             0.78      0      0   1104   1104      0      0
```

Pour connaitre le taux d'occupation de chaque noeud de calcul sur le cluster:

```
>qhost
HOSTNAME                ARCH         NCPU NSOC NCOR NTHR  LOAD  MEMTOT  MEMUSE  SWAPTO
SWAPUS
----------------------------------------------------------------------------------------
---
global                  -               -    -    -    -     -       -       -       -
-
cc2-bigmem1             lx-amd64       32    4   32   32  1.00 1009.7G  193.4G    4.0G
118.5M
cc2-bigmem2             lx-amd64       96    4   48   96 65.55    1.4T   70.6G    4.0G
0.0
cc2-n1                  lx-amd64       48    2   24   48 35.92  188.9G   31.6G    4.0G
204.1M
cc2-n10                 lx-amd64       48    2   24   48 26.79  188.9G   10.9G    4.0G
30.4M
cc2-n11                 lx-amd64       48    2   24   48 39.68  188.9G   18.3G    4.0G
67.3M
cc2-n12                 lx-amd64       48    2   24   48 41.33  188.9G   18.2G    4.0G
2.7G
cc2-n13                 lx-amd64       48    2   24   48 46.83  188.9G   54.4G    4.0G
389.3M
cc2-n14                 lx-amd64       48    2   24   48 47.97  188.9G   43.2G    4.0G
409.4M
cc2-n15                 lx-amd64       48    2   24   48 26.03  188.9G   11.6G    4.0G
227.1M
cc2-n16                 lx-amd64       48    2   24   48 50.98  188.9G   37.6G    4.0G
607.0M
cc2-n17                 lx-amd64       48    2   24   48 36.56  188.9G   26.0G    4.0G
67.9M
...
```

Se connecter directement sur un noeud avec le commande qrsh:

Vous serez alors loggé sur le un noeud de calcul. Vous pouvez y travailler sans gener les autres utilisateurs.

Exemple:

```
>qrsh -q normal.q
>hostname
>monprogramme mon_fichier.txt > ma_sortie.txt
```

Soumettre un job avec la commande qsub

Vous demandez l'execution d'un script sur un noeud de calcul, sans gener les autres utilisateurs.

Un script est un fichier texte qui contient les commandes que vous voulez executer. Le cluster doit comprendre ce que vous lui demandez. Pour cela précisez au minimum.
1/ Votre shell (bash en général)

```
#!/bin/bash
```

2/ La file d'attente

```
#$ -q normal.q
```

3/ Le lieu d'execution (ici le repertoire courant, dans lequel vous êtes en train de travailler.

```
#$ -cwd
```

4/ Le nom de votre job

```
#$ -N monsuperjob
```

5/ Le programme que vous voulez lancez.

```
monprogramme -options fichier_a_traiter.txt > monfichier_resultats.txt
```

Attention, il fait que "monprogramme" soit accessible depuis votre repertoire courant, qui doit également contenir "fichier_a_traiter.txt"

Localiser un programme / Le rendre accessible depuis n'importe quel repertoire

L'ensemble des repertoires dans lesquels des programmes sont accessibles sont repertoriés dans la variable PATH.
La commande

```
>echo $PATH
```

Retourne la liste de ces repertoires. Vous pouvez modifier cette liste en ajoutant un par exemple le avec la commande

```
>export PATH=$PATH:/chemin/repertoire/
```

Ajouter cette commande a votre .bashrc pour rendre cette modification permanente.
Si un programme est accessible (il est dans le ”PATH”), vous pouvez savoir où grace la commande which.

```
>which monprogramme
```

retourne le répertoire dans lequel “monprogramme” est installé.

Un grand nombre des programmes installés sur le cluster sont à charger par le biais de la commande **module**. Ceci afin de pouvoir installer plusieurs version d’un même programme. Par exemple la commande:

```
>module load bioinfo/samtools/1.3
```

Permet de rendre accessible le version 1.3 de la suite de programme samtools.

Ecrire le script

Directement sur le cluster ! (vous pouvez aussi l’écrire sur votre ordinateur puis le copier sur le cluster, mais c’est une perte de temps).
Apprenez à utiliser un editeur en ligne de commande.
Par exemple le programme nano.

```
>nano monscript.sh
```

Vous permet d’éditer le fichier monscript.sh dans votre repertoire courant (rappel: la commande cwd vous permet de savoir quel est le repertoire courant).

Taper (ou coller un exemple que vous modifier ensuite):

```
#!/bin/bash
#$ -q normal.q
#$ -cwd
#$ -N monsuperjob
monprogramme -options fichier_a_traiter.txt > monfichier_resultats.txt
```

Enregistrer/quitter (Ctrl X).
Pour demander au cluster d’executer votre job:

```
>qsub monscript.sh
```

SGE s'occuper de distribuer votre job sur un des noeud de calcul. Les sorties et erreurs standards sont redirigé vers les fichiers monsuperjob.oXXXXX et monsuperjob.eXXXXX (avec XXXXX qui correspond au jobID: voir plus bas) que vous trouverez dans le repertoire dans lequel vous avez travaillé.

Reclamer plus de mémoire pour un job

Par défaut, à chaque job soumis par le biais de la commande qsub est attribué 4 Go de mémoire vive.
Si vous avez besoin de plus de mémoire (par exemple 16 Go), ajoutez à votre script l'option:

```
#$ -l mem_free=16G
```

Si vous avez besoin de plus de 180 Go de mémoire vive, utilisez la queue bigmem.q.
La limite sur bigmem.q est d'1,4 To.

Surveiller l'execution de votre job:

```
>qstat
```

Output:

```
job-ID  prior   name       user         state submit/start at     queue
slots ja-task-ID
-----------------------------------------------------------------------------------------
----------------------
1364382 0.00000 seq        loire        qw    03/20/2016 14:18:55
1
```

Puis

```
job-ID  prior   name       user         state submit/start at     queue
slots ja-task-ID
-----------------------------------------------------------------------------------------
----------------------
1364382 986.55388 seq        loire        r     03/20/2016 14:19:37 normal.q@cc2-n15
1
```

Repérer le job-ID (ici `1364382`) et la colonne "state". Cette derniere vous donne l'état du job:
qw : Queue waiting (Job en attente)
r: Running (en cours d'execution)
Eqw: Problème.

En cas de problème pour obtenir des informations détaillées sur le job, préciser le job-ID dans la commande

```
>qstat -j  1364382
```

Examiner le résultat de cette commande pour identifier le problème.

Si votre job ne démarre pas (“state” reste en qw), les causes possibles sont:

1/ Tous les noeud du cluster sont saturés. Prenez votre mal en patience.

2/ Vous avez reclamé plus de mémoire qu’il n’y en a de disponible. Modifier votre script en conséquence ou utilisez la queue bigmem.q

Liberer la ressource

Vérifier avant de vous déconnecter du cluster que vous n’êtes plus connecté sur un noeud (qrsh)
Pour le savoir tapez
>qstat
Si vous êtes connecté en qrsh sur un noeud, cette commande retournera (entre autre):

```
job-ID  prior   name       user         state submit/start at     queue
slots ja-task-ID
-----------------------------------------------------------------------------------------
----------------------
1364729 986.55387 QRLOGIN    toto         r     03/21/2016 11:19:38 normal.q@cc2-n5
1
```

Ici l’utilisateur “toto” est connecté sur le noeud “cc2-n5”

Si vous êtes connecté sur un noeud pour le libérer depuis le noeud, utilisez la commande:

```
>exit
```

```
>hostname
```

Doit retourner comme info:

```
cc2-admin
```

Si vous avez quitter le cluster sans vous déconnecter du noeud, vous pouvez le quitter avec la commande:

```
>qdel  1364729
```

Ici 1364729 est le job-ID de le session qrsh que vous pouvez retrouver avec la commande qstat

De même, lorsque vous avez fait une erreur lors de la soumission d'un job (qsub), vous pouvez le "tuer" avec la commande:

```
>qdel  1364382
```

Ici, `1364382` est le job-ID que vous pouvez retrouver avec la commande qstat

Parallélisme

Vous pouvez parfois diviser la tâche du programme que vous souhaitez utiliser.

En divisant le fichier à traiter

En cas des données qui peuvent être traité en plusieurs blocs indépendants, vous pouvez diviser un fichier avec la commande split. Par exemple:

```
>split -da -l 4000 gros_fichier_a_diviser.txt
```

Divise le gros fichier en un ensemble de fichiers de 4000 lignes chacun, nommé gros_fichier_a_diviser.txt.part0001

```
>split -da 4 -l $((`wc -l < gros_fichier_a_diviser.txt `/20)) gros_fichier_a_diviser.txt
```

Divise en vingt parties égales (plus un fichier supplémentaire si le compte ne tombe pas juste). Les fichiers résultants seront nommés de gros_fichier_a_diviser.txt.part0001 à gros_fichier_a_diviser.txt.part0021.
A vous ensuite de traiter chaque fichier simultanément.

En utilisant un programme capable d'utiliser plusieurs processeurs

Certains programmes ont une option qui leur permet de mobiliser plusieurs processeurs d'un même noeud en même temps. Il faut alors spécifier au cluster combien de processeurs (plus exactement "threads") vous voulez reservez.
Exemple de script multithread:

```
#!/bin/bash
#$ -N <JOB_NAME>
#$ -q <QUEUE_NAME>
#$ -pe parallel_smp 8
#$ -cwd
blastn -db database -query file.fasta -num_threads 8 -outfmt 6 -out results.blast
```

Ici, l'important est la ligne

```
#$ -pe parallel_smp 8
```

Qui permet de reserver huit threads pour le programme blast (option -num_threads 8);
Vous devez vous assurez vous même qu'il y'a une correspondance entre la valeur passée au programme (ici 8) et la valeur passée à SGE via -pe parallel_smp (8 aussi).

Attention, si vous spécifiez aussi l'option -l mem_free=16G, il est possible que vous réserviez de la mémoire pour chaque thread. Pensez à faire la multiplication pour vous assurez que vous ne réservez pas alors plus de mémoire que disponible (ici 8*16 = 128 Go)

En utilisant un programme capable d'utiliser plusieurs noeuds

Certains programmes particulièrement intensifs utilise des librairies (ex: open-mpi) qui leur permettent de distribuer la tâche sur plusieurs noeud de calcul.
Il faut alors charger les modules qui permettent d'utiliser ces librairies et étudier les paramètres du programme pour être sûr qu'il fonctionnera correctement. A voir au cas pas cas.
Exemple avec d'un script qui lance le programme Abyss (assemblage)

```
#!/bin/bash
#$ -N Abyss-P
#$ -q normal.q
#$ -pe parallel_fill 40
#$ -S /bin/bash
#$ -cwd
#$ -V

# LOAD MPI ENVIRONMENT
module load compiler/gcc
module load mpi/openmpi/1.6.5
# LOAD Program
module load bioinfo/abyss/1.5.2_MPI

abyss-pe k=25 in="reads.fastq"
```

Ici, l'important est

```
#$ -pe parallel_fill 40
```

qui permet de reserver 40 threads sur différents noeud de calcul.
Et

```
module load compiler/gcc
module load mpi/openmpi/1.6.5
```

Qui permet de charger les librairies mpi dont le programme a besoin pour fonctionner.