Cycle de formation Meso@LR

SLURM

SLURM *(Simple Linux Utility for Resource Management)* est une solution open source de gestion de cluster sous Linux. SLURM permet :

- La configuration des grappes de nœuds (droit d'accès, quotas, …) (partie admin).
- L'accès aux ressources par les utilisateurs.
- L'ordonnancement de tâches dans les files d'attentes (arbitrage).
- Accounting

Les commandes de soumission de job

- srun

La commande srun permet d'exécuter plusieurs jobs en parallèles sur les nœuds de calcul.

```
$ srun –n4 –N2 –p fmuse1 hostname
```

```
$ srun –n4 –N2 –p fmuse1 --ntasks-per-node=2 hostname
```

- sbatch

La commande sbatch permet de spécifier les caractéristiques d'un job (ressources, programme à exécuter, …) et de l'exécuter via un fichier de script.

Exemple de fichier de script sbatch :

```
#!/bin/bash
#
#SBATCH -N 2
#SBATCH -n 4
#SBATCH --ntasks-per-node=2
#SBATCH --partition=fmuse1

srun hostname
```

```
$ sbatch mon_fichier.sh
```

Les paramètres sbatch peuvent être renseignés en ligne de commande mais un fichier de shell doit toujours être invoqué.

```
$ sbatch –N 2 –n 4 mon_fichier.sh
```

Bien que l'on retrouve des paramètres semblables pour les deux commandes, srun n'est pas la version en ligne de commande de sbatch.

Les paramètres relatifs aux ressources sont :

- des paramètres d'exécution pour srun.
- des paramètres de réservation pour sbatch.

Dans l'exemple ci-dessus le job occupe quatre cœurs mais sans srun la commande hostname ne sera exécutée qu'une seule fois.

Attention !!! Les ressources réservées sont comptabilisées comme utilisées durant toute la durée du job. Qu'elles soient effectivement utilisées ou pas.

- Principales options sbatch

Remarque : La plupart des options sont également utilisables avec les commandes srun et salloc.

```
-A, --account=<account>
-p, --partition=<partition_names>
-J, --job-name=<jobname>
-o, --output=<filename pattern>
-e, --error=<filename pattern>
--mail-type=<type>
--mail-user=<email_user>
```

type = BEGIN, END, FAIL, ALL, …

```
-n, --ntasks=<number>
-N, --nodes=<minnodes[-maxnodes]>
--ntasks-per-node=<ntasks>
--ntasks-per-core=<ntasks>
--ntasks-per-socket=<ntasks>
-c, --cpus-per-task=<ncpus>
--cores-per-socket=<cores>
--mem=<size[units]>
--mem-per-cpu=<size[units]>

-b, --begin=<time>
-t, --time=<time>
--deadline=<date>
```

Format time / date :

[YYYY-MM-DDT]HH :MM :SS

```
--exclusive
-w, --nodelist=<node name list>
-x, --exclude=<node name list>
```

Remarque : Les options ne sont pas utilisables avec toutes les commandes mais lorsqu'elles le sont, on retrouvera (presque) toujours la même syntaxe.

- scancel

La commande scancel envoie un signal à un job. Généralement elle est utilisée pour stopper des jobs.

```
$ scancel 1206340
$ scancel -u chapuis
$ scancel -t PENDING
```

Un utilisateur ne peut stopper que ses propres jobs.

Les commandes de visualisation d'état

Par défaut les commandes d'état affichent des informations sur les jobs, les partitions et les queues auxquelles vous avez accès. Plusieurs paramètres peuvent être utilisés pour limiter l'affichage par exemple à un utilisateur (-u) ou à une partition (-p).

- squeue : Affiche des informations sur les jobs en cours

```
$ squeue -u chapuis
```

Les principaux états pour un job sont :

- RUNNING (R) : En cours d'exécution
- PENDING (PD) : En attente de ressource
- COMPLETED (CD) : Terminé
- CANCELED (CA) : Tué
- FAILED (F) : Echec

- sinfo : Affiche des informations sur les nœuds et les partitions

```
$ sinfo -p fmuse1
```

La réservation de ressource avec `salloc`

La commande salloc permet de réserver des ressources. Sur le cluster muse la connexion ssh sur un nœud de calcul n'est possible que lorsque vous exécutez un job sur le nœud. Cela pose un problème si vous souhaitez travailler sur un programme en ligne de commande sur un nœud.

La commande salloc permet de réserver des ressources auxquelles on peut ensuite accéder en ssh.

```
$ salloc –w muse078 -p fmuse1 –t 1:30:00
```

```
$ ssh muse078
```

Remarque : Les paramètres sont optionnels. S'ils sont omis les valeurs par défaut sont utilisées.

L'utilisation des ressources allouées ne se limitent pas à ssh. Elles peuvent être utilisées par l'utilisateur pour lancer des jobs. La commande salloc agit comme une nouvelle session shell. L'utilisateur met fin à la réservation de ressource soit en programmant un walltime, soit avec la commande exit.

Attention !!! Les ressources allouées sont comptabilisées comme utilisées durant toute la durée de l'allocation.

Remarque : Il est également possible de créer une session shell avec la commande srun.

```
$ srun --pty bash
```

Contrairement à la commande salloc, la commande srun vous connecte directement au nœud de calcul.

Les variables d'environnement SLURM

- SLURM_JOB_ID : Numéro du job
- SLURM_NODELIST : Nœuds alloués au job
- SLURM_JOB_NAME : Nom du job
- SLURM_SUBMIT_DIR : Répertoire courant
- SLURM_SUBMIT_HOST : Machine d'où est lancé le job
- SLURM_JOB_NUM_NODES : Nombre de nœuds
- SLURM_CPUS_PER_TASK : Nombre de CPU par tache
- SLURM_NTASKS : Nombre de taches
- SLURM_JOB_PARTITION : Partition utilisée

Les Job Array

La fonction Job Array permet d'exécuter plusieurs fois un programme identique sur des jeux de données différents. Contrairement au calcul parallèle, chaque processus est indépendant.

On déclare un Job Array dans un fichier sbatch avec l'option --array.

Exemple :

- `# SBATCH --array=0-27`
- `# SBATCH --array=0-59%28`
- `# SBATCH --array=1,2,5-10`
- `# SBATCH --array=0-99:4`
- `# SBATCH --array=1-100%20:2`

```
#!/bin/bash
#
#SBATCH --job-name=test_array
#SBATCH --array=1-1000%56
#SBATCH -o array-%a.out
#SBATCH --partition=fmeso1

module purge
module load cv-standard R
echo $SLURM_NODELIST
R CMD BATCH --no-save prog.R
```

Attention : On ne précise pas le nombre de nœud (-N) et de cœur (-n) que l'on souhaite. C'est SLURM qui va répartir en fonction des ressources disponibles.

Les jobs array disposent de plusieurs variables d'environnement qui peuvent être utilisées pour contrôler les entrée / sortie.

- SLURM_ARRAY_JOB_ID : Numéro du premier job (différent de SLURM_JOB_ID)
- SLURM_ARRAY_TASK_ID : Numéro de la tache
- SLURM_ARRAY_TASK_COUNT : Nombre de tache dan l'array

Pour plus d'information https://slurm.schedmd.com/job_array.html

Les commandes d'Accounting

Plusieurs commandes SLURM permettent d'afficher des informations sur les jobs et sur l'utilisation du cluster. Les deux principales sont sreport et sacct.

Un utilisateur peut voir les informations sur les jobs des partitions et des groupes auquel il appartient.

Il existe beaucoup d'option pour chacune des deux commandes.

Exemple :

- Nombre d'heures par partition sur une période pour un utilisateur

```
$ sreport cluster UserUtilizationByAccount user=chapuis start=2019-10-01 end= 2019-10-16T23:59:59 -t hours
```

- Liste des jobs sur une période pour un utilisateur

```
$ sacct --format="JobID,CPUTime,User,Account,JobName,Start,End,Node" -u chapuis –X -S 2019-10-15 -E 2019-10-16T23:59:59
```

Le problème est dans l'interprétation des résultats.

La commande sreport indique la consommation sur une période.

La commande sacct indique les jobs qui sont ou ont été actifs sur une période.

Le champ CPUTime de la commande sacct indique le temps CPU utilisé par le job depuis son lancement (temps x nombre de cœur) même si une partie du job s'exécute en dehors de la période. Il est donc difficile de faire un parallèle entre le nombre d'heures affiché par les deux commandes.

- Afficher les informations sur un job actif

```
$ scontrol show jobid <job_id>
```

- Afficher les informations sur un job

```
$ sacct --format="JobID,CPUTime,AllocCPUS,User,Account,JobName, Start,End,State%20,Node,ExitCode" -j 1080739
```