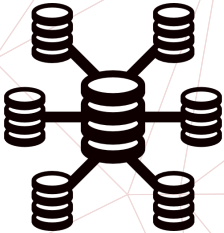


Maxime Chupin
chupin@ceremade.dauphine.fr
www.ceremede.dauphine.fr/~chupin

CNRS, CEREMADE, Université Paris-Dauphine

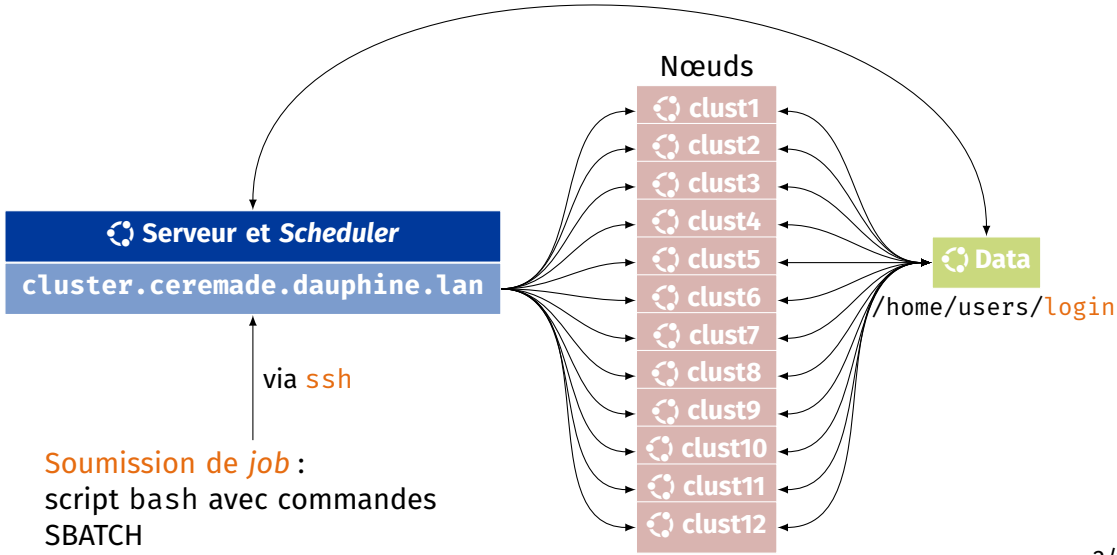
13 juin 2023

Cluster de calcul du CEREMADE

description et utilisation

Description

L'architecture



Le matériel (1)

- clust1 : 40 CPUs, 128Go RAM, GPU (2560 cœurs, 16Go RAM)
- clust2 : 40 CPUs, 128Go RAM, GPU (2560 cœurs,16 Go RAM)
- clust3 : 40 CPUs, 128Go RAM, GPU (2560 cœurs, 8Go RAM)
- clust4 : 40 CPUs, 128Go RAM, GPU (2560 cœurs, 8Go RAM)
- clust5 : 40 CPUs, 128Go RAM, GPU (2560 cœurs, 8Go RAM)
- clust6 : 40 CPUs, 128Go RAM, GPU (2560 cœurs, 8Go RAM)
- clust7 : 40 CPUs, 128Go RAM, GPU (2560 cœurs, 8Go RAM)
- clust8 : 40 CPUs, 128Go RAM, GPU (2560 cœurs, 8Go RAM)
- clust9 : 40 CPUs, 64Go RAM
- clust10 : 40 CPUs, 64Go RAM
- clust11 : 80 CPUs, 192Go RAM, GPU (2304 cœurs, 8Go RAM)
- clust12 : 80 CPUs, 192Go RAM, GPU (2304 cœurs, 8Go RAM)

Attention

Les CPUs sont nombreux, mais ils sont assez « lents ». Variations de *Intel(R) Xeon(R) CPU* entre 2.10 GHz et 2.20 GHz.

Attention

En réalité, « nombre de *CPUs* » : le résultat de *sockets*, *cœurs* et *threads*.

SLURM

- Un tel ensemble de machines → une organisation
- Résultats en termes de temps de calcul → pouvoir utiliser 100% des ressources souhaitées
- Système de gestion des ressources

Définition (SLURM)

SLURM (*Simple Linux Utility for Resource Management*) est une solution *open source* d'ordonnancement de tâches informatiques qui permet de créer des grappes de serveurs.

Utilisation

1 Description

2 Utilisation

3 Gestion des *jobs*

Script SBATCH

- Envoyer les fichiers (sources, scripts, données) sur `cluster`[1]

```
user $> scp -r /home/user/pi/
   login@cluster.ceremade.dauphine.lan:~/
```

et s'y connecter.

- Fabrication[2] d'un fichier SLURM qu'on exécute sur `cluster` via la commande :

```
login@cluster $> sbatch monfichier.SBATCH
```

- Script SLURM est un script bash
- Des commandes SLURM sous forme de commentaires
- Des variables d'environnement

1. sur le serveur de fichiers du laboratoire qui est monté sur `cluster`, sur `www`, sur les `clust`, etc.
2. peut être fait avant la connexion

Début du fichier SBATCH (1)

On définit notre *job* via des commandes SBATCG sous forme de commentaires qui commencent par #SBATCH

```
#!/bin/sh
# Fichier submission.SBATCH
#SBATCH --nodes=1
#SBATCH -c 20
#SBATCH --job-name="MON_JOB"
#SBATCH --output=test.out
#SBATCH --mail-user=chupin@ceremade.dauphine.fr
#SBATCH --mail-type=BEGIN,END,FAIL
```

`-c <n>` est un *raccourcis* pour `--cpus-per-task=<n>`.

Début du fichier SBATCH (2)

Pour les expert·e·s

On demande le nombre de « cœurs » avec l'option `-c 20`, où on a demandé 20 cœurs ici. En réalité, cette commande permet de choisir le nombre de threads total. Pour les utilisateurs et utilisatrices les plus avancé·e·s, on pourra régler tout ceci très finement avec les options suivantes :

```
--sockets-per-node=S   #number of sockets per node to allocate
--cores-per-socket=C   #number of cores per socket to allocate
--threads-per-core=T   #number of threads per core to allocate
```

Quelques exemples (1)

Programme compilé

```
#!/bin/sh
# fichier submission.SBATCH
#SBATCH --nodes=1
#SBATCH -c 10
#SBATCH --job-name="Test_OpenMP"
#SBATCH --partition=erc
#SBATCH --output=%x.%J.out
#SBATCH --error=%x.%J.out
#SBATCH --mail-user=login@ceremade.dauphine.fr
#SBATCH --mail-type=BEGIN,FAIL,END

# on se place dans le répertoire de soumission
cd ${SLURM_SUBMIT_DIR}
# on execute le programme
./compute_pi
```

Quelques exemples (2)

Python

```
#!/bin/sh
# fichier submission.SBATCH
#SBATCH --nodes=1
#SBATCH -c 20
#SBATCH --job-name="Test_Python"
#SBATCH --output=%x.%J.out
#SBATCH --time=10:00
#SBATCH --error=%x.%J.out
#SBATCH --mail-user=login@ceremade.dauphine.fr
#SBATCH --mail-type=BEGIN,FAIL,END

# on se place dans le répertoire de soumission
cd ${SLURM_SUBMIT_DIR}

python3 script.py
```

Matlab

```
#!/bin/sh
#SBATCH --nodes=1
#SBATCH -c 20
#SBATCH --job-name="Test_Matlab"
#SBATCH --output=%x.%J.out
#SBATCH --error=%x.%J.out
#SBATCH --mail-user=login@ceremade.dauphine.fr
#SBATCH --mail-type=BEGIN,FAIL,END

# on se place dans le répertoire de soumission
cd ${SLURM_SUBMIT_DIR}

matlab -nodisplay -nodesktop -r "run('test.m')"
```

Quelques exemples (4)

R

```
#!/bin/bash
#SBATCH --job-name "demo_r"
#SBATCH --cpus-per-task=4
#SBATCH --mail-user="login@ceremade.dauphine.fr"
#SBATCH --ntasks=1
#SBATCH --nodes=1
#SBATCH --mail-type=BEGIN,END,FAIL

# on se place dans le répertoire de soumission

cd ${SLURM_SUBMIT_DIR}

Rscript demo_r2.R
```

Possibilité d'ouvrir une session interactive

Au lieu d'utiliser `sbatch`, on utilisera la commande `srun` :

```
login@cluster $> srun --pty -c 10 -N 1 /bin/bash
```

On arrive alors sur un des nœuds : on passe de `login@cluster` à `login@clust<I>`.

Compilation d'un code

Les nœuds tournent sous Ubuntu 20.04 et les programmes et bibliothèques standards sont normalement installés.
Pour compiler un code, il faut ouvrir une session interactive et compiler sur un nœud.

Session interactive (2)

Jupyter

Pour lancer un *notebook* Jupyter, on utilisera, une fois connecté sur un nœud, la commande suivante

```
login@clust<I> $> jupyter notebook --ip=0.0.0.0
```

La sortie nous donne l'adresse web à copier et mettre dans son navigateur web.

```
   To access the notebook, open this file in a browser:
   file:///mnt/nfs/nrdata02-users-data/chupin/.local/share/jupyter/runtime/nbserver
       -1425187-open.html
Or copy and paste one of these URLs:
   http://clust12:8889/?token=0fe474b1c101733d456a07c83a20d7a51b61b5054914293a
or http://127.0.0.1:8889/?token=0fe474b1c101733d456a07c83a20d7a51b61b5054914293a
```

Rajouter `.ceremade.dauphine.lan` après le `http://clust12`.

Système

- Les nœuds sont installés avec les bibliothèques standard pour le calcul scientifique.
- Pour les langages Python/Julia/R, il faut recourir aux gestionnaires de bibliothèques locaux (Anaconda, pip, Pkg, etc.).

Calcul sur GPU

```
#!/bin/bash
# Fichier submission.SBATCH
#SBATCH --nodes=1
#SBATCH -c 20
#SBATCH --gres=gpu:1
#SBATCH --job-name="MON_JOB"
#SBATCH --output=%x.%J.out
#SBATCH --error=%x.%J.out
#SBATCH --mail-user=<login>@ceremade.dauphine.fr
#SBATCH --mail-type=BEGIN,END,FAIL

# on se place dans le répertoire de Soumission

cd ${SLURM_SUBMIT_DIR}

# Exécution de programme
```

Gestion des *jobs* (1)

Visualiser l'ensemble des jobs

```
login@cluster ctop -t -s 1
```

```
login@cluster squeue
```

Gestion des *jobs* (2)

Information sur un *job*

```
login@cluster sstat 150
```

où 150 est le #JOBID donné à la soumission ou avec `smap`. **Supprimer un job en cours**

```
login@cluster scancel 150
```

Visualisation graphique

Visualisation globale

https://www.ceremade.dauphine.fr/affichage/cluster/cluster.html

Visualisation GPU

https://www.ceremade.dauphine.fr/affichage/cluster/cluster_gpu.html

Options SBATCH (1)

Options	Raccourcis	Description
`#SBATCH --job-name=<name>`	`-J <name>`	Définit le nom du job tel qu'il sera affiché dans les différentes commandes Slurm (squeue, sstat, sacct)
`#SBATCH --output=<stdOutFile>`		La sortie standard (stdOut) sera redirigée vers le fichier défini par `--output` ou, si non définie, un fichier par défaut `slurm-%j.out` (Slurm remplacera `%j` par le JobID).
`#SBATCH --error=<stdErrFile>`		La sortie d'erreur (stdErr) sera redirigée vers le fichier défini par `--error` ou, si non définie, vers la sortie standard.
`#SBATCH --input=<stdInFile>`		L'entrée standard peut aussi être redirigée avec `--input`. Par défaut, `/dev/null` est utilisé (aucune/vide).

Options SBATCH (2)

Options	Raccourcis	Description
`#SBATCH --mail-user=<e-mail>`		Renseigne l'email à qui écire.
`#SBATCH --mail-type=<BEGIN,END,FAIL,TIME_LIMI`		Permet d'être notifié par e-mail d'un évènement particulier dans la vie du job : début de l'exécution (`BEGIN`), fin d'exécution (`END`, `FAIL` et `TIME_LIMIT`)...
`#SBATCH --sockets-per-node=<n>`		Nombre de sockets par *node*
`#SBATCH --cores-per-socket=<n>`		Nombre de cœurs par *sockets*
`#SBATCH --threads-per-core=<n>`		Nombre de *threads* par cœur
`#SBATCH --cpus-per-task=<n>`	`-c <n>`	Définit le nombre de CPUs à allouer par *Task*. L'utilisation effective de ces CPUs est à la charge de chaque *Task*.

Options SBATCH (3)

Options	Raccourcis	Description
`#SBATCH --ntasks=<n>`	`-n <n>`	Definit le nombre maximum de Tasks exécutées en parallèle.
`#SBATCH --mem-per-cpu=<n>`		Définit la RAM en Mo allouée à chaque CPU.
`#SBATCH --nodes=<minnodes[-maxnodes]>`	`-N <n>`	Nombre `minimum[-maximum]` de nœuds sur lesquels distribuer les Tasks.
`#SBATCH --ntasks-per-node=<n>`		Utilisée conjointement avec `--nodes`, cette option est une alternative à `--ntasks` qui permet de contrôler la distribution des Tasks sur les différents nœuds.

Variables d'environnement

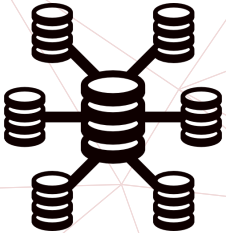
Nom de la variable	Description
SLURM_JOB_ID	L'identifiant du job (calcul)
SLURM_JOB_NAME	Nom du job défini avec l'option -J
SLURM_JOB_NODELIST	Nom d'un fichier qui est fabriqué par SLURM et qui co
SLURM_SUBMIT_HOST	Nom de l'hôte sur lequel SBATCH a été exécuté (chez
SLURM_SUBMIT_DIR	Répertoire depuis lequel le job est soumis
SLURM_JOB_NUM_NODES	Nombre de nœuds requis pour le job
SLURM_NTASKS_PER_NODE	Nombre de cœurs par nœud requis pour le job
SLURM_JOB_CPUS_PER_NODE	Nombre total de threads par nœud

Améliorations possibles

- Gestion plus fine des ressources (RAM, GPU, etc.)
- Interface graphique web pour la Soumission
- Amélioration des ressources (RAM et GPU), et augmentation du nombre de nœuds
- Réserver quelques nœuds pour des étudiant·e·s en master (?)
- etc.

Documentation

https://www.ceremade.dauphine.fr/doc/fr/cluster-slurm



Merci !