



Utilisation d'un cluster de calcul

Initiation à SLURM



Julien Seiler
IFB Core Cluster Taskforce
 @julozi

À propos



Julien Seiler

seilerj@igbmc.fr

Directeur informatique à l'IGBMC, Strasbourg

Co-responsable du National Network of Computing
Resources Cluster de l'IFB

Intermittent de la formation cluster

Qu'est-ce qu'un cluster de calcul ?

Votre ordinateur peut-il faire de la bioinformatique ?



Un ou deux microprocesseurs

Un microprocesseur est chargé de l'exécution des instructions élémentaires demandées par le logiciel

4 à 8 Go de mémoire vive (RAM)

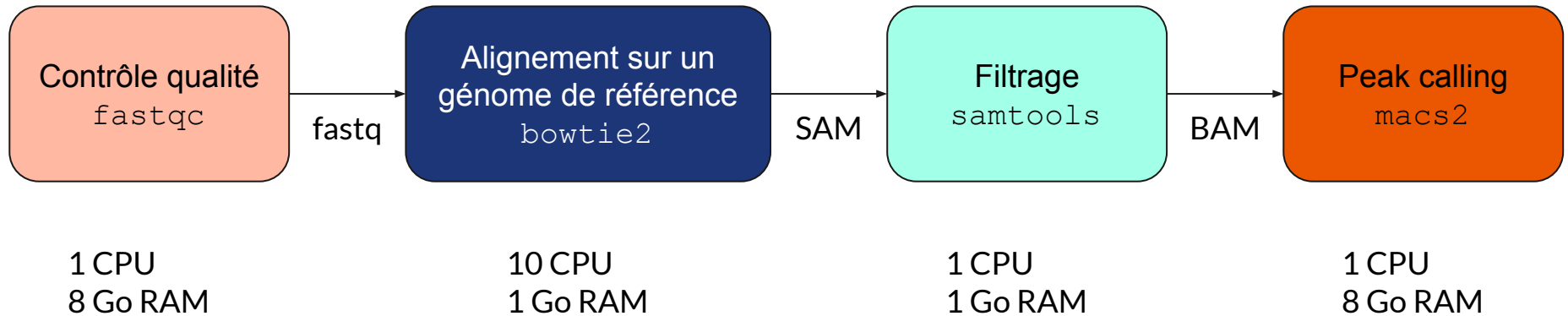
La mémoire vive est utilisée par le microprocesseur pour traiter les données

≈ 1 To d'espace de stockage

L'espace de stockage est utilisé pour conserver de grandes quantités de données de manière plus permanente



Votre ordinateur peut-il faire de la bioinformatique ?



L'exécution de ce workflow nécessite au minimum toutes les ressources d'un ordinateur de bureau pendant plusieurs heures et ceci seulement pour 1 seul fichier fastq.

Pour faire ce type d'analyse nous avons besoin d'ordinateurs plus puissants !

Du data center au coeur



Le Data Center de l'IDRIS
Un bâtiment conçu pour accueillir des infrastructures informatiques

Du data center au coeur

Groupe froid
Pour refroidir les
équipements



Du data center au coeur

Groupe électrogène
Pour garantir l'alimentation
électrique



Du data center au coeur



Les armoires de l'IFB
Chaque armoire peut contenir
80 super-ordinateurs

Du data center au coeur

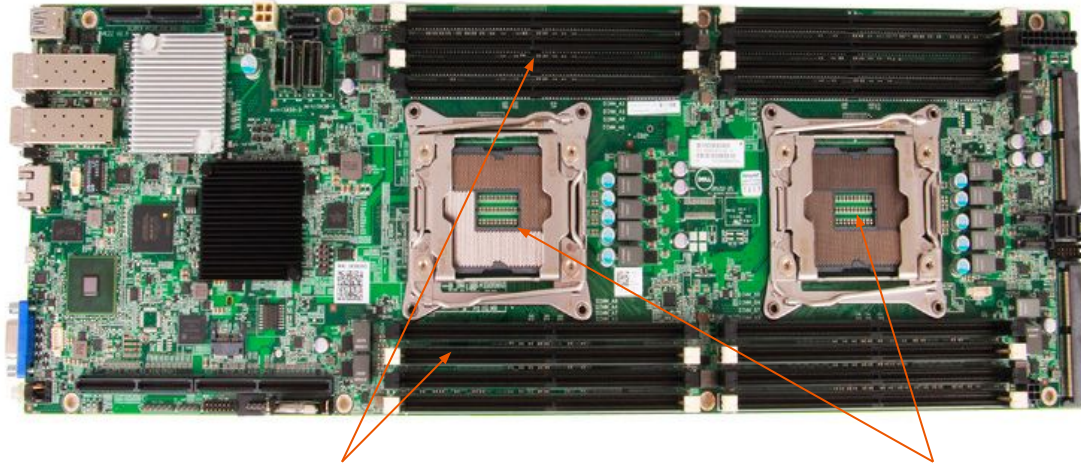


ordinateurs de calcul

Baies de stockage

Du data center au coeur

Un ordinateur ou **noeud** de calcul



Mémoire vive

Supports processeurs

Du data center au coeur

Un microprocesseur



Un microprocesseur contient plusieurs **cœurs**
Chaque coeur se comporte comme un microprocesseur unique.



La fédération de cluster de l'IFB (NNCR)

Cluster	Localisation du Data center	Coeurs	RAM (Go)	Stockage (To)
IFB Core	IDRIS - Orsay	5 042	26 542	2 000
Genotoul	Toulouse	6 128	34 304	3 000
ABiMS	Roscoff	2 608	10 600	2 500
GenOuest	Rennes	1 824	7 500	2 300
Migale	Jouy en Josas	1 084	7 000	350
BiRD	Nantes	560	4 000	500

Accéder au cluster



Les pré-requis pour passer une bonne matinée

- Savoir se connecter à JupyterHub et lancer son serveur Jupyter
- Savoir ouvrir un terminal Unix dans Jupyter
- Savoir quelques commandes Unix de base



A vos claviers

Votre environnement pour le cours cluster

Nous allons créer quelques fichiers d'exercices durant ce cours.

Créer un dossier “cluster” dans votre dossier personnel du projet dubii2021 :

```
/shared/projects/dubii2021/<votre login>/cluster
```

 Terminal

```
$ mkdir /shared/projects/dubii2021/$USER/cluster  
$ cd /shared/projects/dubii2021/$USER/cluster
```




A vos claviers
5 minutes

Exercice 1 - mise en jambe

Quel mot se trouve dans le fichier `toto.txt` dans l'archive

`/shared/projects/dubii2021/trainers/module1/cluster/exercice1.tar.gz` ?

Astuce : copier l'archive dans votre répertoire

`/shared/projects/dubii2021/<votre login>/cluster`

<https://www.wooclap.com/NQXQRF>



Exercice 1 - mise en jambe

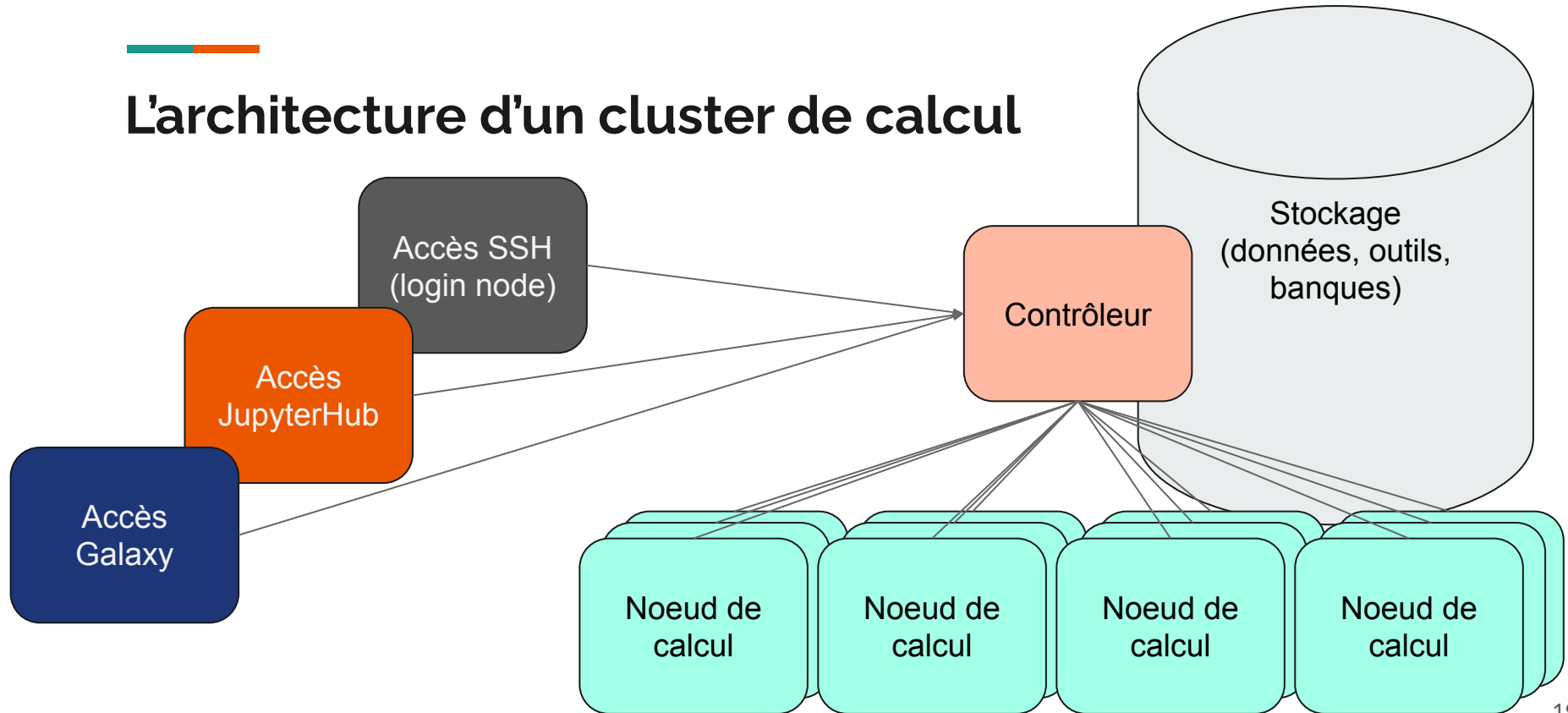
Quel mot se trouve dans le fichier toto.txt dans l'archive

`/shared/projects/dubii2021/trainers/module1/cluster/exercice1.tar.gz` ?

\$ Terminal

```
$ cp /shared/projects/dubii2021/trainers/module1/cluster/exercice1.tar.gz .  
$ tar xzf exercice1.tar.gz  
$ cat toto.txt  
sbatch
```

L'architecture d'un cluster de calcul



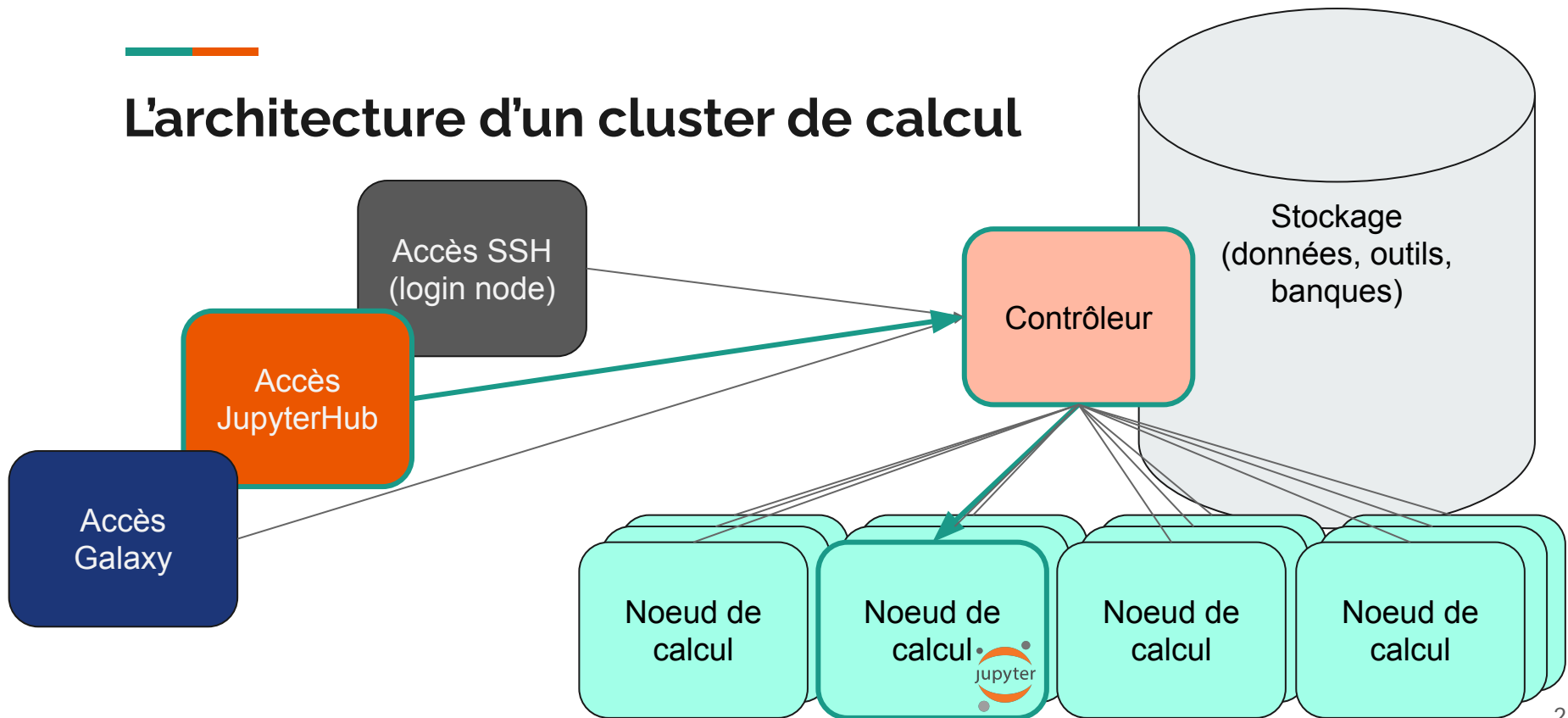


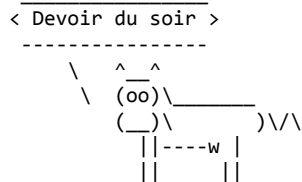
A vos souris
1 minute

**Exercice 2 : lorsque vous utilisez le terminal
Unix ou un notebook dans Jupyter, où sont-ils
exécutés ?**

<https://www.wooclap.com/NQXQRF>

L'architecture d'un cluster de calcul





Et en SSH ?

Exercice à réaliser à la maison :

1. Lisez la documentation : <https://ifb-elixirfr.gitlab.io/cluster/doc/logging-in/> pour apprendre à vous connecter au cluster à l'aide de SSH
2. Chargez le module `cowpy`
3. Lancez la commande : `cowpy I know SSH`
4. Envoyez-moi une capture d'écran de votre terminal sur le Slack du DUBii



Le stockage de données

Les espaces de stockage du cluster sont accessibles depuis l'ensemble des noeuds de calcul ainsi que depuis l'accès SSH (login node) :

`/shared/home/<votre login>`: votre répertoire personnel (7 Go max)

`/shared/projects/<votre projet>`: vos répertoires projets (250 Go max par défaut)

`/shared/bank`: les banques de données de références



Les logiciels

Plus de **400 outils** sont pré-installés sur le cluster.

Ces outils sont déployés à l'aide de



Afin de faciliter la **reproductibilité des analyses**, l'ensemble des versions d'outils installées sur le cluster est conservé.

Pour charger un outil dans votre environnement de travail, il faut utiliser **module**.



Utilisation de module pour charger les logiciels

`module avail -l <nom>` : recherche le logiciel <nom> dans la bibliothèque de logiciels du cluster

`module load blast` : charge la dernière version de blast disponible sur le cluster

`module load blast/2.6.3` : charge la version 2.6.3 de blast

`module list` : liste les outils actuellement chargés dans votre environnement

`module switch blast/2.7.1` : remplace le blast actuellement chargé par la version 2.7.1 de blast

`module unload blast` : décharge blast de votre environnement

`module purge` : décharge tous les outils

Introduction à SLURM



Le gestionnaire de tâches SLURM



SLURM (Simple Linux Utility for Resource Management) est le logiciel utilisé pour assurer la gestion des ressources sur le cluster de calcul de l'IFB.

Il permet de réserver des ressources et lancer des programmes sur les noeuds de calcul du cluster.

Les commandes SLURM commencent toutes par la lettre **s**



Le vocabulaire SLURM

CPU : le CPU est la plus petite unité d'un processeur d'ordinateur.

Cette unité correspond généralement à un thread ou un coeur *hyperthreadé*.

A l'IFB sur la plupart des noeuds nous avons 2 processeurs Haswell E5-2695 v3

Chaque processeur a 14 coeurs physiques et 28 threads (ou coeurs virtuels)

SLURM considère donc **56 CPU** par noeud.

RAM : la RAM est la mémoire utilisée par le processeur pour stocker les données analysées

A l'IFB nous avons en moyenne 252 Go de mémoire vive par noeud

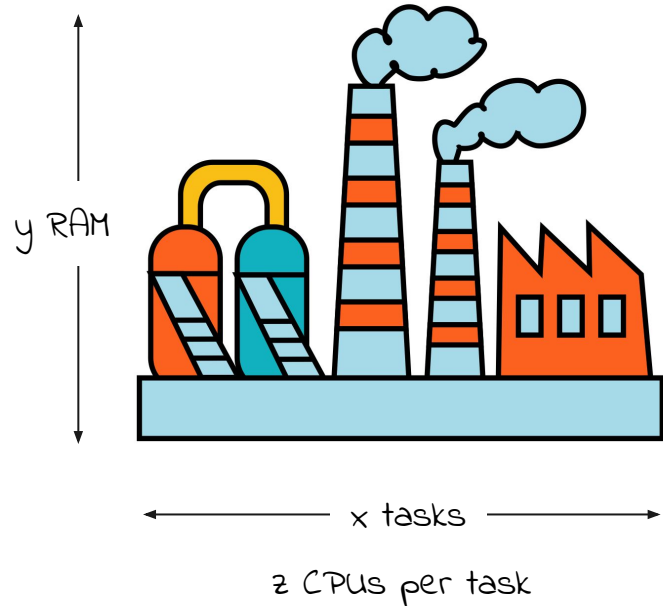
Task : une task est un processus (exécution d'un outil). Elle peut utiliser plusieurs CPU.

Le vocabulaire SLURM

Job : un job est une réservation de ressources (CPU, RAM et tasks) pour effectuer une analyse.

On peut imaginer un job comme **une usine** dans laquelle on va organiser l'exécution de son analyse.

La taille de l'usine va être définie par une quantité de CPU, de RAM et de tasks (on définit généralement le nombre de CPU en fonction du nombre de tasks).

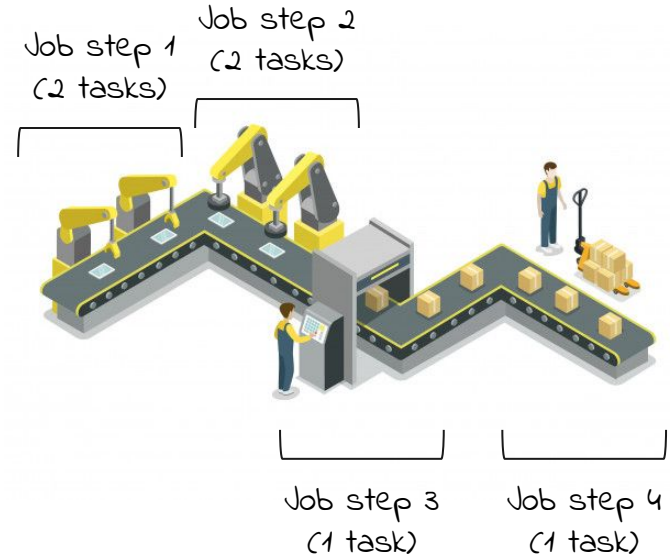


Le vocabulaire SLURM

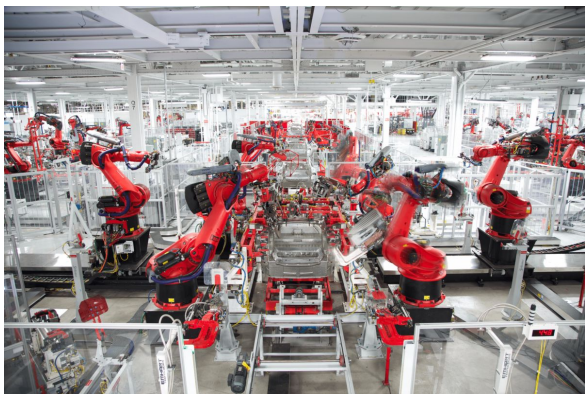
Job step : un “job step” est la partie d’un job qui consiste à exécuter un programme

Un “job step” peut utiliser plusieurs “tasks” mais ceci est spécifique à certains programmes compatibles avec le calcul parallèle. **Dans la plupart des cas, un job step utilise une “task”.**

On peut imaginer un job step comme **un atelier de la chaîne de production** dans notre usine.



Maîtriser SLURM



Usine Tesla à Fremont, CA, USA

Au travers de l'utilisation des commandes SLURM nous allons apprendre à réserver des ressources (une usine) et mettre en route des chaînes de production pour nos analyses.

Plus nous aurons de grandes chaînes de production et plus nos ateliers seront gourmands en ressources, plus nous aurons besoin de réserver de grandes usines !

Soumettre un “job”



Lancer des jobs ou analyse interactive

Il existe deux modes d'utilisation d'une infrastructure de calcul :

Analyse par soumission de jobs

Nécessite l'écriture de scripts

Est adaptée pour des traitements longs et/ou répétitifs

Est adaptée aux outils non interactifs (ne nécessitant pas d'interaction avec un humain)

Analyse interactive

L'utilisateur est intégré au coeur du processus d'analyse

Est adaptée aux traitements courts et uniques

Est adaptée aux outils interactifs (nécessitant une interaction avec un humain)



Nous allons tout d'abord apprendre à soumettre des jobs



Les commandes sbatch et srun

Une analyse bioinformatique que l'on souhaite lancer sur le cluster se présente sous la forme **d'un ou plusieurs scripts shell**.

Pour soumettre un job sur le cluster on utilise principalement deux commandes SLURM :

- `sbatch` permet de réserver des ressources (l'usine) et demander le lancement de son script
- `srun` permet de lancer un "job step" dans son script



Les options les plus utiles

sbatch et srun proposent les mêmes options de base :

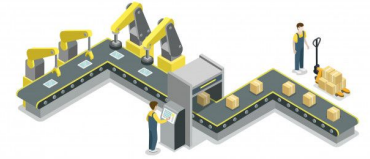
`--cpus-per-task`: Nombre de CPU par tasks

`--mem`: Quantité de mémoire vive allouée par noeud de calcul (exprimée en Mo par défaut)

`--mem-per-cpu`: Quantité de mémoire exprimée en fonction du nombre de CPU

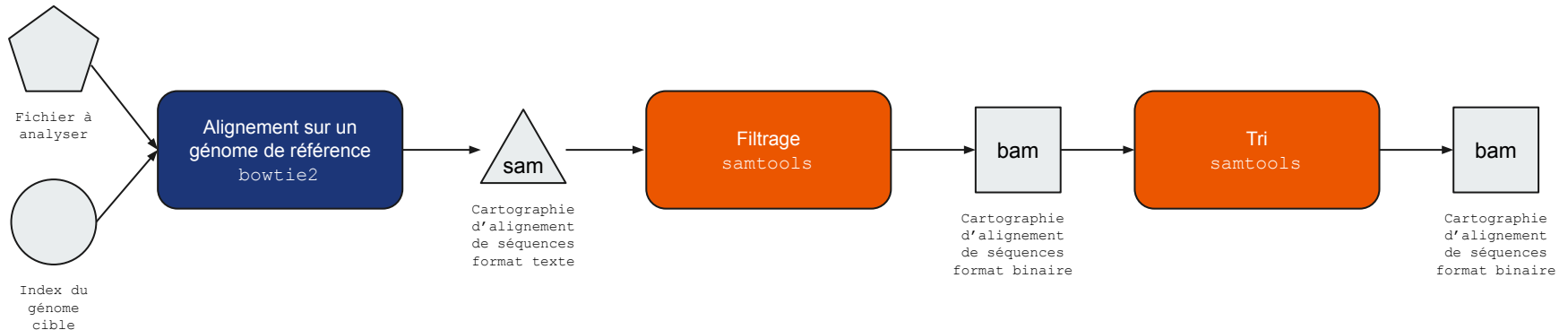
`--ntasks`: Nombre de tasks (par défaut un job est dimensionné avec 1 task)

Effectuer un alignement de séquences

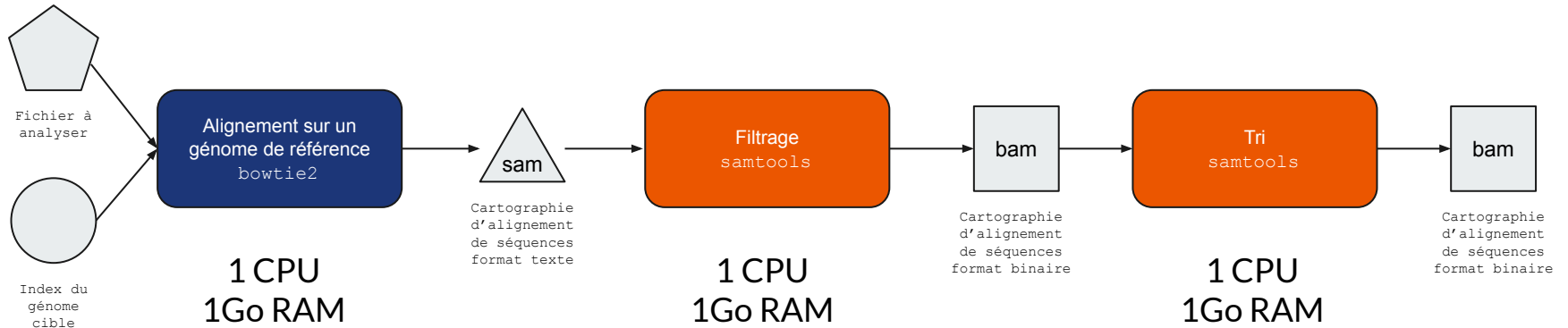


Nous allons utiliser deux outils de bioinformatique pour réaliser notre première analyse avec SLURM :

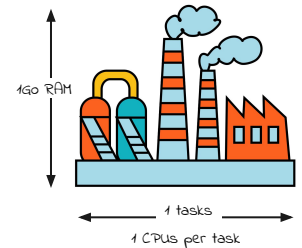
- **bowtie2** : permet d'aligner une séquence sur un génome de référence. On utilise un index réalisé à partir du génome de référence pour effectuer l'alignement
- **samtools** : permet de filtrer, trier et convertir une cartographie d'alignement de séquences



Effectuer un alignement de séquences



Nous devons réserver au moins **1 task**, **1 CPU** et **1Go** de RAM pour réaliser ce traitement





A vos claviers

Effectuer un alignement de séquences

Copier le fichier `/shared/projects/dubii2021/trainers/module1/cluster/exercice3.sh`
dans votre dossier cluster :

\$_ Terminal

```
$ cp /shared/projects/dubii2021/trainers/module1/cluster/exercice3.sh .
```

Effectuer un alignement de séquence

Soumettez votre script à l'aide de sbatch sans oublier d'indiquer vos besoins de ressources :

shebang

```
1  #!/bin/bash
2
3  module load bowtie2 samtools
4
5  reference_index="/shared/bank/arabidopsis_thaliana/TAIR10.1/bowtie2/Arabidopsis_thaliana.TAI
6  R10.1_genomic"
7  file_id="K01_1"
8
9  srunch bowtie2 -x "${reference_index}" -U
10 "/shared/projects/dubii2021/trainers/module1/cluster/${file_id}.fastq.gz" -S
11 "${file_id}.sam"
12 srunch samtools view -hbS -q 30 -o "${file_id}.filtered.bam" "${file_id}.sam"
13 srunch samtools sort -o "${file_id}.bam" "${file_id}.filtered.bam"
```

3 job
steps

\$_ Terminal

```
$ sbatch --cpus-per-task=1 --mem=1G exercice3.sh
Submitted batch job 15274185
```

job ID



Suivre son job avec squeue

La commande **squeue** permet de visualiser des informations sur les jobs dans la file d'attente de SLURM

`squeue -u $USER`: permet de voir les jobs de votre utilisateur

`squeue -u $USER -t RUNNING` : permet de voir les jobs en cours d'exécution

`squeue -u $USER -t PENDING`: permet de voir les jobs en attente

`squeue -j <jobid>`: permet de voir l'état d'un job à partir de son job id



A vos claviers

Suivre son job avec squeue

\$ Terminal							
\$ squeue -u \$USER -t RUNNING							
JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
15274015	fast	jupyter	jseiler	R	1:38:52	1	cpu-node-9
15274185	fast	exercice	jseiler	R	5:57	1	cpu-node-9

Mon serveur Jupyter

Mon alignement



Suivi de l'utilisation des ressources avec sacct

SLURM intègre un mécanisme pour suivre la consommation des ressources de chaque job.

Les jobs ne respectant pas leur réservation de ressources peuvent être tués automatiquement par le cluster.



Suivi de l'utilisation des ressources avec sacct

La commande **sacct** permet d'interroger la base de données de SLURM afin de suivre la consommation des ressources :

Consulter les informations de bases du job `job_id`

```
sacct -j <job_id>
```

Consulter les informations de l'ensemble de ses jobs depuis le 1 mars 2021

```
sacct --start=2021-03-01
```

Afficher des informations détaillées pour le job `job_id`

```
sacct --format=JobID,JobName,State,Start,Elapsed,CPUTime,NodeList -j <job_id>
```

Pour connaître l'ensemble des informations disponibles, on peut utiliser `sacct --helpformat`



A vos claviers

Suivi de l'utilisation des ressources avec sacct

Terminal

```
$ sacct --format=JobID,JobName,State,Start,Elapsed,CPUTime,NodeList -j 15274185
```

JobID	JobName	State	Start	Elapsed	CPUTime	NodeList
15274324	exercice3+	RUNNING	2021-03-04T11:35:02	00:09:00	00:09:00	cpu-node-9
15274324.ba+	batch	RUNNING	2021-03-04T11:35:02	00:09:00	00:09:00	cpu-node-9
15274324.0	bowtie2	RUNNING	2021-03-04T11:35:03	00:08:59	00:08:59	cpu-node-9

Le job step **bowtie2** est en cours...

batch est un job step particulier pour toutes les commandes du script lancées sans `srun`



Effectuer un alignement de séquence

La soumission du job a entraîné la création d'un fichier `slurm-<jobID>.out`

Ce fichier contient le résultat des commandes exécutées au sein du job :

```
5181347 reads; of these:
  5181347 (100.00%) were unpaired; of these:
    1322956 (25.53%) aligned 0 times
    3394416 (65.51%) aligned exactly 1 time
    463975 (8.95%) aligned >1 times
74.47% overall alignment rate
[bam_sort_core] merging from 1 files and 1 in-memory blocks...
```



Annuler un job

La commande **scancel** permet d'arrêter un ou plusieurs jobs

```
scancel <job_id>: arrête le job job_id
```

```
scancel -u $USER: arrêter tous ses jobs (à utiliser avec prudence)
```



A vos claviers

Annuler un job

```
$ _ Terminal  
$ scancel 15274185
```

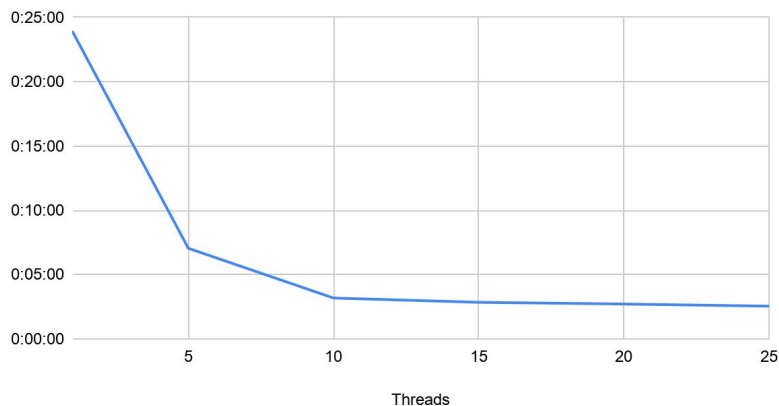
Améliorer les performances de son analyse

La commande **bowtie2** est capable d'utiliser plusieurs threads (fils d'exécution) en parallèle :

```
--threads <int> number of alignment  
threads to launch
```

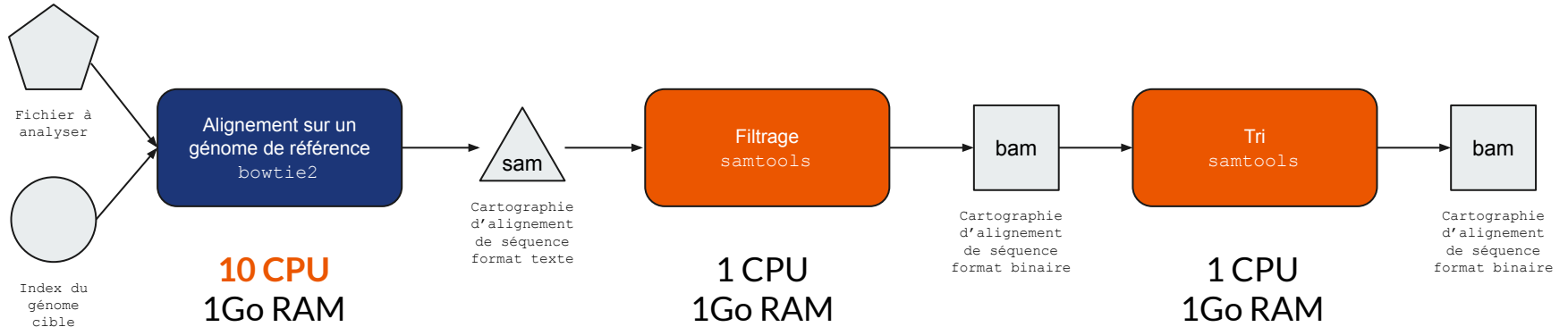
Pour accélérer l'alignement nous pouvons donc réserver plus de CPU et indiquer à bowtie2 de les utiliser au travers de l'option `--threads`

Bowtie2 performances in relation to the number of threads

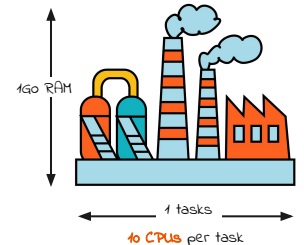


Durée d'exécution de bowtie2 en fonction du nombre de threads sur le fichier KO1_1.fastq.gz

Améliorer les performances de son analyse



Nous devons réserver au moins **1 task**, **10 CPU** et **1Go** de RAM pour réaliser ce traitement





A vos claviers

Améliorer les performances de son analyse

Modifiez le fichier `exercice3.sh` puis soumettez-le à l'aide de `sbatch` :

`exercice3.sh`

```
1  #!/bin/bash
2
3  module load bowtie2 samtools
4
5  reference_index="/shared/bank/arabidopsis_thaliana/TAIR10.1/bowtie2/Arabidopsis_thaliana.TAI
6  R10.1_genomic"
7  file_id="K01_1"
8
9  srun bowtie2 --threads=10 -x "${reference_index}" -U
10 "/shared/projects/dubii2021/trainers/module1/cluster/${file_id}.fastq.gz" -S
    "${file_id}.sam"
11 srun samtools view -hbs -q 30 -o "${file_id}.filtered.bam" "${file_id}.sam"
12 srun samtools sort -o "${file_id}.bam" "${file_id}.filtered.bam"
```

\$_ Terminal

```
$ sbatch --cpus-per-task=10 --mem=1G exercice3.sh
Submitted batch job 15274185
```



Bonnes pratiques pour la reproductibilité

```
1 #!/bin/bash
2
3 #SBATCH --cpus-per-task=10
4 #SBATCH --mem=1G
5
6 module load bowtie2/2.4.1 samtools/1.10
7
8 reference_index="/shared/bank/arabidopsis_thaliana/TAIR10.1/bowtie2/Ar
9 abidopsis_thaliana.TAIR10.1_genomic"
10 file_id="K01_1"
11
12 srun bowtie2 --threads="${SLURM_CPUS_PER_TASK}" -x
13 "${reference_index}" -U
14 "/shared/projects/dubii2021/trainers/module1/cluster/${file_id}.fastq.
15 gz" -S "${file_id}.sam"
16
17 srun samtools view -hbS -q 30 -o "${file_id}.filtered.bam"
18 "${file_id}.sam"
19
20 srun samtools sort -o "${file_id}.bam" "${file_id}.filtered.bam"
```

Déclarez les options de sbatch
directement dans le script du job

Précisez les versions des outils utilisés

Utilisez les variables d'environnement
de SLURM pour l'utilisation des
ressources



Plus d'options pour sbatch ou srun

Options	Description
<code>--cpus-per-task</code>	Nombre de CPU par tasks
<code>--mem</code>	Quantité de mémoire vive allouée au job par noeud (exprimée en Mo par défaut)
<code>--mem-per-cpu</code>	Quantité de mémoire exprimée en fonction du nombre de CPU
<code>--ntasks</code>	Nombre de tasks (par défaut un job est dimensionné avec 1 task)
<code>-J</code>	Nom du job ou du job step
<code>--time</code>	Durée maximum (D-HH:MM:SS)
<code>--output</code>	Orientation de la sortie standard (peut contenir l'identifiant du job : %j)
<code>--error</code>	Orientation de la sortie d'erreur (peut contenir l'identifiant du job : %j)
<code>--mail-type</code>	Activer les notifications par mail (BEGIN, END, FAIL, REQUEUE ou ALL)
<code>--mail-user</code>	Adresse email pour les notifications par mail



Un script complet

```
1  #!/bin/bash
2
3  #SBATCH --cpus-per-task=10
4  #SBATCH --mem=1G
5  #SBATCH --time=05:00
6  #SBATCH --output=alignment-%j.out
7  #SBATCH --error=alignment-%j.err
8  #SBATCH --mail-type=END,FAIL
9  #SBATCH --mail-user=mon@email.fr
10
11 module load bowtie2/2.4.1 samtools/1.10
12
13 reference_index="/shared/bank/arabidopsis_thaliana/TAIR10.1/bowtie2/Arabidopsis_thaliana.TAIR10.1_genomic"
14 file_id="K01_1"
15
16 srunch bowtie2 --threads="${SLURM_CPUS_PER_TASK}" -x "${reference_index}" -U
17   "/shared/projects/dubii2021/trainers/module1/cluster/${file_id}.fastq.gz" -S "${file_id}.sam"
18 srunch -J "filter" samtools view -hbs -q 30 -o "${file_id}.filtered.bam" "${file_id}.sam"
19 srunch -J "sort" samtools sort -o "${file_id}.bam" "${file_id}.filtered.bam"
```

Le calcul parallèle

Comment aligner plusieurs fastq en parallèle ?

Dans l'exercice précédent, notre job traitait l'alignement d'un seul fichier fastq.

Pour aligner plusieurs fichiers fastq, nous avons deux grandes stratégies possibles :



Fichiers 1 à 6

Un grand job



Fichier
1



Fichier
2



Fichier
3



Fichier
4



Fichier
5



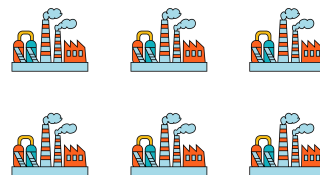
Fichier
6

Plusieurs petits jobs



La meilleure stratégie ?

Un grand job	Plusieurs petits jobs
Un script complexe à écrire (avec une boucle <code>for</code>) et l'utilisation de tasks parallèles	Un script simple pour chaque job
Nécessite de faire une grande réservation de ressources : toutes les ressources nécessaires doivent être disponibles en même temps	Nécessite des petites réservations de ressources : permet de lancer une partie des alignements si toutes les ressources ne sont pas immédiatement disponibles
En cas de crash, il faut modifier le script pour ne relancer que les alignements n'ayant pas réussi	En cas de crash, il suffit de relancer les jobs qui n'ont pas réussi



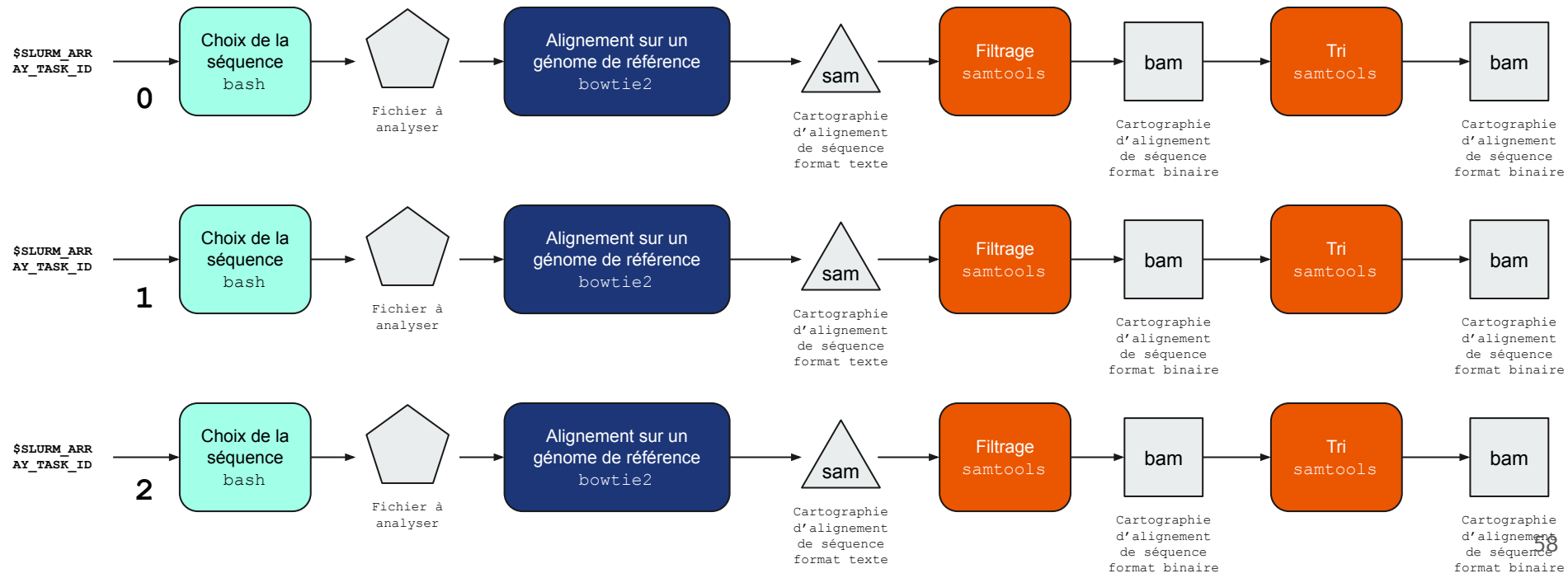
Utilisation d'un "job array"

SLURM propose un mécanisme appelé job array permettant de lancer plusieurs fois un job similaire sous la forme d'une collection (ou array) de jobs.

Chaque job de l'array aura un **index** qui lui est propre.

On peut retrouver l'index du job courant à l'aide de la variable d'environnement `$SLURM_ARRAY_TASK_ID`

Job array et SLURM_ARRAY_TASK_ID





Utilisation d'un "job array"

On utilise l'option **--array** de `sbatch` pour lancer une collection de jobs

```
--array=1,2,3
```

soumet 3 jobs au cluster, index 1 pour le premier job, index 2 pour le second et index 3 pour le dernier

```
--array=0-15
```

soumet 16 jobs, avec une séquence d'index allant de 0 à 15, index 0 pour le premier job et 15 pour le dernier job

```
--array=0-15:2
```

définit une séquence d'index de 0 à 15 avec un pas de 2

équivalent à `--array=0,2,4,6,8,10,12,14`

```
--array=5-15%4
```

définit une séquence d'index de 0 à 15, mais seulement 4 jobs maximum sont exécutés simultanément








Comment aligner plusieurs séquences en parallèle ?

Le dossier

`/shared/projects/dubii2021/trainers/module1/cluster`
contient 6 fichiers de séquences

Nous devons donc créer un array de 6 jobs (donc 6 numéro d'index) !

Comment associer chaque numéro d'index de job à un fichier de séquence unique ?

/ ... / module1 / cluster /	
Name	Last Modified
 exercice1.tar.gz	2 days ago
 KO1_1.fastq.gz	21 hours ago
 KO2_1.fastq.gz	21 hours ago
 KO3_1.fastq.gz	21 hours ago
 WT1_1.fastq.gz	21 hours ago
 WT2_1.fastq.gz	21 hours ago
 WT3_1.fastq.gz	21 hours ago

Comment aligner plusieurs séquences en parallèle ?

```
1  #!/bin/bash
2
3  #SBATCH --array=0-5
4  #SBATCH --cpus-per-task=10
5  #SBATCH --mem=1G
6
7  module load bowtie2/2.4.1 samtools/1.10
8
9  reference_index="/shared/bank/arabidopsis_thaliana/TAIR10.1/bowtie2/Arabidopsis_thal
10 iana.TAIR10.1_genomic"
11 file_id="K01_1"
12
13 srun bowtie2 --threads="${SLURM_CPUS_PER_TASK}" -x "${reference_index}" -U
14 "/shared/projects/dubii2021/trainers/module1/cluster/${file_id}.fastq.gz" -S
   "${file_id}.sam"
15 srun samtools view -hbS -o "${file_id}.filtered.bam" "${file_id}.sam"
16 srun samtools sort -o "${file_id}.bam" "${file_id}.filtered.bam"
```

6 jobs (index : 0, 1, 2, 3, 4, 5)

toujours le même fichier d'entrée

toujours le même fichier de sortie

Comment aligner plusieurs séquences en parallèle ?





Comment aligner plusieurs séquences en parallèle ?

SLURM_ARRAY_TASK_ID	file_id
0	KO1_1
1	KO2_1
2	KO3_1
3	WT1_1
4	WT2_1
5	WT3_1



Comment aligner plusieurs séquences en parallèle ?

Bash vient à notre secours !

```
files=(/shared/projects/dubii2021/trainers/module1/cluster/*fastq.gz)
```

La variable `files` contient à présent un tableau avec dans chaque case le chemin d'un fichier `fastq.gz` différent.

Par exemple `${files[0]}` vaut

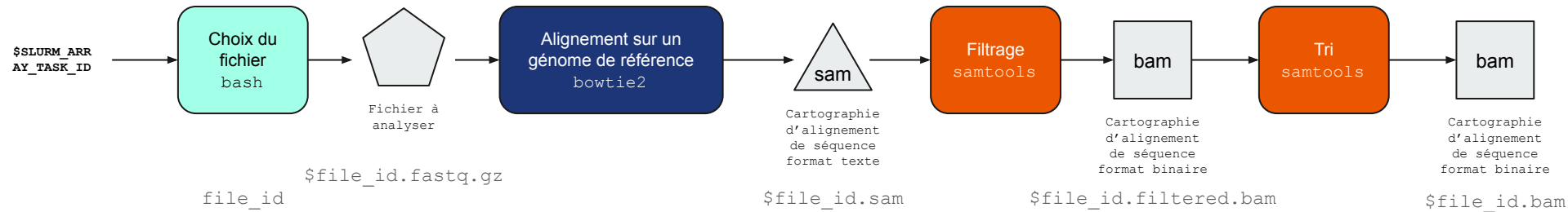
```
/shared/projects/dubii2021/trainers/module1/cluster/K01_1.fastq.gz
```

Pour ne garder que l'identifiant de la séquence (K01_1) :

```
file_id=$(basename -s .fastq.gz ${files[0]})
```

que le nom du fichier   sans le suffixe `.fastq.gz`

Choix du fichier fastq en fonction de l'index du job



0	K01_1	K01_1.fastq.gz	K01_1.sam	K01_1.filtered.bam	K01_1.bam
1	K02_1	K02_1.fastq.gz	K02_1.sam	K02_1.filtered.bam	K02_1.bam
2	K03_1	K03_1.fastq.gz	K03_1.sam	K03_1.filtered.bam	K03_1.bam
3	WT1_1	WT1_1.fastq.gz	WT1_1.sam	WT1_1.filtered.bam	WT1_1.bam
4	WT2_1	WT2_1.fastq.gz	WT2_1.sam	WT2_1.filtered.bam	WT2_1.bam
5	WT3_1	WT3_1.fastq.gz	WT3_1.sam	WT3_1.filtered.bam	WT3_1.bam



Comment aligner plusieurs séquences en parallèle ?

- Faites le ménage dans votre dossier `cluster`
- Et copiez le fichier `/shared/projects/dubii2021/trainers/module1/cluster/exercice4.sh` dans votre dossier `cluster` :

 Terminal

```
$ rm *.sam *.bam *.out
$ cp /shared/projects/dubii2021/trainers/module1/cluster/exercice4.sh .
```



A vos claviers

Comment aligner plusieurs séquences en parallèle ?

```
1 #!/bin/bash
2
3 #SBATCH --array=0-5
4 #SBATCH --cpus-per-task=10
5 #SBATCH --mem=1G
6
7 module load bowtie2/2.4.1 samtools/1.10
8
9 reference_index="/shared/bank/arabidopsis_thaliana/TAIR10.1/bowtie2/Arabidopsis_thaliana.TAIR10.1
10 _genomic"
11 files=(/shared/projects/dubii2021/trainers/module1/cluster/*fastq.gz)
12 file_id=$(basename -s .fastq.gz "${files[$SLURM_ARRAY_TASK_ID]}")
13
14 srun -J "${file_id} bowtie2" bowtie2 --threads=${SLURM_CPUS_PER_TASK} -x "${reference_index}" -U
15 "/shared/projects/dubii2021/trainers/module1/cluster/$file_id.fastq.gz" -S "$file_id.sam"
16 srun -J "${file_id} filter" samtools view -hbS -q 30 -o "$file_id.filtered.bam" "$file_id.sam"
17 srun -J "${file_id} sort" samtools sort -o "$file_id.bam" "$file_id.filtered.bam"
```



A vos claviers

Comment aligner plusieurs séquences en parallèle ?

\$ Terminal

```
$ sbatch exercice4.sh  
Submitted batch job 15291683
```

```
$ sacct --format=JobID%20,JobName%15,State,Start,Elapsed,CPUTime,NodeList -j 15291683
```

JobID	JobName	State	Start	Elapsed	CPUTime	NodeList
15354488_0	exercice4.sh	RUNNING	2021-03-08T20:44:56	00:01:43	00:17:10	cpu-node-12
15354488_0.batch	batch	RUNNING	2021-03-08T20:44:56	00:01:43	00:17:10	cpu-node-12
15354488_0.0	KO1_1 bowtie2	RUNNING	2021-03-08T20:44:56	00:01:43	00:17:10	cpu-node-12
15354488_1	exercice4.sh	RUNNING	2021-03-08T20:44:56	00:01:43	00:17:10	cpu-node-48
15354488_1.batch	batch	RUNNING	2021-03-08T20:44:56	00:01:43	00:17:10	cpu-node-48
15354488_1.0	KO2_1 bowtie2	RUNNING	2021-03-08T20:44:57	00:01:42	00:17:00	cpu-node-48
15354488_2	exercice4.sh	RUNNING	2021-03-08T20:44:56	00:01:43	00:17:10	cpu-node-14
15354488_2.batch	batch	RUNNING	2021-03-08T20:44:56	00:01:43	00:17:10	cpu-node-14
15354488_2.0	KO3_1 bowtie2	RUNNING	2021-03-08T20:44:57	00:01:42	00:17:00	cpu-node-14
15354488_3	exercice4.sh	RUNNING	2021-03-08T20:44:56	00:01:43	00:17:10	cpu-node-15
15354488_3.batch	batch	RUNNING	2021-03-08T20:44:56	00:01:43	00:17:10	cpu-node-15
15354488_3.0	WT1_1 bowtie2	RUNNING	2021-03-08T20:44:57	00:01:42	00:17:00	cpu-node-15
15354488_4	exercice4.sh	RUNNING	2021-03-08T20:44:56	00:01:43	00:17:10	cpu-node-79
15354488_4.batch	batch	RUNNING	2021-03-08T20:44:56	00:01:43	00:17:10	cpu-node-79
15354488_4.0	WT2_1 bowtie2	RUNNING	2021-03-08T20:44:57	00:01:42	00:17:00	cpu-node-79
15354488_5	exercice4.sh	RUNNING	2021-03-08T20:44:56	00:01:43	00:17:10	cpu-node-79
15354488_5.batch	batch	RUNNING	2021-03-08T20:44:56	00:01:43	00:17:10	cpu-node-79
15354488_5.0	WT3_1 bowtie2	RUNNING	2021-03-08T20:44:57	00:01:42	00:17:00	cpu-node-79

L'analyse interactive



Analyse interactive

Une analyse interactive consiste à exécuter des outils de calcul
sans passer par la soumission d'un script

Comment réserver des ressources si l'on n'utilise plus sbatch ?



salloc



Analyse interactive

`salloc` permet de réserver des ressources de calcul et de les associer à un shell (bash par défaut)

Ce shell est ensuite utilisé pour exécuter des commandes `srun` permettant de lancer des tasks



Essayons un alignement interactif

 Terminal

```
$ salloc --cpus-per-task=10 --mem=1G
salloc: Granted job allocation 15299355
$ module load bowtie2/2.4.1
$ srun bowtie2 --threads=10 --met-stderr -x
/shared/bank/arabidopsis_thaliana/TAIR10.1/bowtie2/Arabidopsis_thaliana.TAIR10.1_genomic -U
/shared/projects/dubii2021/trainers/module1/cluster/KO1_1.fastq.gz -S KO1_1.sam
```


Conclusion

Construisez autant d'usines que nécessaire : `sbatch`

Chargez vos outils avec module

Définissez chaque étape de vos chaînes de production : `srun`

Distribuez vos analyses à l'aide des jobs array



Passez en mode interactif (`salloc + srun`) pour **débugger** ou **découvrir vos outils**

Besoin d'aide ?

Il manque un outil !

J'ai plus de place !

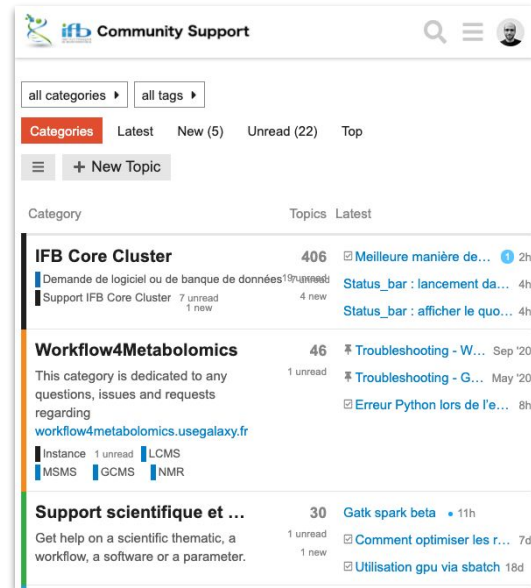
HEEEELP !

Je ne trouve pas un index...

Rejoignez la communauté IFB

Rendez-vous sur :

<https://community.france-bioinformatique.fr>



Besoin d'un outil ?

Demande d'installation de FusionInspector

IFB Core Cluster | Demande de logiciel ou de banque de données

This is the first time Camille has posted — let's welcome them to our community!

Camille

Bonjour,

Je me permets de vous contacter car dans le cadre de mon projet de master 2, j'aurais besoin de l'outil FusionInspector pour lequel mon ordinateur personnel n'est pas assez puissant. C'est un outil permettant d'analyser les fusions en RNAseq. Ne sachant pas comment procéder, serait-il possible de vous l'installer ? Voici l'adresse <https://github.com/FusionInspector/FusionInspector/wiki/installing-FusionInspector>.

Je vous serais très reconnaissante de votre aide.

Cordialement,

Camille Baron encadrée par Audrey Gros à l'université de Bordeaux

✓ Solved by [gildaslecorguille](#) in post #4

Hop: module load fusion-inspector/2.2.1

created 20d last reply 19d 4 replies 25 views 2 users 2 links

Installation (nonpareil) @team.software

IFB Core Cluster | Demande de logiciel ou de banque de données

echase

Bonjour @team.software

Serait-il possible d'installer nonpareil sur le cluster, SVP ?

GitHub

[Imrodriguez/nonpareil](#)

Estimate metagenomic coverage and sequence diversity - Imrodriguez/nonpareil

Merci pour votre temps,

Emily

✓ Solved by [julien](#) in post #3

nonpareil 3.3.4 est à présent disponible sur le cluster : module load nonpareil/3.3.4 Bonne journée, Julien

created 3 Feb last reply 4 Feb 2 replies 28 views 2 users 2 links 1 link

Installation pyslim @team.software

IFB Core Cluster | Demande de logiciel ou de banque de données

Guillaume_Lan-Fong

Bonjour,

Faisant suite à mon post pour l'installation de SLiM, j'aimerais savoir s'il était possible d'installer le package python "pyslim" (<https://pyslim.readthedocs.io/en/latest/installation.html>) permettant de travailler sous python sur les .trees générés par SLiM.

L'installation devrait être possible via un simple : `python3 -m pip install pyslim`

En vous remerciant par avance,

Lan-Fong Guillaume

✓ Solved by [gildaslecorguille](#) in post #7

Hop: module load pyslim/0.501

created 24d last reply 11d 7 replies 38 views 3 users 1 like 1 link