Inserm

La science pour la santé
From science to health

Guide d'utilisation

Extraire des variants génomiques en utilisant le cluster HPC CPU

Inserm - Cloud self-service - offre HPC

Objectif du document

Ce tutoriel vous guidera pas-à-pas dans le cas d'usage "extraction de variants génomiques sur un cluster HPC basé sur CPU" et la mise en place d'un pipeline de calcul avec Apptainer, Biocontainer et l’orchestrateur Slurm.

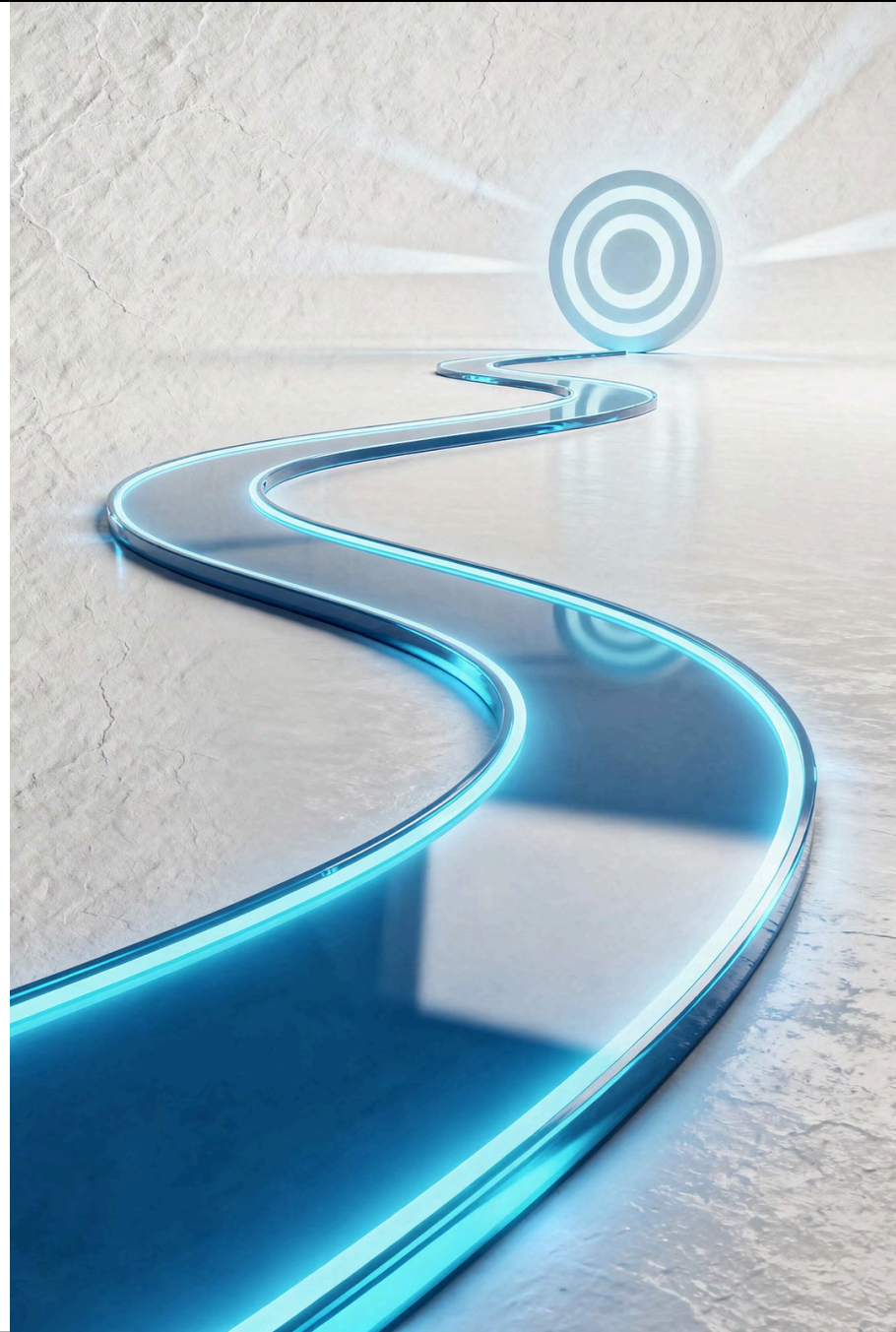

Table des matières

1 PRESENTATION DU CAS D'USAGE ET DE LA STRATEGIE DE MISE EN OEUVRE

1.1 Présentation du cas d'usage

Le cas d'usage présenté dans ce tutoriel est l'extraction des variants d'un séquençage de la bactérie Escherichia coli.

Pour constituer une chaîne de traitement permettant de transformer des données de séquençage brutes en informations exploitables sur le génome, nous allons mettre en œuvre 3 étapes et 3 outils, qui seront exécutés sur la plateforme HPC.

1. Alignement

Bowtie2 aligne les lectures brutes sur le génome de référence.

2. Traitement

Samtools organise les alignements dans des formats binaires efficaces.

3. Détection de variantes

BCFtools identifie les variantes génomiques à partir des alignements.

1.2 Stratégie de mise en œuvre

1.2.1 Biocontainers et Apptainer

- Ces trois outils Bowtie2, Samtools et BCPtools sont disponibles sous formes de **biocontainer** (https://biocontainers.pro/) ce qui facilite leur utilisation sans nécessité l'installation de ces outils sur les nœuds de calculs.
- Les containers seront mis en œuvre au travers de Apptainer, disponible sur chaque nœud du cluster HPC.
- **Apptainer** est un moteur de conteneurisation destiné aux charges de travail hautes performances (HPC), aux applications scientifiques et aux environnements nécessitant une sécurité stricte. Il s'agit d'un fork et d'une continuation communautaire du célèbre projet Singularity, qui s'est concentré sur les conteneurs adaptés au calcul scientifique et HPC.

1.2.2 Utilisation de la commande sbatch

La chaine de traitement sera mise en œuvre au travers de la command **Slurm sbatch :**

Commande sbatch

La commande **sbatch** est utilisée dans le gestionnaire de ressources Slurm pour soumettre des scripts de tâches (jobs) à la file d'attente. Ces tâches peuvent être des simulations, des calculs complexes ou tout autre processus qui doit être exécuté sur un cluster slurm tel que le HPC le met en œuvre.

Répertoire partagé

Nous avons fait le choix de stocker ce projet dans un répertoire accessible à tous les utilisateurs de la plateforme HPC dans `/shared/examples/genomics`

Ce répertoire partagé "genomics" contient les sous-répertoires suivants :

/shared/examples/genomics/ecoli-rel606/ref

Contient le fichier génome de Escherichia coli rel606

/shared/examples/genomics/ecoli-rel606/data

Contient les fichiers de séquençage génomique

/shared/examples/genomics/ecoli-rel606/out

Contiendra les résultats au format vcf de l'extraction des variants.

/shared/examples/genomics/images

Contiendra les biocontaineurs outils.

Note : ces données génomiques ont été importées via le processus sécurisé en vigueur. Un autre document tutoriel "Tuto HPC bucket S3" explique comment utiliser un stockage S3 pour effectuer du transfert de données externes vers la plateforme HPC.

2 CONCEPTION

2.1 Pipeline de traitement

Le script doit permettre de faire s'enchainer 6 traitements séquentiellement :

1) Exécuter la commande ci-dessous pour créer les fichiers d'index nécessaires pour permettre à l'aligneur de rechercher efficacement les séquences dans le génome de référence.

```
#build index for E. coli reference and put the index files in the same directory as reference sequence

bowtie2-build $REF_DIR/NC_012967.1.fasta $REF_DIR/NC_012967.1

# now that we have built the index, we can go ahead call the aligner
# the alignment takes some time around 10min to finish if run with single core
# so we use $NSLOTS to pass as the number of threads to make sure the command only use the number of cores requested
# incorporate JOB_NAME in the output file name to distinguish output accordingly
```

2) Exécuter la commande ci-dessous pour aligner les reads FASTQ (paired-end) sur la référence pour produire un fichier d'alignement au format SAM.

```
bowtie2 -p $NSLOTS -t -x $REF_DIR/NC_012967.1 -1 $DATA_DIR/SRR030257_1.fastq -2 $DATA_DIR/SRR030257_2.fastq -S $OUT_DIR/${JOB_NAME}_SRR030257.sam

# Use samtools view to convert the SAM file into a BAM file.
# BAM is the binary format corresponding to the SAM text format.
# It's more storage efficient.
```

3) Exécuter la commande ci-dessous pour transformer le fichier SAM (texte) en BAM (binaire compressé), plus léger et plus rapide à manipuler.

```
samtools view -bS $OUT_DIR/${JOB_NAME}_SRR030257.sam > $OUT_DIR/${JOB_NAME}_SRR030257.bam

# Use samtools sort to convert the BAM file to a sorted BAM file.
# sorted BAM is a useful format because the alignments are
# (a) compressed, which is convenient for long-term storage, and
# (b) sorted, which is conveneint for variant discovery.
```

4) Exécuter la commande ci-dessous pour trier les alignements pour obtenir un BAM trié, nécessaire pour les analyses downstream (variants, visualisation, indexation).

```
samtools sort $OUT_DIR/${JOB_NAME}_SRR030257.bam -o $OUT_DIR/${JOB_NAME}_SRR030257.sorted.bam

# To generate variant calls in VCF format, use bcftools mpileup
# Run against reference:
```

5) Exécuter la commande ci-dessous pour générer les variants (SNP/indels) au format BCF/VCF à partir du BAM trié et du génome de référence.

```
bcftools mpileup -f $REF_DIR/NC_012967.1.fasta $OUT_DIR/${JOB_NAME}_SRR030257.sorted.bam | bcftools call -mv -Ob -o $OUT_DIR/${JOB_NAME}_SRR030257.raw.bcf

# Then to view the variants, run:
```

6) Exécuter la commande ci-dessous pour afficher le contenu du fichier de variants généré.

```
bcftools view $OUT_DIR/${JOB_NAME}_SRR030257.raw.bcf
```

2.2 Entête du script sbatch

Configuration de l'entête sbatch

Notre script sbatch doit contenir un entête qui donnera à Slurm les indications pour réserver puis faire exécuter le script par un nœud de calcul :

```
#!/bin/bash
#SBATCH --cpus-per-task=12
#SBATCH --ntasks=1
#SBATCH --partition=CPU-16
#SBATCH --time=01:00:00
#SBATCH --job-name=alignement-genomics
#SBATCH --output=alignement-genomics.%j.stdout
#SBATCH --error=alignement-genomics.%j.stderr
```

1

Le calcul utilisera 12 VCPUs :

`--cpus-per-task=12`

2

Comme le calcul est séquentiel, sans parallélisation, ntasks=1

`--ntasks=1`

3

Le nœud sera réservé dans la partition CPU-16 (16vcpu) car les outils des séquençages utilisés ne mettent pas en œuvre de traitement GPU :

`--partition=CPU-16`

4

Le nœud sera réservé au maximum pour une durée de 1h :

`--time=01:00:00`

5

Le nom du Job sera « alignement-genomics» :

`--job-name=alignement-genomics`

6

Les sorties stderr et stdout seront redirigées vers les fichiers alignement-genomics.%j.stderr, et alignement-genomics.%j.stdout, %j prendra pour valeur le numéro du job.

`--output=alignement-genomics.%j.stdout`

`--error=alignement-genomics.%j.stderr`

2.3 Variabilisation du script sbatch

Notre script de traitement doit mettre en œuvre les outils au travers de Apptainer.

Pour faciliter la lecture du script et retrouver la structure de notre traitement, nous avons fait le choix d'utiliser des variables.

```
# Do not changed the following parameters!!
APPTAINER_OPTION="-e --writable-tmpfs --sharens --bind /shared:/shared --nv"
CALL="apptainer exec $APPTAINER_OPTION"

# Biocontainer images to perform genomics stuff
BOWTIE2_IMG="/shared/examples/genomics/images/bowtie2_2.5.4--he96a11b_5.sif"
SAMTOOLS_IMG="/shared/examples/genomics/images/samtools_1.19.2--h50ea8bc_0.sif"
BCFTOOLS_IMG="/shared/examples/genomics/images/bcftools_1.21--h3a4d415_1.sif"

REF_DIR="/shared/examples/genomics/ecoli_rel606/ref"
DATA_DIR="/shared/examples/genomics/ecoli_rel606/data"
OUT_DIR="/shared/examples/genomics/ecoli_rel606/out"
TMP_DIR="/shared/examples/genomics/ecoli_rel606/tmp"
```

2.4 Mise en œuvre du traitement dans le script sbatch

Ensuite, nous mettons en œuvre la chaine de traitement en appelant séquentiellement chaque outil Bowtie2, samtools et bcftools via srun :

```
#build index for E. coli reference and put the index files in the same directory as reference sequence
echo "** build index for E. coli reference"
srun --job-name=index $CALL $BOWTIE2_IMG bash -c "bowtie2-build $REF_DIR/NC_012967.1.fasta $TMP_DIR/NC_012967.1"

# now that we have built the index, we can go ahead call the aligner
# the alignment takes some time around 10min to finish if run with single core
# so we use $NSLOTS to pass as the number of threads to make sure the command only use the number of cores requested
# incorporate JOB_NAME in the output file name to distinguish output accordingly
echo "** run Aligner"
srun --job-name=aligner $CALL $BOWTIE2_IMG bash -c "bowtie2 -p $SLURM_CPUS_PER_TASK -t -x $TMP_DIR/NC_012967.1 -1 $DATA_DIR/SRR030257_1.fastq -2 $DATA_DIR/SRR030257_2.fastq -S
$OUT_DIR/${SLURM_JOB_NAME}-${SLURM_JOB_ID}_SRR030257.sam"

# Use samtools view to convert the SAM file into a BAM file.
# BAM is the binary format corresponding to the SAM text format.
# It's more storage efficient.
echo "** Convert SAM to BAM"
srun --job-name=sam2bam $CALL $SAMTOOLS_IMG bash -c "samtools view -bS $OUT_DIR/${SLURM_JOB_NAME}-${SLURM_JOB_ID}_SRR030257.sam >
$OUT_DIR/${SLURM_JOB_NAME}-${SLURM_JOB_ID}_SRR030257.bam"

# Use samtools sort to convert the BAM file to a sorted BAM file.
# sorted BAM is a useful format because the alignments are
# (a) compressed, which is convenient for long-term storage, and
# (b) sorted, which is conveneint for variant discovery.
echo "** SORT BAM"
srun --job-name=sort_bam $CALL $SAMTOOLS_IMG bash -c "samtools sort $OUT_DIR/${SLURM_JOB_NAME}-${SLURM_JOB_ID}_SRR030257.bam -o
$OUT_DIR/${SLURM_JOB_NAME}-${SLURM_JOB_ID}_SRR030257.sorted.bam"

# To generate variant calls in VCF format, use bcftools mpileup (instead of
# samtools mpileup, which is depreciated). Run against reference:
echo "** Generate Variant in VCF Format"
srun --job-name=generate_variant $CALL $BCFTOOLS_IMG bash -c "bcftools mpileup -f $REF_DIR/NC_012967.1.fasta $OUT_DIR/${SLURM_JOB_NAME}-${SLURM_JOB_ID}_SRR030257.sorted.bam | bcftools call -mv -Ob -o
$OUT_DIR/${SLURM_JOB_NAME}-${SLURM_JOB_ID}_SRR030257.raw.bcf"

# Then to view the variants, run:
echo "** The variants are:"
srun --job-name=compute_variant $CALL $BCFTOOLS_IMG bash -c "bcftools view $OUT_DIR/${SLURM_JOB_NAME}-${SLURM_JOB_ID}_SRR030257.raw.bcf >
$OUT_DIR/${SLURM_JOB_NAME}-${SLURM_JOB_ID}_NC_012967.1_variants.txt "
```

3 EXECUTION SUR LA PLATEFORME HPC

1) Accéder à la fenêtre HPC Shell Access

2) Se déplacer dans le répertoire : `/shared/examples/genomics`

```
[saucouturier@ood ~]$ cd /shared/examples/genomics/
[saucouturier@ood genomics]$ ls -lsa
total 24
4 drwxrwxrwx. 4   1986800008 1986800008 4096 Jan 21 11:02 .
4 drwxrwxrwx. 4   1986800008 1986800008 4096 Jun 10  2025 ..
4 -rw-r--r--. 1 saucouturier 1986800008 3208 Jan 21 10:54 ecoli_genomics_s3.sbatch
4 -rw-rw-rw-. 1 saucouturier 1986800008 3235 Jan 21 11:02 ecoli_genomics.sbatch
4 drwxr-xr-x. 5   1986800008 1986800008 4096 Jan 20  2025 ecoli_rel606
4 drwxrwxrwx. 2   1986800008 1986800008 4096 Jan  7 11:48 images
[saucouturier@ood genomics]$
```

3) Exécuter la commande via `sbatch ecoli_genomics.sbatch` pour lancer l'exécution du traitement

```
[saucouturier@ood genomics]$ sbatch  ecoli_genomics.sbatch
Submitted batch job 145
[saucouturier@ood genomics]$
```

4) Surveiller le calcul via le job id avec : `watch -n 5 squeue -j 145`

2 minutes plus tard, le job est achevé.

```
[saucouturier@ood genomics]$ watch -n 5  squeue -j 145
```

```
Every 5.0s: squeue -j 145                                                ood.lab-6844459598-9wjsq: Wed Jan 21 11:45:41 2026

             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
               145    CPU-16 aligneme saucoutu  R       0:23      1 podscpu-0
```

5) Exécuter la commande `sacct` pour obtenir les détails du traitement

```
[saucouturier@ood genomics]$ sacct -j 145 --format=JobID,JobName,Partition,State,ExitCode,Elapsed
JobID           JobName  Partition      State ExitCode    Elapsed
------------ ---------- ---------- ---------- -------- ----------
145          alignemen+     CPU-16  COMPLETED      0:0   00:01:52
145.batch         batch             COMPLETED      0:0   00:01:52
145.0             index             COMPLETED      0:0   00:00:04
145.1           aligner             COMPLETED      0:0   00:00:21
145.2           sam2bam             COMPLETED      0:0   00:00:23
145.3         sort_bam              COMPLETED      0:0   00:00:28
145.4        generate_+             COMPLETED      0:0   00:00:35
145.5        compute_v+             COMPLETED      0:0   00:00:01
[saucouturier@ood genomics]$
```

6) Vérifier les fichiers de sortie avec le répertoire qui contient les fichiers stderr et stdout du traitement

```
[saucouturier@ood genomics]$ ls -lsa
total 44
 4 drwxrwxrwx. 4   1986800008             1986800008  4096 Jan 21 11:45 .
 4 drwxrwxrwx. 4   1986800008             1986800008  4096 Jun 10  2025 ..
 4 -rw-r--r--. 1 saucouturier Utilisateurs du domaine  2112 Jan 21 11:47 alignement-genomics.145.stderr
16 -rw-r--r--. 1 saucouturier Utilisateurs du domaine 12548 Jan 21 11:47 alignement-genomics.145.stdout
 4 -rw-r--r--. 1 saucouturier             1986800008  3208 Jan 21 10:54 ecoli_genomics_s3.sbatch
 4 -rw-rw-rw-. 1 saucouturier             1986800008  3395 Jan 21 11:45 ecoli_genomics.sbatch
 4 drwxrwxrwx. 7   1986800008             1986800008  4096 Jan 21 11:17 ecoli_rel606
 4 drwxrwxrwx. 2   1986800008             1986800008  4096 Jan  7 11:48 images
[saucouturier@ood genomics]$
```

7) Consulter les résultats du traitement depuis le fichier `ecoli_rel606/out/alignement-genomics-145_NC_012967.1_variants.txt` depuis HPC Shell Access

Si vous avez des questions, contacter vos référents habituels sur le HPC.