

Introduction to the iPOP-UP HPC cluster

Julien Rey Olivier Kirsh Magali Hennion

Juin 2023



Table of Contents

- 1 Introduction
- 2 The iPOP-UP@RPBS HPC cluster
- 3 In practice
- 4 Slurm usage
- 5 Tools
- 6 Conclusion



Who is this training for

- You work at Université Paris Cité
- You need (or might need) more computational power than you currently have
- You are familiar with Unix systems and Bash



The RPBS platform

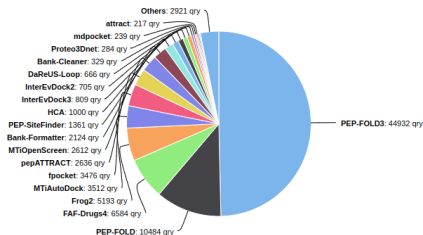
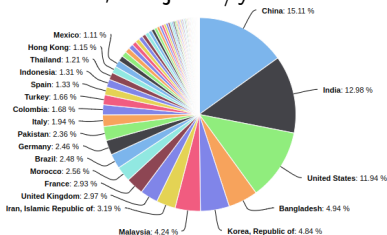
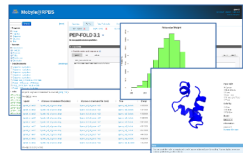


Missions:

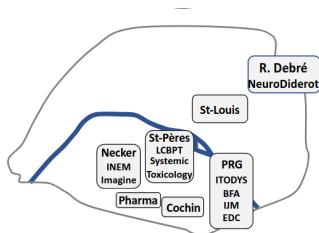
- Development of Structural Bioinformatics methods
- Services deployment on the computing resource
- Expertise and training
- Hardware hosting

The RPBS Web portal

- <https://mobyline.rpbs.univ-paris-diderot.fr>: free access
- 30+ Structural Bioinformatics services
- 1 million CPU hours /year
- 80,000 jobs /year



The iPOP-UP project



- iPOP-UP: Integrative Platform for Omics Projects at Universit  de Paris
- Multisite Bioinformatics platform
- Ranging from multiple 'Omics' techniques to structural and chemo- *in silico* Bioinformatics
- Compute nodes

iPOP-UP : people involved

Executive committee

- **Franck Letourneur** (Cochin)
- **Valérie Mezger** (EDC)
- **Pierre Tufféry** (BFA)
- **Michel Werner** (IJM)
- Karine Audouze (T3S)
- Marc Baaden (LBT)
- Florent Barbault (ITODYD)
- Pierre Gressens (Robert-Debré)
- Pascale Lesage (St Louis)
- Nicolas Leuliot (CiTCoM)
- Bruno Lucas (Cochin)
- Fabiola Terzi (Necker)

Technical committee

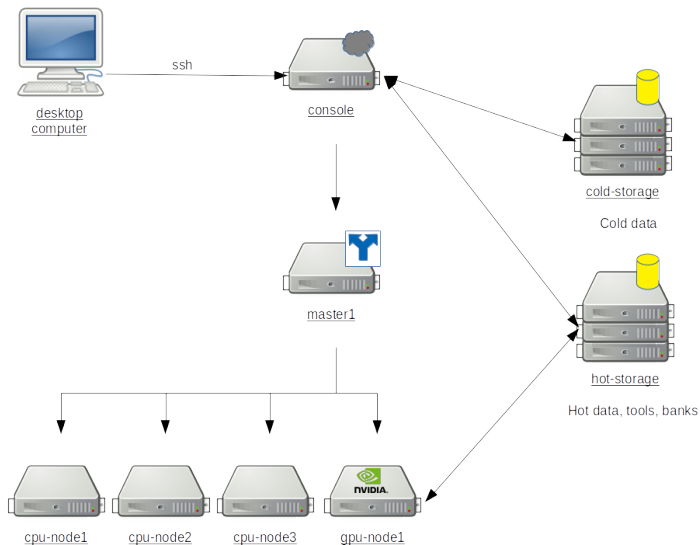
- Christophe Cérin (LIPN,USPN)
- Yves Clément (IJM)
- Magali Hennion (EDC)
- Jean-Philippe Jais (Necker)
- Olivier Kirsh (EDC)
- Pierre Poulain (IJM)
- **Julien Rey** (BFA)
- Guillaume Seith (IGBMC / IFB)
- Nicklas Setterblad (St Louis)

What is a HPC cluster for ?

- High hardware resources needs
- Long running analyses
- A lot of similar analyses
- Shared work between users
- Free your desktop from the task



What is a HPC cluster ?



Computational hardware

Partitions (groups of compute nodes):

- ipop-up: 16 nodes, 2048 CPUs
- rpbs: 20 nodes, 832 CPUs, 9 GPUs (+ 9 nodes, 520 CPUs, 4 GPUs)
- cmpli: 6 nodes, 264 CPUs, 14 GPUs (+ 1 node, 64 CPUs, 2 GPUs)
- epigen: 4 nodes, 128 CPUs
- master-bi: 1 node, 32 CPUs, 3 GPUs

Storage:

- hot-storage: 125TB, very fast
- cold-storage: 240TB, slow + backup (soon)

Virtualization servers, etc...



Note about ssh and security

Your IP will be banned after 5 failed authentication attempts. Each public key count as one authentication attempt. To disable public key authentication:

```
ssh -o PubkeyAuthentication=no rey@ipop-up.rpbs.univ-paris-diderot.fr
```

A good thing to do, change your password:

```
[rey@ipop-up ~]$ passwd
```



Where you can go, write, or execute

User environments

`/shared/home/username`

Computations (hot data)

`/shared/projects/projectname`

Processed data (cold data)

`/cold-storage/username`

Data banks (read-only)

`/shared/banks/`

Note about quotas

We operate quotas to limit the amount of disk space a group or a project can use on the hot storage filesystem. To check available space for all your projects:

```
[rey@ipop-up ~]$ statusBars
alphafold [###-----]      847 /    5000 GB
training  [-----]          0 /    1024 GB
```

```
[rey@ipop-up ~]$ lfsquotas training
Disk quotas for grp 6011 (gid 6011):
      Filesystem  used   quota  limit  grace  files  quota  limit  grace
 /shared/projects/training 96.26M  1T     2T     -      4      0      0      -
gid 6011 is using default file quota setting
```

Jobs **must** be launched from a project directory.



About Slurm



Slurm is the job scheduling system.

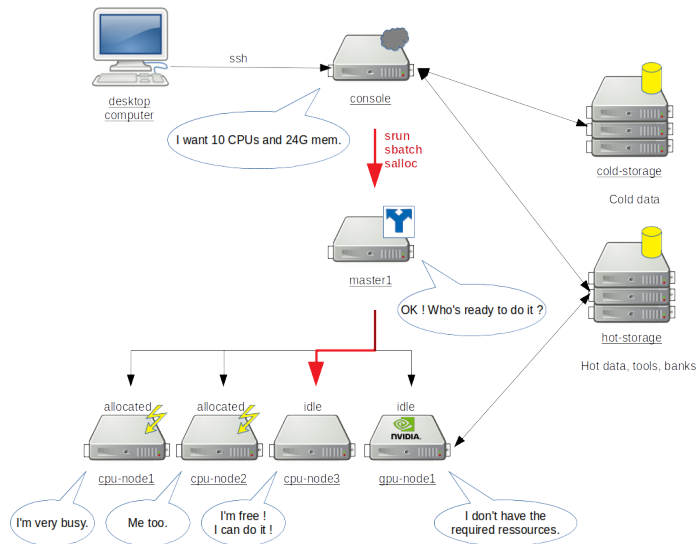
It is what will take your code and distribute it on the computing nodes, while ensuring they have the CPU(s) and RAM that you asked for.

It requires specific commands to run (srun, sbatch, salloc, etc...).

You **must** launch jobs with Slurm.



Flowsheet



srun

Launch a (simple) interactive job.

```
[rey@ipop-up ~]$ srun hostname
```

Some parameters can be added to the command line:

–**partition**/–**p**: request a specific partition

–**account**/–**A**: select the (project) account

–**cpus-per-task**/–**c**: request that ncpus be allocated (default: 1 cpu)

–**mem-per-cpu**: specify the required memory per cpu (default: 2GB)

Example:

```
[rey@ipop-up ~]$ srun -A training -p ipop-up -c 8 hostname
```



sbatch

Launch more complex jobs.

myscript.sbatch

```
#!/bin/bash
#SBATCH --partition=ipop-up
#SBATCH --account=training
#SBATCH --cpus-per-task=8
#SBATCH --mem-per-cpu=4GB
#SBATCH --output=resultat.log
hostname
```

Example:

```
[rey@ipop-up ~]$ sbatch myscript.sbatch
```



About associations

Cluster, account, partition and qos must match your user's associations.
To check your associations:

```
[rey@ipop-up ~]$ sacctmgr show user rey withassoc
```

User	Def Acct	Cluster	Account	Partition	QOS	Def QOS
rey	demo	production	training	ipop-up	normal	
rey	demo	production	demo	ipop-up	normal	
rey	demo	production	alphafold	cmpli	normal	
rey	demo	production	alphafold	rpbs	normal	

squeue

List submitted jobs on the cluster:

```
[rey@ipop-up ~]$ squeue
      JOBID PARTITION    NAME    USER  ST       TIME  NODES NODELIST(REASON)
    1008002      cmpli  argtoser domingue R    20:58:09      1 gpu-node14
    993325      cmpli  calcam-p  ghoula R 6-20:23:17      1 gpu-node8
    993574      ipop-up amyloid_  meuret PD        0:00      1 (AssocGrpCPUMinutesLimit)
943019_[165-299]      ipop-up kappa_va badaoui PD        0:00      1 (JobArrayTaskLimit)
    943019_117      ipop-up kappa_va badaoui R 7-17:55:27      1 cpu-node132
```

Some parameters can be added to filter jobs or show more infos:

- partition**/**-p**: specify the partition to view
- user**/**-u**: request jobs from a list of users
- format**/**-o**: specify the information to be displayed

Example:

```
[rey@ipop-up ~]$ squeue --me -p ipop-up
[rey@ipop-up ~]$ squeue -o "%10i %.9P %.8j %.12u %.15T %.15M %.15l %.6D %.20R %.6C %.10b %.10Q"
```



scancel

Kill a job:

```
[rey@ipop-up ~]$ scancel job_id
```



sacct

Display accounting data for your running and completed jobs:

```
[rey@ipop-up ~]$ sacct --format=JobID,JobName,Start,Elapsed,NCPUS,NodeList,ReqMeM,State --start
↩ 2023-06-01 --end 2023-06-30
```

JobID	JobName	Start	Elapsed	CPUS	NCPUS	NodeList	ReqMem	State
1010778	sleep	2023-06-08T15:37:19	00:00:30	00:00:30	1	cpu-node133	2000Mc	COMPLETED



seff

Report a job's efficiency:

```
[rey@ipop-up ~]$ seff 981654
Job ID: 981654
Cluster: production
User/Group: lejal/cmpli
State: COMPLETED (exit code 0)
Cores: 1
CPU Utilized: 04:15:58
CPU Efficiency: 99.72% of 04:16:41 core-walltime
Job Wall-clock time: 04:16:41
Memory Utilized: 243.88 GB
Memory Efficiency: 97.01% of 251.40 GB
```

Some vocabulary

A *job* consists of one or more *steps*, each consisting of one or more *tasks* each using one or more *CPUs*.

- job: A script, typically started with *sbatch*
- step: A step in the job, typically started with *srun*
- task: Requested at the job or step level, with *-array* or *-ntasks*



Job arrays

Job arrays offer a mechanism for launching a lot of tasks at the same time. Each task of the job will have the environment variable `SLURM_ARRAY_TASK_ID` set to its array index value.

`myscript.sbatch`

```
#!/bin/bash
#SBATCH --partition=ipop-up
#SBATCH --account=training
#SBATCH --output=resultat_%a.log
#SBATCH --array=1-3
case "$SLURM_ARRAY_TASK_ID" in
  1) fruit='orange';;
  2) fruit='apple';;
  3) fruit='banana';;
esac
echo $fruit
```

Job arrays

```
[rey@ipop-up ~]$ sbatch myscript.sbatch
```

Results:

```
[rey@ipop-up ~]$ ls
resultat_1.log resultat_2.log resultat_3.log
[rey@ipop-up ~]$ tail resultat_*
==> resultat_1.log <==
orange

==> resultat_2.log <==
apple

==> resultat_3.log <==
banana
```

Job arrays

Another example:

myscript.sbatch

```
#!/bin/bash
#SBATCH --partition=ipop-up
#SBATCH --account=training
#SBATCH --output=resultat_%a.log
#SBATCH --array=0-2000%50
#SBATCH --cpus-per-task=2
INPUTS=(*.fq.gz)
fastqc ${INPUTS[$SLURM_ARRAY_TASK_ID]}
```

Job arrays

```
#SBATCH --array=0-15
```

= 16 jobs (\$SLURM_ARRAY_TASK_ID: from 0 to 15 included).

```
#SBATCH --array=10-16:2
```

= 4 jobs (\$SLURM_ARRAY_TASK_ID: 10,12,14,16).

```
#SBATCH --array=2,3-7:2,11,13
```

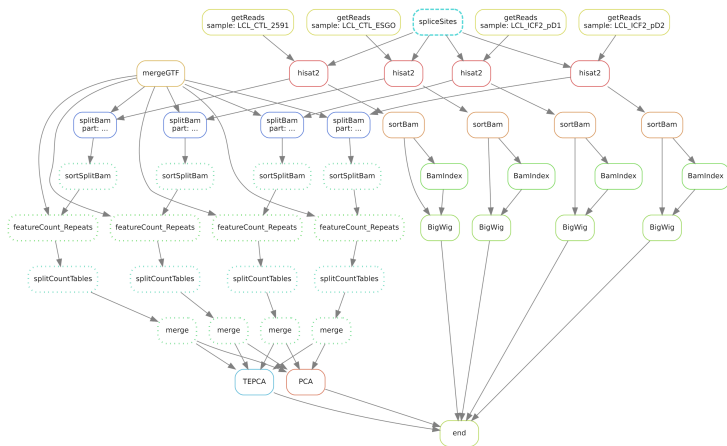
= 6 jobs (\$SLURM_ARRAY_TASK_ID: 2,3,5,7,11,13).

```
#SBATCH --array=1-10000%32
```

= 10 000 jobs, max 32 simultaneous jobs



Complex workflows



Use workflow managers such as Snakemake or Nextflow.

Environments

Tools are installed on the cluster in virtual environments:

- each tool has its own dependencies (libraries) and it's not possible to make them all coexist in the same environment
- reproducibility: some users need a specific version of a tool



Conda environments



Containers (Apptainer)

Modules

They can be loaded with the module command.

Look for the different versions of multiqc:

```
[rey@ipop-up ~]$ module avail multiqc  
multiqc/1.3  multiqc/1.6  multiqc/1.7  multiqc/1.9
```

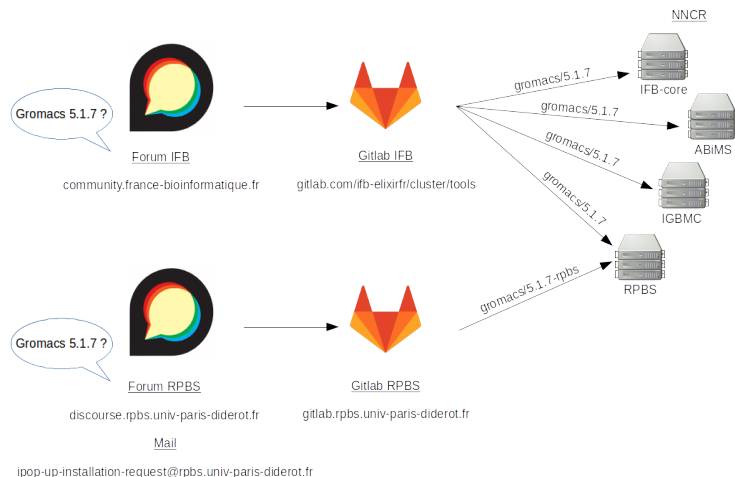
Load an environment:

```
[rey@ipop-up ~]$ module load multiqc/1.9
```

List loaded environments:

```
[rey@ipop-up ~]$ module list  
Currently Loaded Modulefiles:  
  1) multiqc/1.9      2) blast/2.13.0
```

How tools are installed



Useful resources

To find out more, the SLURM manual : `man sbatch` or
<https://slurm.schedmd.com/sbatch.html>

Ask for help or signal problems on the cluster :
<https://discourse.rpbs.univ-paris-diderot.fr/>

iPOP-UP cluster documentation:
<https://ipop-up.docs.rpbs.univ-paris-diderot.fr/documentation/>



Thanks



Alix Silvert

iPOP-UP's technical and steering committees



Exercices

<https://tinyurl.com/ipop-train>



https://parisepigenetics.github.io/bibs/cluster/training_230612/training/