ABiMS

Cluster Initiation édition

slurm workload manager

Cycle de formation 2021

Gildas Le Corguillé

Crédit: Alexandre Cormier, Camille Vacquié, Gildas Le Corguillé

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

cnrs

Biogenouest

ifb INSTITUT FRANÇAIS DE BIOINFORMATIQUE

EMBRC FRANCE
CENTRE NATIONAL DE RESSOURCES BIOLOGIQUES MARINES

OCEANOMICS

IDEALG
seaweed for the future

http://abims.sb-roscoff.fr/resources/cluster/

http://abims.sb-roscoff.fr/resources/cluster/howto#slurm

https://ifb-elixirfr.gitlab.io/cluster/doc/

- user guide
- advanced guide
- Cookbook

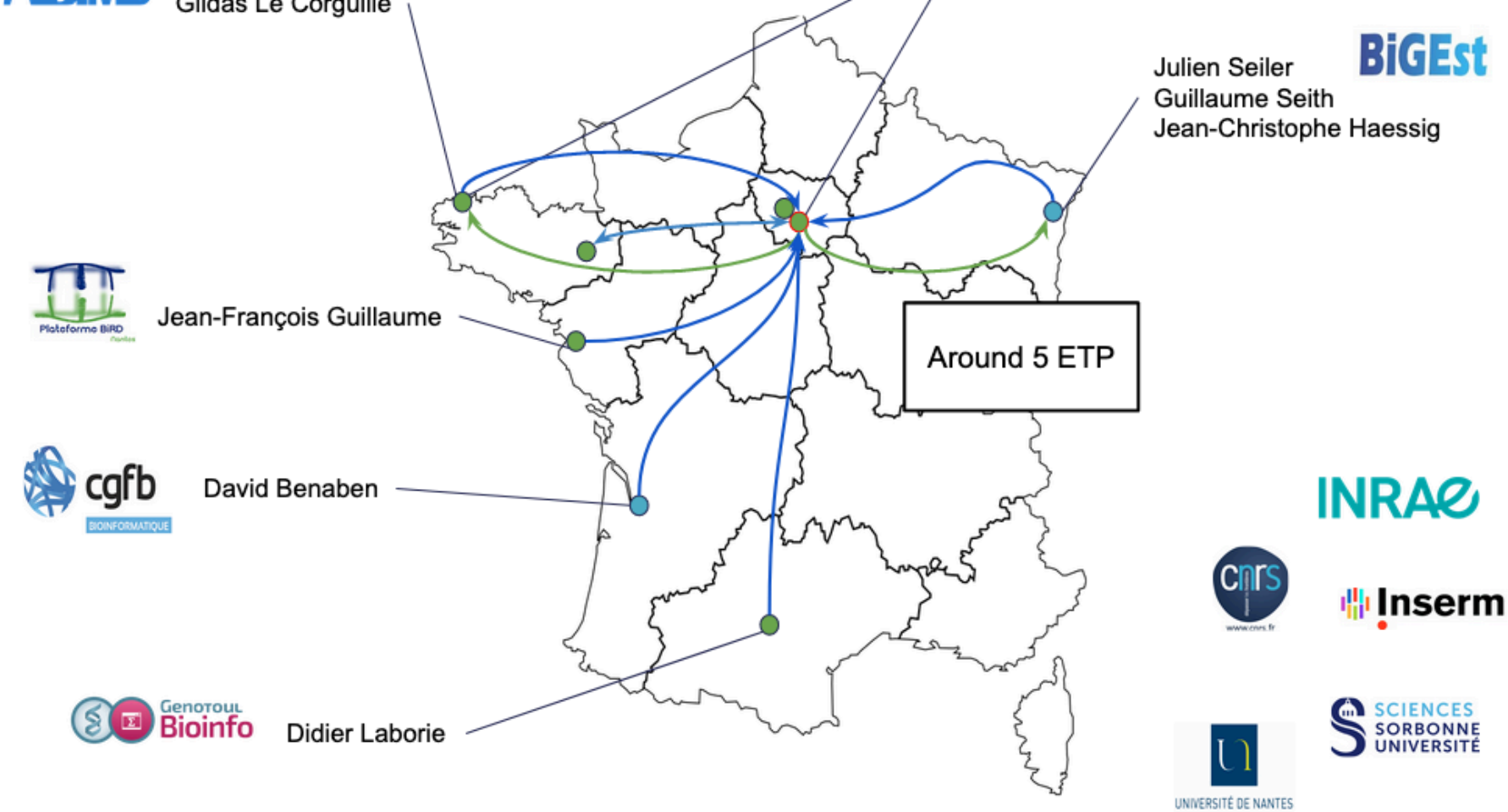
IFB NNCR Cluster - Task Force

François Gerbes
Nicole Charrière
David Christiany

Gildas Le Corguillé

Julien Seiler
Guillaume Seith
Jean-Christophe Haessig

Jean-François Guillaume



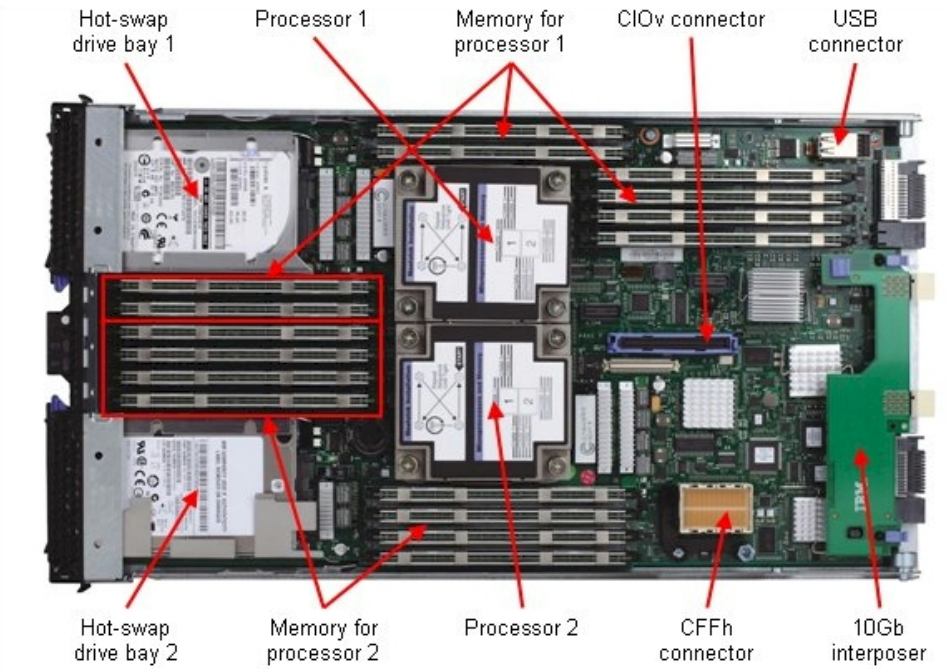
David Benaben

Didier Laborie

Principle

- Aggregation of computers / machines
  - Machine = node
- Distributed computing + shared access
- Transparent managment for users
- Community system → rules!

Hardware


Hot-swap drive bay 1
Processor 1
Memory for processor 1
CIOv connector
USB connector
Hot-swap drive bay 2
Memory for processor 2
Processor 2
CFFh connector
10Gb interposer

DISTRIBUTED COMPUTING

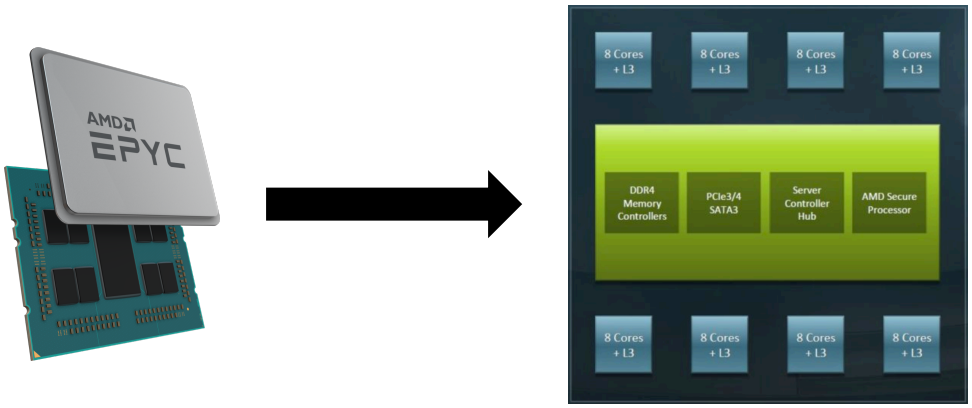
Distributed computing

- Distribution
  - Make a job as atomic as possible
  - Simple and robust
  - Linear gain

- Generate independent tasks
  - Split the data
  - Change parameters

CPU - Core CPU - Threads …

A microprocessor is a physical chip



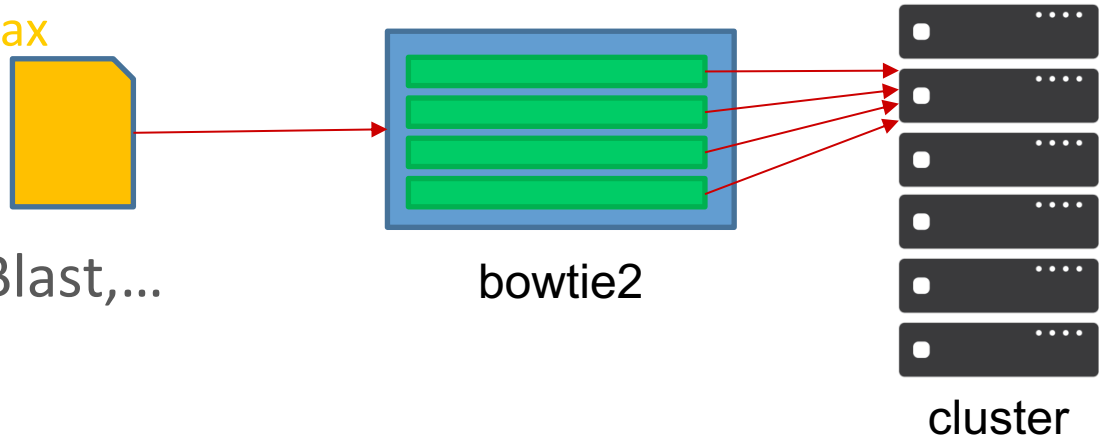
~~Core = CPU = Central Processing **Unit**~~

At ABiMS, the last 4 nodes:

1 node = 2 sockets = 2 microprocessors = 2 x 64 cores = 128 x 2 threads = 256 "CPU"

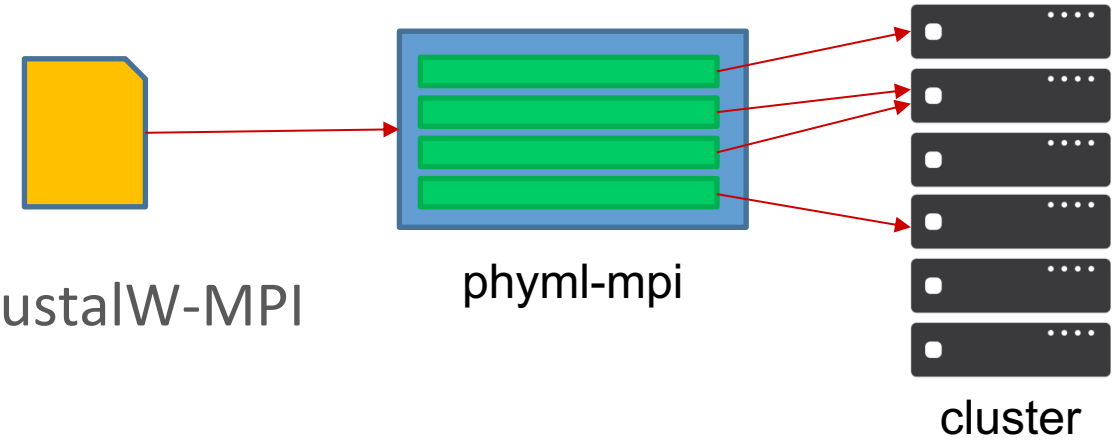
Multithreading

- Thread
  - Tasks running on the same machine but on several CPU or core
  - Shared memory
  - Nonlinear gain
    - For Blast, 4 CPU max
  - Ex: Bowtie2, CLC, Blast,…



Parallel computing

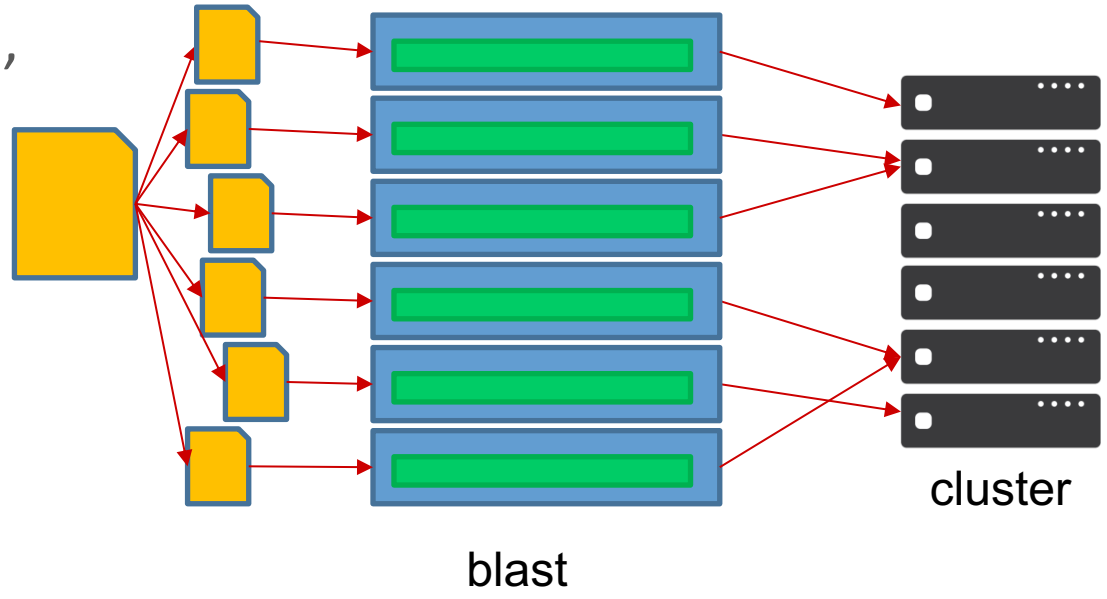
- MPI (Message Parsing Interface)
  - Tasks are running on different machines
  - Communication between tasks over the network
  - Variable gain. Nonlinear in general



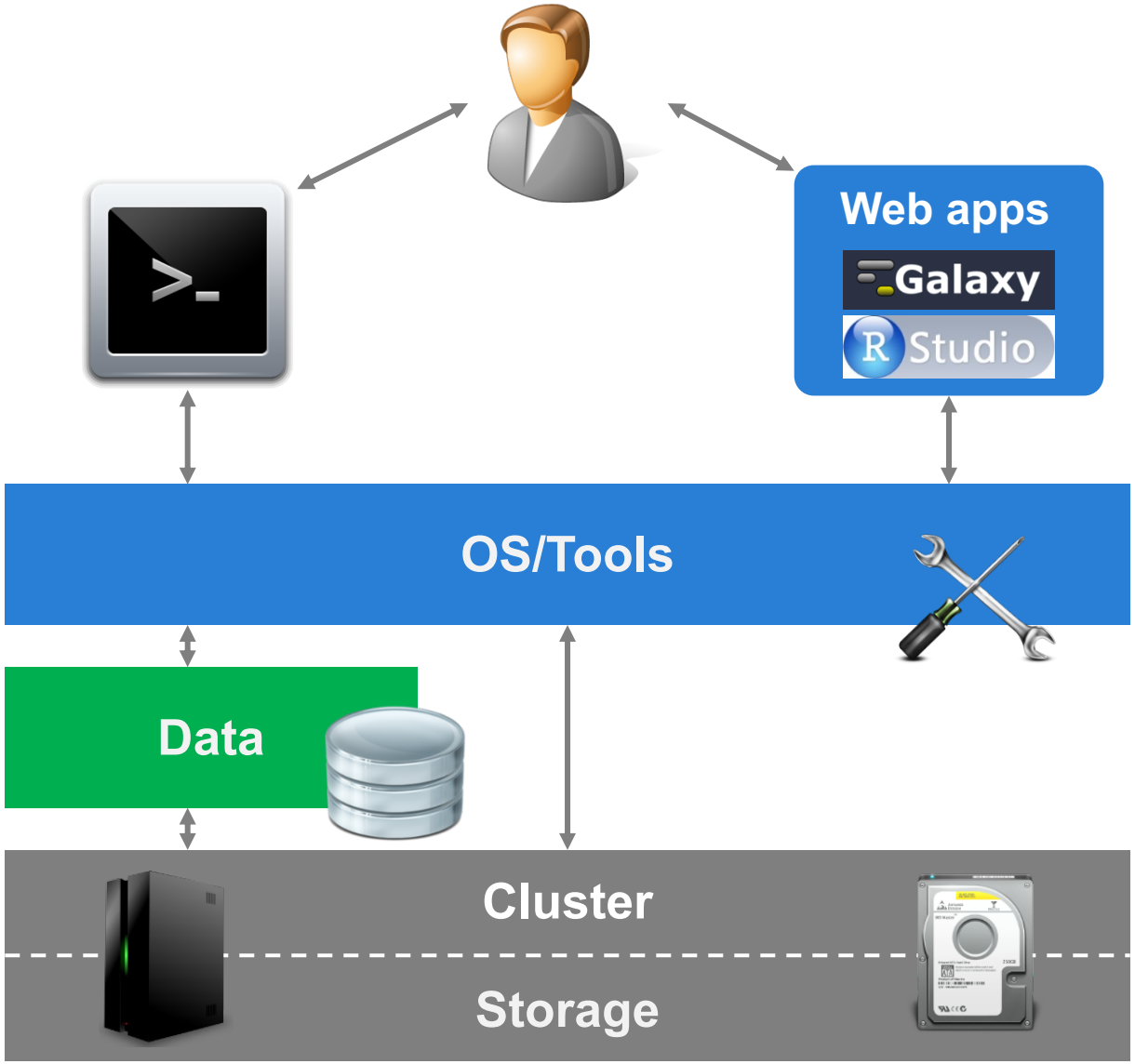
  - Ex: PhyML-MPI, ClustalW-MPI

Multithreading

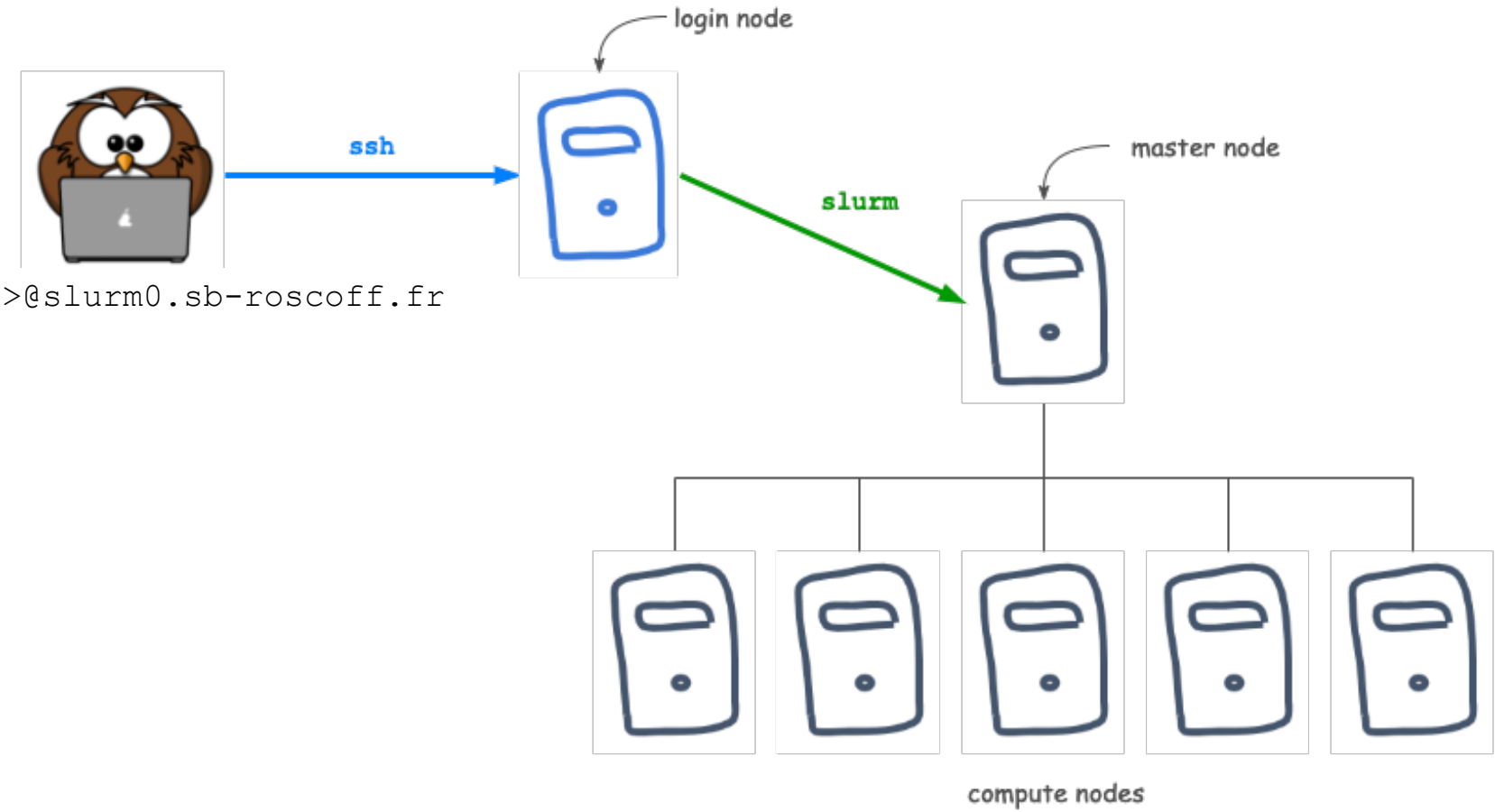
- Splited file -> Job array
  - 1 input file -> n sub-files
  - Each sub-file can be process independently
  - Linear gain
  - Ex: Blast, Diamond, Interproscan …



Environment



Cluster

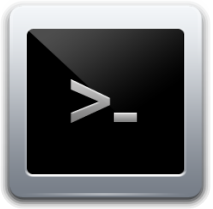



```
ssh <login>@slurm0.sb-roscoff.fr
```

CONNEXION AND STORAGE

Prerequisite

- Account
  → http://abims.sb-roscoff.fr/account
  → support.abims@sb-roscoff.fr
- Email
- X11 terminal
  - Windows: Putty or MobaXterm (X11)
  - Mac OS : XQuartz (X11)
  - Linux: integrated
- Text editor
  - Vim, nano, gedit, emacs…
- SFTP client



CNRS • SORBONNE UNIVERSITÉ
Station Biologique de Roscoff

ABiMS

Connection

```
$ ssh -Y foobar@slurm0.sb-roscoff.fr # -Y → for graphic (X11) flux redirection
```

Connection

```
$ ssh -Y foobar@slurm0.sb-roscoff.fr # -Y → for graphic (X11) flux redirection
```

```
------------------------------------------------------------

              _    ____   4  __  __  ____
             / \   | __ )  _ |  \/  |/ ___|
            / _ \  |  _ \ | || |\/| |\___ \
           / ___ \ | |_) || || |  | | ___) |
          /_/   \_\|____/ |_||_|  |_||____/

    Analysis and Bioinformatics for Marine Science
  http://abims.sb-roscoff.fr - support.abims@sb-roscoff.fr
------------------------------------------------------------

Please have a look at the training material:
 - Doc ABiMS: http://abims.sb-roscoff.fr/resources/cluster/howto#slurm
 - Doc IFB:   https://ifb-elixirfr.gitlab.io/cluster/doc/slurm_user_guide/
 - Slides:    https://ifb-elixirfr.gitlab.io/cluster/trainings/slurm/index.html
 - Video:     https://asciinema.org/a/zZrSazw5Fh7YmpHvUfvVQnzNi
 - SGE2Slurm: https://frama.link/2020-09-SGE2Slurm

IMPORTANT:
   - slurm0:   Never launch job on this server -> Use srun
   - /home:    Never launch job from this space
   - /projet:  Use your /projet folder for its performance, its volumetry
               and its independence from the /home space
   - /scratch: For your huge temporary files, please use /scratch
               but note that files older than 90 days are automatically deleted
   - installed softwares can be listed by using:
       - module load avail

CITATION: Please cite the plateform ABiMS in the Acknowledgement of your future publication

          _____  _
         / ____)| |
        ( (____  | |  _   _   ____  ____
         \____ \ | | | | | | / ___)|    \
         _____) )| | | |_| || |    | | | |
        (______/  \_)|____/ |_|    |_|_|_|

IMPORTANT -> current Slurm Reservation(s) (scontrol show reservation):
ReservationName=bbmose_7 StartTime=2020-11-02T16:40:00 EndTime=2020-11-17T12:00:00 Duration=14-19:20:00
   Nodes=n100 NodeCnt=1 CoreCnt=64 Features=(null) PartitionName=(null) Flags=SPEC_NODES
   TRES=cpu=128
   Users=(null) Accounts=bbmose Licenses=(null) State=ACTIVE BurstBuffer=(null) Watts=n/a

TIPS: Maybe, you can still submit jobs in the meantime by indicate to Slurm the smaller job duration: example with 5 days long
       sbatch --time=5-00:00:00 -p bigmem myscript.sbatch

Last login: Mon Nov  9 14:13:39 2020 from 78.218.174.71
[lecorguille@slurm0 ~]$ 4
```

Important

WORKING DIRECTORIES

Data

- Personal data → /home

- Scientific data:
  - ~~By team / group~~
  - ~~By UMR~~
  - Project
  - Public data

- Databank
  - Genbank, Uniprot, InterPro banks, etc.
  - Format/Index : Fasta, Blast, Bowtie2, BWA, Daemon, etc.
  - Private & Public

Workspace

project

~~→ per person~~
~~→ by team~~
→ by subject/study

home

→ only for connexion (Environment variable)

db/bank

→ Databank (Blast, Genbank, Interpro…)

nz11 - storage server

Workspace

scratch2 → nz12

50 To

Space for all the primary analysis - generated huge amount of temporary/useless files

- **Mutualised storage** between all users
- Data are **not backed up**
- All files older than **90 days** are automatically **deleted**

When I'm connecting, I arrive in my:

home

```
$ pwd #print working directory

/home/fr2424/sib/foobar
```

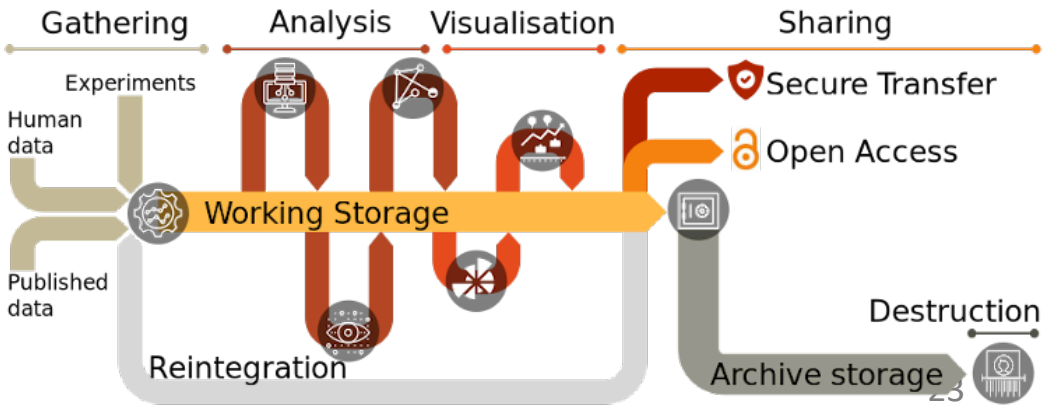
Not for storage / computing

/shared/projects/* directories

project

General issues:

- To ensure good data lifecycle management with our users, we strive to identify the projects that are processed on the cluster.
- We want to avoid large unfinishable project folders with a mess inside (sub-sub-projects).
- A project has a perimeter: a subject, a goal, a beginning, an end... A typical project should take place in a window of 1 to 3 years. We can consider the project as a funding grant (obtained or not :/).
- Note that if you manage several parallel projects, it is quite possible for you to request more project spaces with their own lifetime and list of contributors.



https://ifb-elixirfr.gitlab.io/cluster/doc/project/

/shared/projects/* directories

- If existing projects, we are migrating the current project folders:
  - /projet/umr7144/abice/toto
  - /projet/sbr/mein_grosses_projekt/unterprojekt1 …

  To -> /shared/project/

- A good project name:
  - The ideal name for a project is its ANR acronyme.
  - But you can compose with: the species, the response being studied, the technologies or methods involved
  - The ideal name isn't too long :)
  - Avoid: your name, the name of your laboratory or team, ...

NOTE : to submit jobs through Slurm you need at least one Slurm account. The Slurm accounts stick to the project names.

Storage limitation

Regularly, check the disk usage of my project to prevent saturation.

```
$ du -sh * #size of each file/folder -> who is the biggest?

  68G     assembly
  341G    pagit
  3,8G    remapping
  12K     cache_tmp
  17M     chr_similarity
  1008M   galaxy_dataset
  669M    metrics
  2.1M    Tes

$ du -sh assembly/*

  11G     assembly/transcriptome_V1
  9.8G    assembly/transcriptome_V2
  48G     assembly
```

```
    admin_check_tools [--------------------]       5 /   10000 hCPU
                      [--------------------]       0 /  476837 MB
              ebaii2020 [####################] 1156725 /   10000 hCPU
                      [#-------------------]   41504 /  476837 MB
     fair_training2020 [#################---]  148419 /  166666 hCPU
                      [--------------------]     228 /  476837 MB
                gildas [##########----------]    5286 /   10000 hCPU
  project_glecorguille [--------------------]       0 /   10000 hCPU
                      [--------------------]    2482 /  476837 MB
              taskforce [####################] 361498 /   10000 hCPU
                      [--------------------]   12663 /  476837 MB
Last update: 2020-11-09 00:08 - default account in bold
```

At the login

Management of my project directory

Compress your data!

```
$ ll -h

  -rw-rw-r--+ 1 foobar    sib  26G mars  8 08:16 140220_SND393_B_L006_GPO-12_R1.fastq
  -rw-rw-r--+ 1 foobar    sib  26G mars  8 08:16 140220_SND393_B_L006_GPO-12_R2.fastq

$ gzip 140220_SND393_B_L006_GPO-12_R1.fastq
$ gzip 140220_SND393_B_L006_GPO-12_R2.fastq

$ ll -h

  -rw-rw-r--+ 1 foobar    sib 7,7G mars  7 12:25 140220_SND393_B_L006_GPO-12_R1.fastq.gz
  -rw-rw-r--+ 1 foobar    sib 7,9G mars  7 12:29 140220_SND393_B_L006_GPO-12_R2.fastq.gz
```

Some softwares are capable to use directly compressed data (TopHat2, Trimmomatic,…)

RESOURCES

Software

History:

- 2007-2019 `export PATH=/usr/local/genome[2]/bin:$PATH`
- 2017-2020 `source $CONDA3/activate blast-2.9.0`
- 2020- `module load blast/2.9.0`

CentOS 6/7

ubuntu

`module load last/2.9.0`

ENVIRONMENT MODULES

BIOCONDA®

by default

singularity

if a licence must be accepted
if not in Bioconda and hard to integrate or urgent
if a Docker container exists

`module load _genome2-legacy`

Software

In practice:

```
$ module avail
------------------------------------------------------------------------------ /shared/software/modulefiles ------------------------------------------------------------------------------
_genome2-legacy          charger/0.5.2          fastqc/0.11.5              isoseq3/3.3.0          mosdepth/0.2.6          picrust/1.1.3              sample/3.5.1          subread/1.6.1
abyss/2.2.1              charger/0.5.4          fastqc/0.11.7              java-jdk/8.0.112       mothur/1.41.0           picrust2/2.1.1_b           samtools/1.3.1        subread/2.0.1
adxv/1.9.14              circlator/1.5.5        fastqc/0.11.8              jpredapi/1.5.6         mothur/1.44.11          pilon/1.23                 samtools/1.5          swarm/3.0.0
anvio/6.1                clustalo/1.2.4         fastqc/0.11.9              kaiju/1.7.3            mrbayes/3.2.6           pindel/0.2.5b9             samtools/1.9          targetp/2.0
arcs/1.1.0               compilers/1.0.4        fasttree/2.1.10            kallisto/0.44.0        msprime/0.7.3           pipmir/1.1                 samtools/1.10         texlive/20180414
augustus/3.3.3           concoct/1.1.0          fastx_toolkit/0.0.14       keras/2.2.4            multiqc/1.6             plink/1.90b4               scikit-allel/1.2.1    tmux/3.1
bam2fastx/1.3.0          coreutils/8.25         filtlong/0.2.0             kraken2/2.0.9beta      multiqc/1.7             porechop/0.2.3_seqan2.1.1  scikit-learn/0.22.1   toil/3.14.0
bc/1.07.1                crossmap/0.3.9         flye/2.6                   lima/1.11.0            multiqc/1.9             ppanggolin/1.0.1           screen/4.8.0          topaz/0.2.4
bcftools/1.9             cryolo/1.7.4-gpu       freebayes/1.2.0            links/1.8.7            music/1.0.0             prokka/1.14.6              seqtk/1.3             tophat/2.1.1
bcftools/1.10.2          csvkit/1.0.3           frogs/3.1.0                lordec/0.9             nemo-age/0.29.0         psipred/4.01               shortstack/3.8.5      trim-galore/0.5.0
bcl2fastq/2.20.0         cufflinks/2.2.1        gappa/0.5.1                lumpy-sv/0.3.0         nextflow/18.10.1        pyega3/3.0.39              sickle-trim/1.33      trim-galore/0.6.5
bedtools/2.26.0          cutadapt/1.8.3         gatk4/4.0.10.0             macs2/2.1.1.20160309   nextflow/20.04.1        pypy/2.7-5.10.0            signalp/5.0           trimal/1.4.1
bedtools/2.27.1          cutadapt/1.10          gatk4/4.1.4.1              macs2/2.2.7.1          obitools/1.2.11         pypy/3.5-7.0.0             simka/1.5.2           trimmomatic/0.36
bedtools/2.29.2          cutadapt/2.10          gatk4/4.1.7.0              mafft/7.407            obitools/3.0.0_beta5    pyscenic/0.9.19            singularity           trimmomatic/0.39
berokka/0.2.3            dadi/2.0.5             gautomatch/0.56_sm62_cu8.0 maker/2.31.10          openmpi/4.0.4           pyscenic/0.10.0            snakemake/5.3.0       trinity/2.8.4
bio-vcf/0.9.2            deeptools/2.5.3        gblocks/0.91b              manta/1.4.0            orffinder/0.4.3         python/2.7                 snakemake/5.7.4       trinity/2.8.5
$ module load trinity/2.8.4 fastqc/0.11.7
$ module load snakemake/5.3.0

$ module unload trinity/2.8.4
$ module purge
```

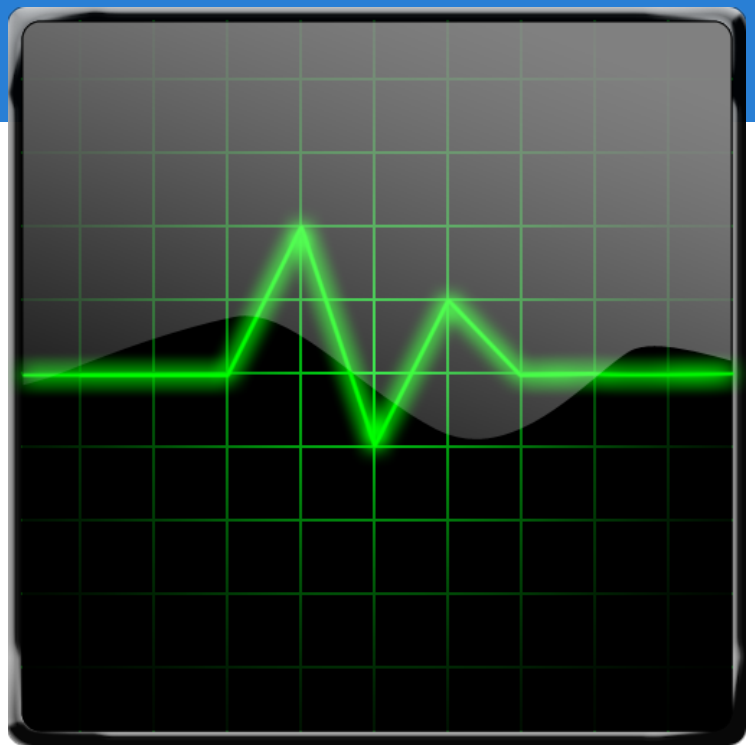
https://ifb-elixirfr.gitlab.io/cluster/doc/module/

Genomes databank

- /shared/bank
  - Public databank:
    - NCBI
    - GenBank
    - UniProt
    - InterPro
    - Etc
  - SBR databank
    - Start with the prefix “sbr_”
    - Description of these databank is currently in progress

BioMAJ

Or not



How to use the cluster?

SLURM

Job management system

- SLURM
  - **S**imple **Li**nux **U**tility for **R**esource **M**anagement
  - Scheduler in charge of the jobs management
  - User interface for submitting and controlling jobs

- Task scheduling
  - Resources allocation
  - Nodes load
  - Priority
- Management policy and resource sharing
  - CPU / Memory
  - Execution time
- Reporting and errors
  - History
  - Usage statistics

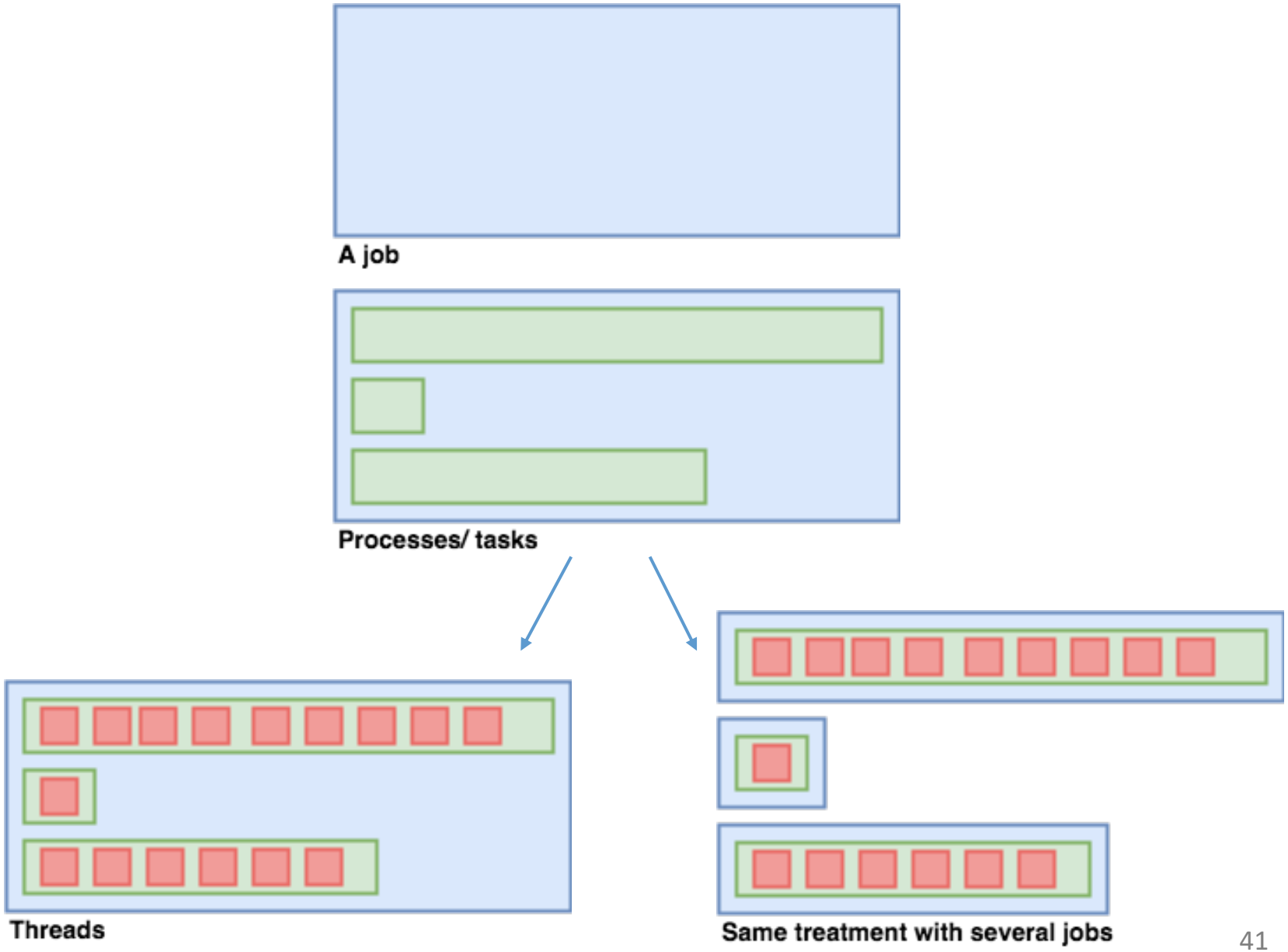
Concepts

- **Account**: A logical group of users
  - Resource consumption is associated with an account.
    On the ABiMS cluster an account is created for each project.
    You may have access to multiple accounts (if you have several projects).
    Your first project will be your default account.
- **Resources**: nodes, CPUs, memory

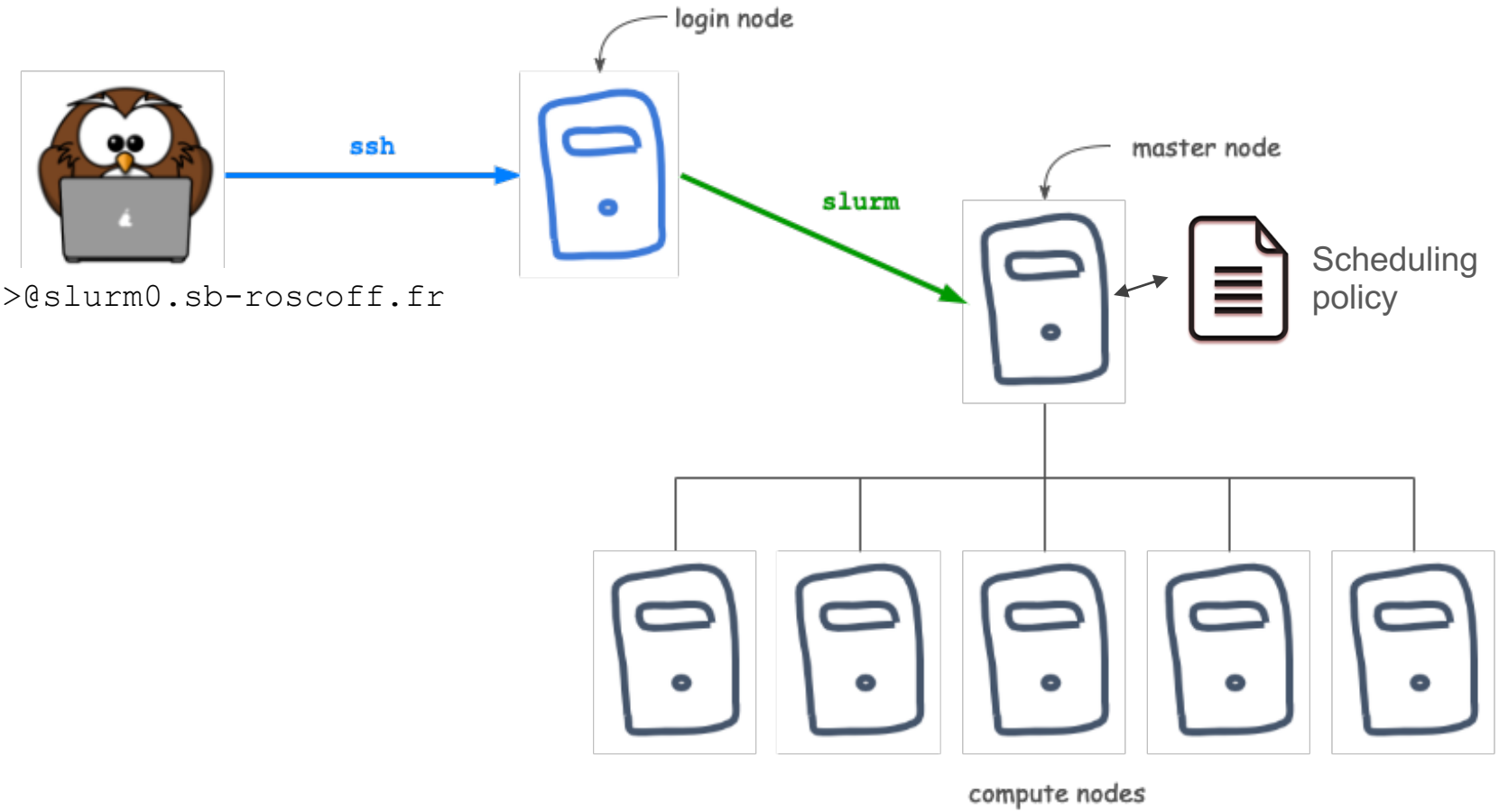
Concepts

- **Job**
  - **User's point of view**: a calculation or data analysis
  - It could be a single program or a full pipeline (a succession of programs).
  - **Slurm's point of view**: an allocation of resources
- **Job steps**
  - The processes that actually do the real work
- **Process/Task**
  - An instance of a computer program executed by one or many threads
- **Thread**
  - A thread of execution is the smallest sequence of programmed instructions that can be executed

Concepts


A job
Processes/ tasks
Threads
Same treatment with several jobs

Job management system




```
ssh <login>@slurm0.sb-roscoff.fr
```

Resources

The list of resources:

- The nodes
- The memory
- The CPU/Core
- The partition
- The time

Your priority

Resources | Nodes

```
$ sinfo
$ sinfo -Nel
$ sinfo -Nel --format "%.8N %.9R %.4c %.7z %.7m"
```

TIPS: alias

```
NODELIST PARTITION    STATE CPUS CPU_LOAD   S:C:T FREE_MEM  MEMORY WEIGHT REASON
    n75      fast    mixed  254    48.03 2:127:1    44262  257445      1 none
    n82      fast    mixed   30     7.95  2:15:1      466  128882      1 none
    n84      fast    mixed   30    24.00  2:15:1     1900  128882      1 none
    n84      long    mixed   30    24.00  2:15:1     1900  128882      1 none
    n86      fast    mixed   30    24.00  2:15:1    25744  128882      1 none
    n86      long    mixed   30    24.00  2:15:1    25744  128882      1 none
    n88      fast    mixed   30    24.00  2:15:1    56542  128882      1 none
    n88      long    mixed   30    24.00  2:15:1    56542  128882      1 none
    n89      fast    mixed   30    24.00  2:15:1      788  128882      1 none
    n89      long    mixed   30    24.00  2:15:1      788  128882      1 none
    n94      fast     idle   30     0.00  2:15:1   118199  128882      1 none
    n94      long     idle   30     0.00  2:15:1   118199  128882      1 none
    n95      fast     idle   30     0.00  2:15:1   100980  128882      1 none
    n95      long     idle   30     0.00  2:15:1   100980  128882      1 none
    n96      fast draining  254     1.00 2:127:1   118856  257445      1 none
    n96      long draining  254     1.00 2:127:1   118856  257445      1 none
    n97      fast  drained  254     1.00 2:127:1   133272  257445      1 none
    n97      long  drained  254     1.00 2:127:1   133272  257445      1 none
    n98      fast    mixed  254     1.00 2:127:1   133206  257445      1 none
    n98      long    mixed  254     1.00 2:127:1   133206  257445      1 none
    n99   bigmem     idle   80     0.00  4:10:2   139285 1032076     10 none
   n100   bigmem reserved  128     0.13  4:16:2   389702 2064169     10 none
   n115      fast    mixed   46    39.47  2:23:1     3739  257859      1 none
```

Resources | Memory `--mem`

`--mem=<size[units]>`

Specify the real memory required per node.

The default units is MB (Default: 2GB)

The job is killed if it exceeds the limit

TIPS: You can learn from your past jobs
See: `sacct` section

Note that you can use the variable `$SLURM_MEM_PER_NODE` in the command line to synchronize the software settings and the resource allocated.

https://ifb-elixirfr.gitlab.io/cluster/doc/slurm_user_guide/#parameters-to-control-the-job

Resources | CPU

`--cpus-per-task=<ncpus>, --cpus, -c`

Request a number of CPUs (default 1)

Note that you can use the variable `$SLURM_CPUS_PER_TASK` in the command line to avoid mistake between the resource allocated and the job.

Coming soon: cgroups will limit your job within the number of cpus you request, even if you use more than that

https://ifb-elixirfr.gitlab.io/cluster/doc/slurm_user_guide/#parameters-to-control-the-job

Resources | Partitions `--partition`

`--partition, -p`

```
$ scontrol show partition
$ sacctmgr list qos format=Name,Priority,MaxTRESPU%20
```

En résumé

	Time out	Default resources	Max resources / user	Purpose
fast	24 hours	--cpu-per-task 1 --mem 2GB	cpu=200, mem=750GB	Regular jobs
long	30 days	--cpu-per-task 1 --mem 2GB	cpu=200, mem=750GB	Long jobs
bigmem	60 days	--cpu-per-task 1 --mem 2GB	mem=1.9TB	On demande. For treatements requiring a lot of RAM

Default:
- fast
- 1 CPU
- 2 GB de RAM

http://abims.sb-roscoff.fr/resources/cluster/

Resources | Time `--time`

`--time=<time>, -t`

Not mandatory
But recommanded

Set a limit on the total run time of the job allocation.

The job is killed if it exceeds the limit

Acceptable time formats include:

`"minutes", "minutes:seconds", "hours:minutes:seconds", "days-hours", "days-hours:minutes" and "days-hours:minutes:seconds".`

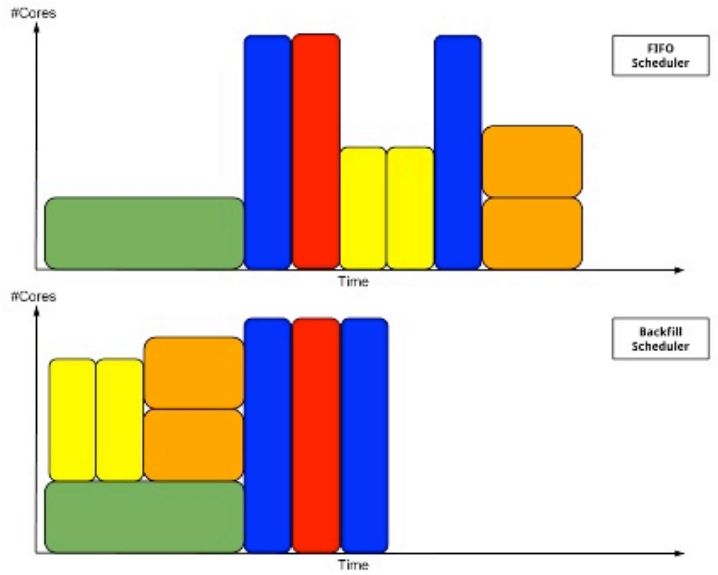
https://ifb-elixirfr.gitlab.io/cluster/doc/slurm_user_guide/#parameters-to-control-the-job

Your FairShare / Priority

JobPriority = PriorityWeightAge * AgeFactor
+ PriorityWeightFairshare * FairShareFactor
+ PriorityWeightJobSize * JobSizeFactor
+ PriorityWeightPartition * PartitionFactor
+ PriorityWeightQOS * QosFactor

Slurm as a Batch Scheduler

Slurm BackFill Concept



Ref: https://wiki.auckland.ac.nz/pages/viewpage.action?pageId=88902184

TIPS:

`--time` interests:

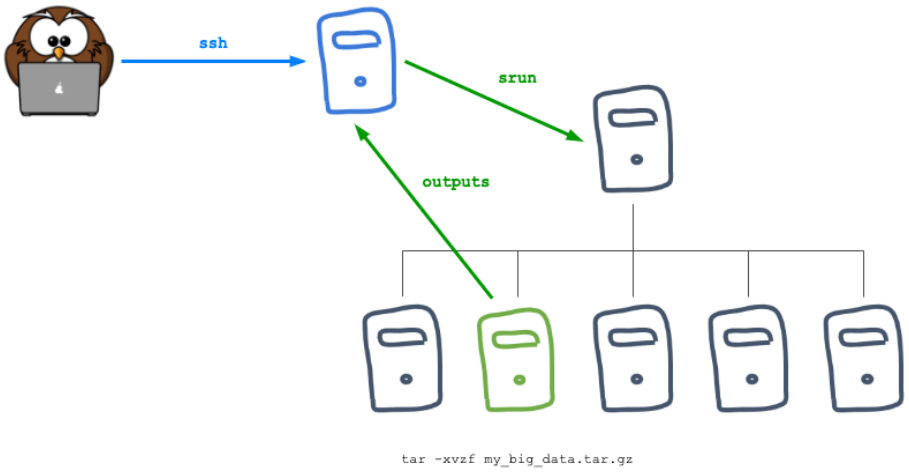
- Allow BackFill
- In case of a planned maintenance or reservation

Jobs type

"Interactive" mode: `srun`

- Short/Quick job
- and/or development
- Prerequisite: none

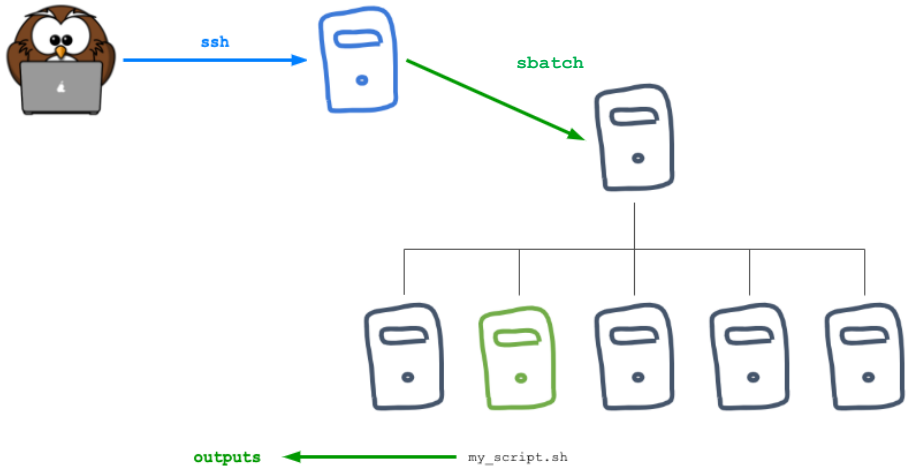
The job is killed if the terminal is closed or the network is cut off

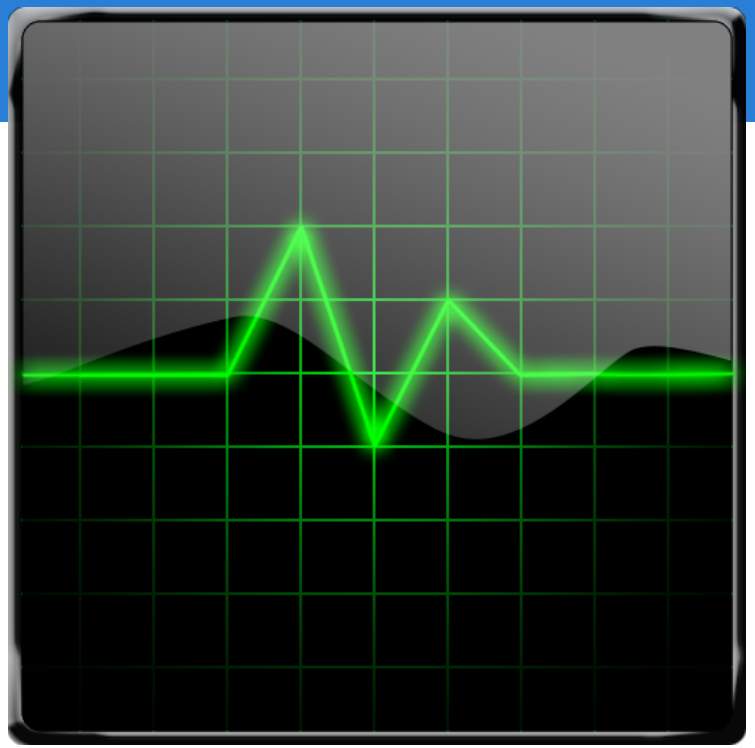


Batch mode: `sbatch`

- Heavy jobs
- Prerequisite: text editor
- One script per job

Better for reproducibility because it's self documented



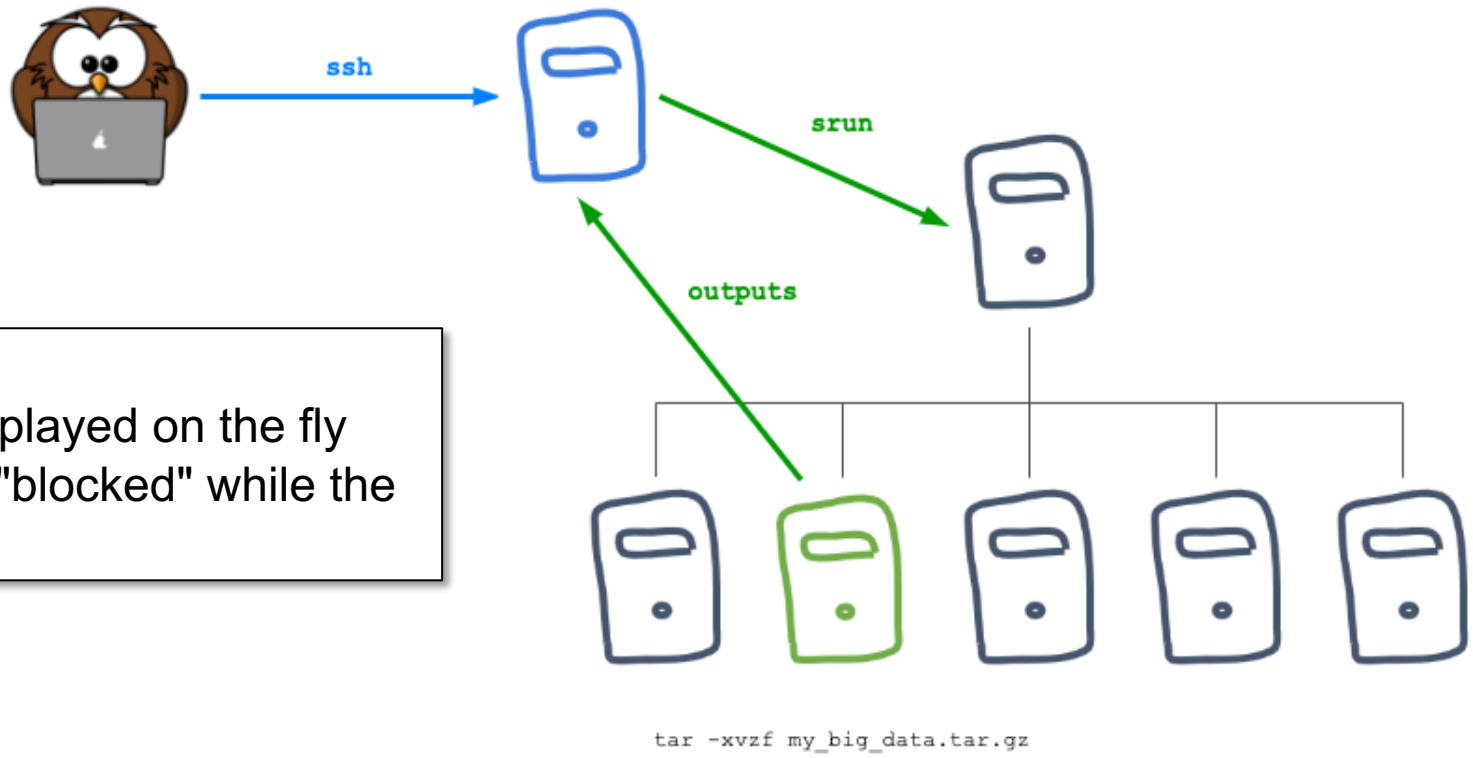


srun

SLURM

srun

```
$ srun ln -s Ec597.fsa Ec597_v4.1.fsa
ln: failed to create symbolic link 'Ec597.fsa': File exists

$ srun tar -xvzf my_big_data.tar.gz
big_file1
big_file2
```



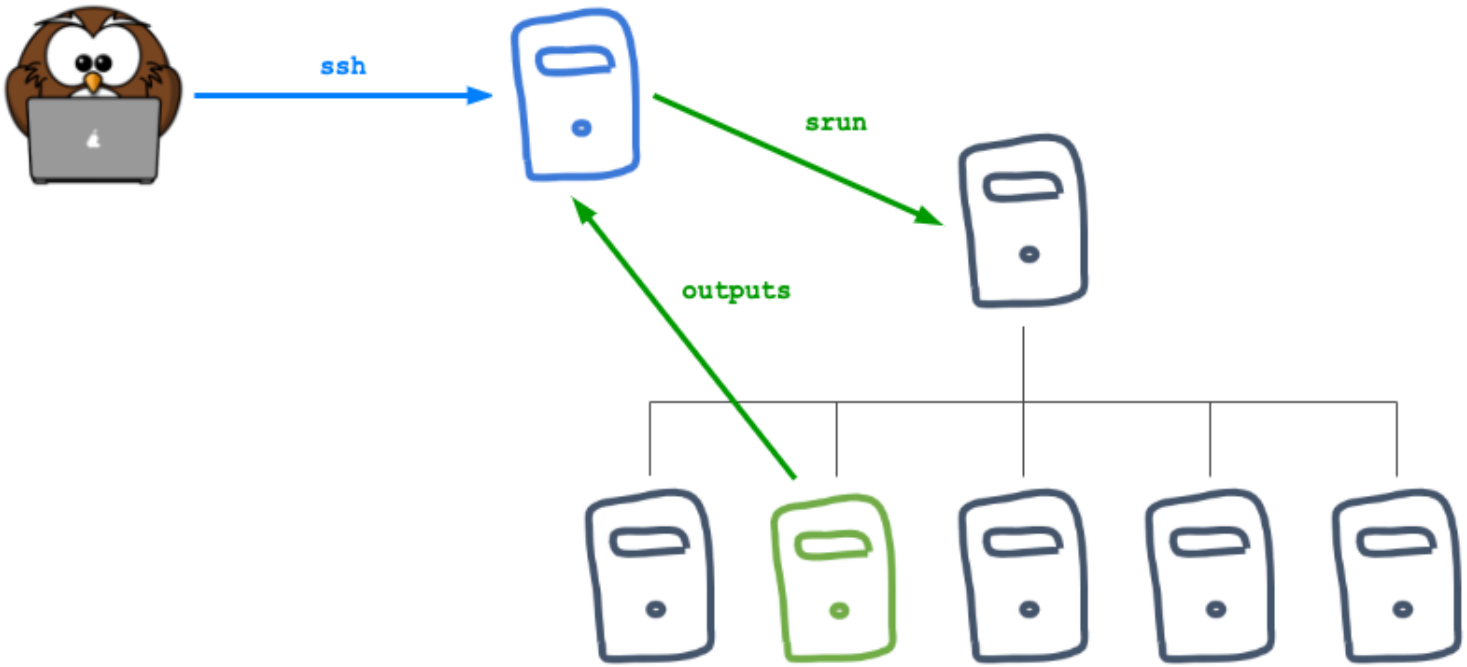
To know:
- Outputs are displayed on the fly
- The console is "blocked" while the job is running

srun

```
$ srun --mem 4GB fastqc *.fq.gz
```

Common parameters:
`--cpus-per-tasks`: number of CPU
`--mem`: memory for the whole job
`--mem-per-cpu`: memory per CPU

Default settings:
`--cpus-per-tasks=1`
`--mem=2GB`




```
tar -xvzf my_big_data.tar.gz
```

srun | interactif

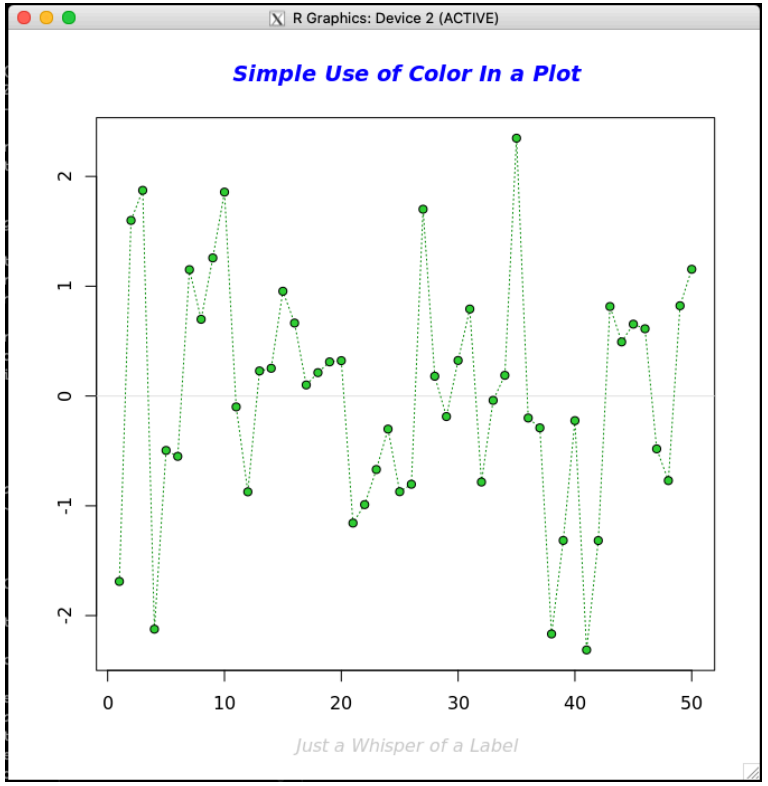
`--pty`: pseudo terminal mode

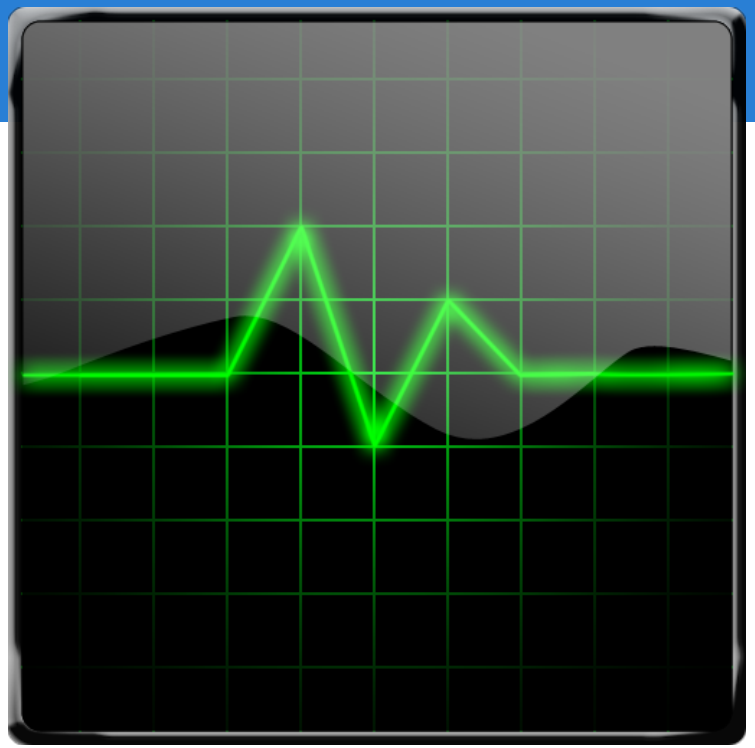
To know:
- Outputs are displayed on the fly
- ~~The console is "blocked" while the job is running~~
- Allow interaction like prompt

```
[slurm0]$ srun --pty --cpus-per-tasks 4 R

R version 4.0.2 (2020-06-22) -- "Taking Off Again"
> 1+1

[slurm0]$ srun --pty bash
[n115]$


[slurm0]$ srun --pty seqret
Read and write (return) sequences
Input (gapped) sequence(s):
```

Sinteractive for graphical application

```
[slurm0]$ module load r/4.0.3
[slurm0]$ sinteractive
[n97]$ R
> demo(graphics)
> opar <- par(bg = "white")
> plot(x, ann = FALSE, type = "n")
```



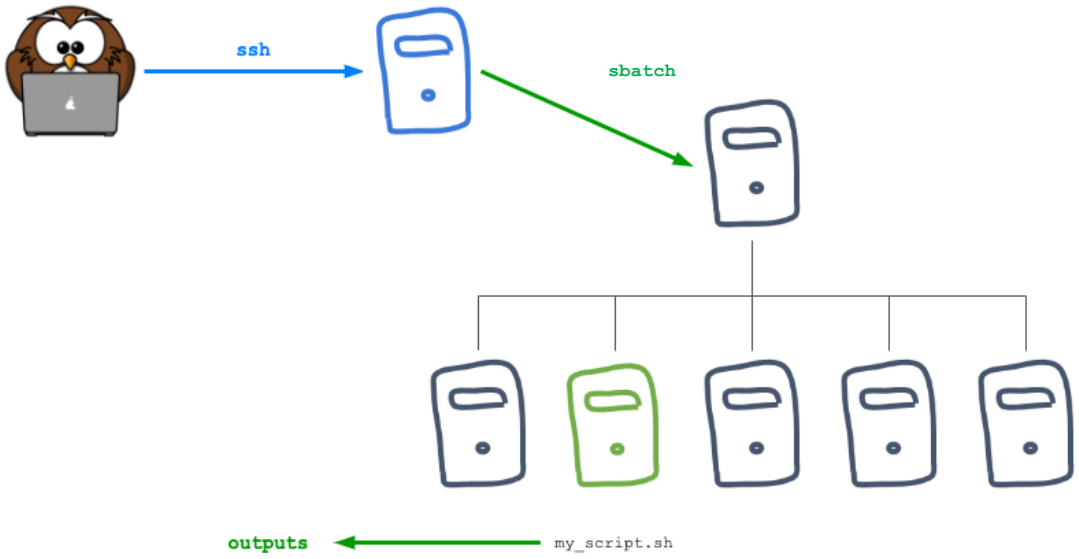


sbatch

SLURM

sbatch: Batch mode

- Progress:
  - Script edition
  - Choose the right partition
  - Submitting → Execution → Results
- Edition
  - In command line: vi, vim, nano…
  - In graphic mode: gedit, kate…



How to?

1. Prepare script of executable commands
2. Submit to batch system
3. Use the job ID for job control (query status, cancel, ...)
4. Check the job status (no execution error)

sbatch script

The minimum

script.sbatch

Header
cmd lines

```
#!/bin/bash

echo "Hello world!" > output.txt
```

Essential for sbatch:

- The header:
  - Shell path: can be bash, python …
- The command line(s)

sbatch script

Other practical settings

script.sbatch

```
#!/bin/bash
#SBATCH -o slurm-%N-%j.out
#SBATCH -e slurm-%N-%j.err
# This a comment

module load trinity/2.11.0
Trinity --single rep1.fq.gz
```

- -o stdout filename with the Node name and the Job ID
- -e stderr filename

By default, 2 files are generated:

- **slurm-%j.out**
- **slurm-%j.err**

sbatch script

Other practical settings

script.sbatch

```
#!/bin/bash
#SBATCH --mail-user foo.bar@sb-roscoff.fr
#SBATCH --mail-type END


module load trinity/2.11.0
Trinity --single rep1.fq.gz
```

```
--mail-type= NONE, BEGIN, END, FAIL, REQUEUE, ALL
```

sbatch script

Other practical settings

By default
`--mem=2G`
`--cpus-per-task=1`

script.sbatch

```
#!/bin/bash
#SBATCH --mem=100G
#SBATCH --cpus-per-task=16

module load trinity/2.11.0
Trinity --single rep1.fq.gz --max_memory 100G --CPU 16
```

script.sbatch

```
#!/bin/bash
#SBATCH --mem=100G
#SBATCH --cpus-per-task=16

module load trinity/2.11.0
Trinity --single rep1.fq.gz --max_memory $SLURM_MEM_PER_NODE --CPU $SLURM_CPUS_PER_TASK
```

sbatch script

Othe

By default

sk=1

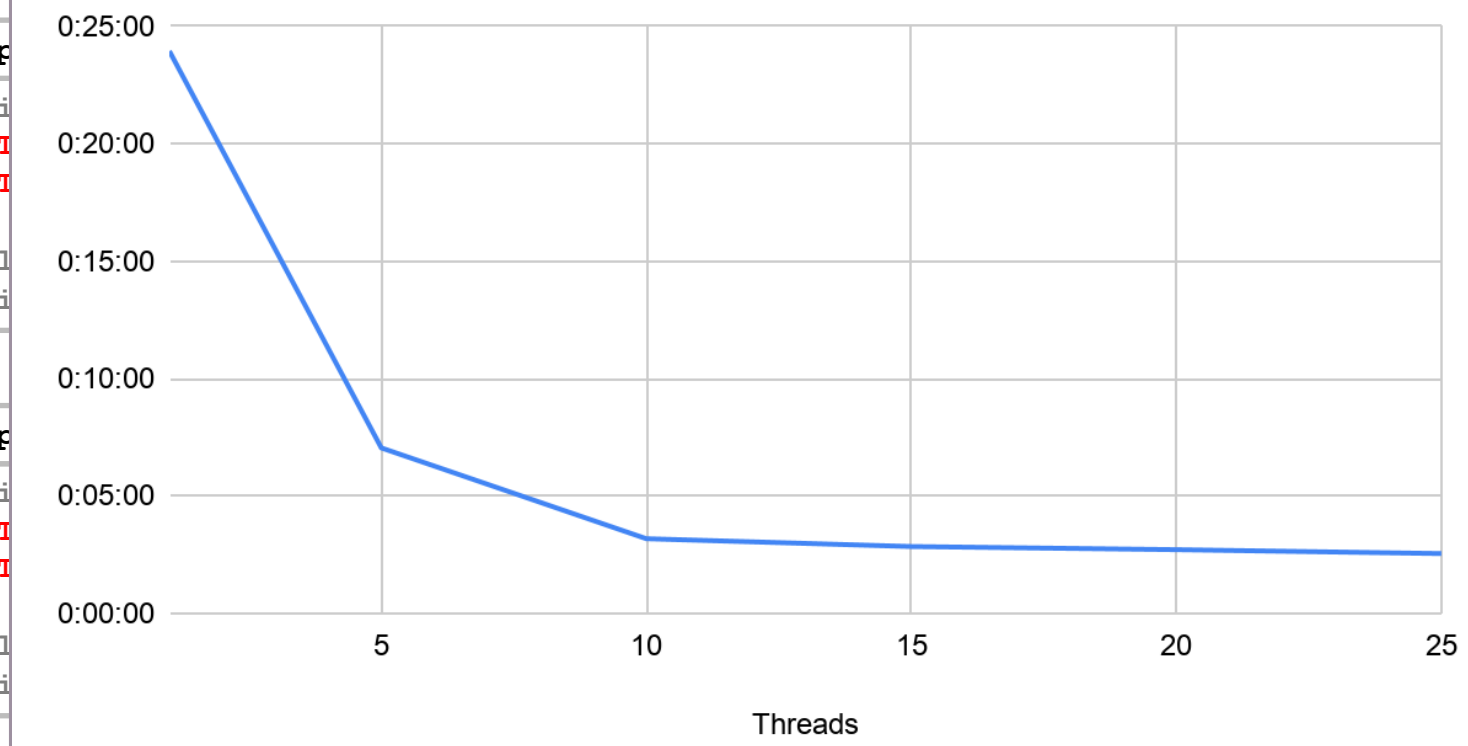
scrip

```
#!/bi
#SBAT
#SBAT

modul
Trini
```

scrip

```
#!/bi
#SBAT
#SBAT

modul
Trini
```

R_TASK



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

sbatch script

Other practical settings

script.sbatch

```
#!/bin/bash
#SBATCH --partition=long

module load trinity/2.11.0
Trinity --single rep1.fq.gz
```

```
--partition = fast|long|bigmem
```

	Time out	Default resources	Max resources / user	Purpose
fast	24 hours	--cpu-per-task 1 --mem 2GB	cpu=200, mem=750GB	Regular jobs
long	30 days	--cpu-per-task 1 --mem 2GB	cpu=200, mem=750GB	Long jobs
bigmem	60 days	--cpu-per-task 1 --mem 2GB	mem=1.9TB	On demande. For treatements requiring a lot of RAM

Submit the batch script

At runtime

```
$ sbatch -p fast --mem 100G script.sbatch
  Submitted batch job 203738
```

Or within the sbatch script

script.sbatch

```
#!/bin/bash
#SBATCH --partition fast
#SBATCH --mem 100G

module load trinity/2.11.0
Trinity --single rep1.fq.gz --max_memory $SLURM_MEM_PER_NODE
```

TIPS:
This method is better for reproducibility

```
$ sbatch script.sbatch
  Submitted batch job 203739
```

Submit the batch script

slurm@slurm-controller.sb-roscoff.fr 22:13
SLURM Job_id=203738 Name=test.sbatch Ended, Run time 00:00:02, COM...
Ce message est vide.

slurm@slurm-controller.sb-roscoff.fr 22:13
SLURM Job_id=203738 Name=test.sbatch Began, Queued time 00:00:00
Ce message est vide.

Monitoring / Checking / Control

```
[slurm0 ~]$ squeue -u lecorguille
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            200845    bigmem rnaspade    corre PD       0:00      1 (QOSMaxMemoryPerUser)
56616_[401-500%20]      long   MLboot   ekayal PD       0:00      1 (JobArrayTaskLimit)
            200844    bigmem rnaspade    corre  R 1-00:14:07      1 n99
          59250_2      long       pb   ekayal  R 1-11:02:54      1 n97
          59250_3      long       pb   ekayal  R 1-11:02:54      1 n98
          59250_1      long       pb   ekayal  R 1-11:03:49      1 n96
        56616_382      long   MLboot   ekayal  R 1-11:03:49      1 n89
        56616_383      long   MLboot   ekayal  R 1-11:03:49      1 n90
        56616_384      long   MLboot   ekayal  R 1-11:03:49      1 n90
```

TIPS

```
[lecorguille@slurm0 tmp]$ alias squeue
alias squeue='squeue --format "%.18i %.9P %.8j %.8u %.2t %.10M %.5C %.8m %.6D %R" '

[lecorguille@slurm0 tmp]$ squeue
     JOBID PARTITION     NAME     USER ST       TIME  CPUS MIN_MEMO  NODES NODELIST(REASON)
    200845    bigmem rnaspade    corre PD       0:00     1     400G      1 (QOSMaxMemoryPerUser)
401-500%20]      long   MLboot   ekayal PD       0:00    12       2G      1 (JobArrayTaskLimit)
    200844    bigmem rnaspade    corre  R 1-00:17:00    20     400G      1 n99
   59250_2      long       pb   ekayal  R 1-11:05:47    12      12G      1 n97
   59250_3      long       pb   ekayal  R 1-11:05:47    12      12G      1 n98
   59250_1      long       pb   ekayal  R 1-11:06:42    12      12G      1 n96
 56616_382      long   MLboot   ekayal  R 1-11:06:42    12       2G      1 n89
 56616_383      long   MLboot   ekayal  R 1-11:06:42    12       2G      1 n90
 56616_384      long   MLboot   ekayal  R 1-11:06:42    12       2G      1 n90
```

Monitoring / Checking / Control

- Information during the run

```
$ scontrol show -dd jobid=55954
JobId=55954 JobName=MLtree
   UserId=foobar(8571) GroupId=sib(1000) MCS_label=N/A
   Priority=10207535 Nice=0 Account=betatest QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   DerivedExitCode=0:0
   RunTime=6-06:58:06 TimeLimit=20-00:00:00 TimeMin=N/A
   SubmitTime=2020-09-03T11:10:20 EligibleTime=2020-09-03T11:10:20
   StartTime=2020-09-03T11:10:28 EndTime=2020-09-23T11:10:28 Deadline=N/A
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   LastSchedEval=2020-09-03T11:10:28
   Partition=long AllocNode:Sid=slurm0:15330
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=n97
   BatchHost=n97
   NumNodes=1 NumCPUs=18 NumTasks=1 CPUs/Task=18 ReqB:S:C:T=0:0:*:*
   TRES=cpu=18,mem=36G,node=1,billing=18
   Socks/Node=* NtasksPerN:B:S:C=1:0:*:* CoreSpec=*
     Nodes=n97 CPU_IDs=0-17 Mem=36864 GRES_IDX=
   MinCPUsNode=18 MinMemoryCPU=2G MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   Gres=(null) Reservation=(null)
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/scratch2/myproject/raxml_besttree/raxml_besttree.slurm
   WorkDir=/scratch2/myproject/raxml_besttree
   StdErr=/scratch2/myproject/raxml_besttree/MLtree.out
   StdIn=/dev/null
   StdOut=/scratch2/myproject/raxml_besttree/MLtree.out
```

Monitoring / Checking / Control

- Control the pending jobs

```
$ scontrol --help                    # :P
$ scontrol update jobid=203738 ...
```

- Cancel jobs

```
$ scancel 203738

$ scancel -u lecorguille

$ scancel -u lecorguille --state PENDING

$ scancel -u lecorguille --state RUNNING
```

Monitoring / Checking / Controls

Running and Post mortem

- To check failure: memory exceed, walltime ...
- To get resources consumed

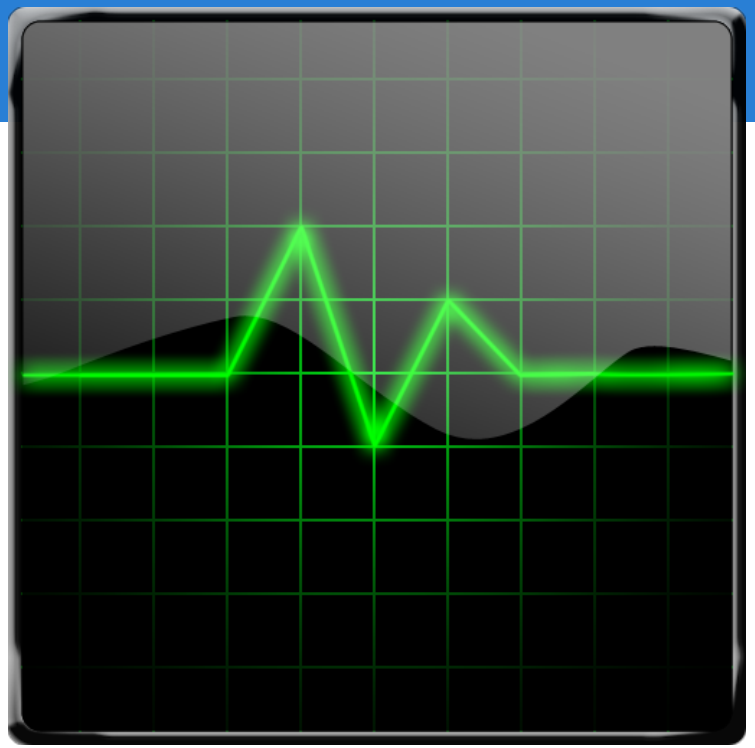
  to better fit your next --mem, --time

```
$ sacct --format=JobID,JobName,User,Submit,ReqCPUS,ReqMem,Start,NodeList,State,CPUTime,MaxVMSize%15 -j 55931
       JobID    JobName   User             Submit ReqCPUS ReqMem               Start NodeList  State     CPUTime   MaxVMSize
------------ ---------- ------ ------------------- ------- ------ ------------------- -------- ------ ---------- -----------
55931        rnaspades+  corre 2020-09-02T22:06:31      20  200Gn 2020-09-03T23:32:38      n97 FAILED 26-12:25:00
55931.batch       batch        2020-09-03T23:32:38      20  200Gn 2020-09-03T23:32:38      n97 FAILED 26-12:25:00  207991596K
```

TIPS:

```
# .bashrc
alias sacctReq='sacct --format=JobID,JobName,User%15,Partition,ReqCPUS,ReqMem,State,CPUTime,MaxVMSize%15'
```

```
$ sacctReqMem -j 55931
       JobID    JobName   User             Submit ReqCPUS ReqMem               Start NodeList  State     CPUTime   MaxVMSize
------------ ---------- ------ ------------------- ------- ------ ------------------- -------- ------ ---------- -----------
55931        rnaspades+  corre 2020-09-02T22:06:31      20  200Gn 2020-09-03T23:32:38      n97 FAILED 26-12:25:00
55931.batch       batch        2020-09-03T23:32:38      20  200Gn 2020-09-03T23:32:38      n97 FAILED 26-12:25:00  207991596K
```



srun - sbatch - sbatch - srun

SLURM

srun in sbatch

- Prefix each "steps" with `srun`

script.sbatch

```
#!/bin/bash
#SBATCH --mem=100G
#SBATCH --cpus-per-task=16

module load trinity/2.11.0
srun Trinity --single rep1.fq.gz --max_memory $SLURM_MEM_PER_NODE --CPU $SLURM_CPUS_PER_TASK
srun align_and_estimate_abundance.pl --transcripts Trinity.fasta --thread $SLURM_CPUS_PER_TASK
```

Why?

```
$ sacctReq -j 205709
       JobID    JobName             User  Partition  ReqCPUS     ReqMem      State    CPUTime       MaxVMSize        NodeList
------------ ---------- --------------- ---------- -------- ---------- ---------- ---------- --------------- ---------------
205709       trinity.s+         stage10       fast        1      100Gn  COMPLETED 1-00:19:46                       n115
205709.batch      batch                                    1      100Gn  COMPLETED 1-00:19:46       81578052K        n115
205709.0        Trinity                                    1      100Gn  COMPLETED 1-00:10:26       68125415K        n115
205709.1     align_and+                                    1      100Gn  COMPLETED   00:09:20       13452637K        n115
```

sbatch --wrap

`sbatch --wrap`

`sbatch` will wrap the specified command string in a simple "sh" shell script, and submit that script to the slurm controller.

```
[slurm0 ~] srun gunzip archive.gz
```

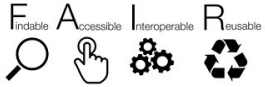
interruption → X

```
[slurm0 ~] sbatch --wrap "gunzip archive.gz"
Submitted batch job 57507
[slurm0 ~]exit
logout
```

interruption → V

TIPS:

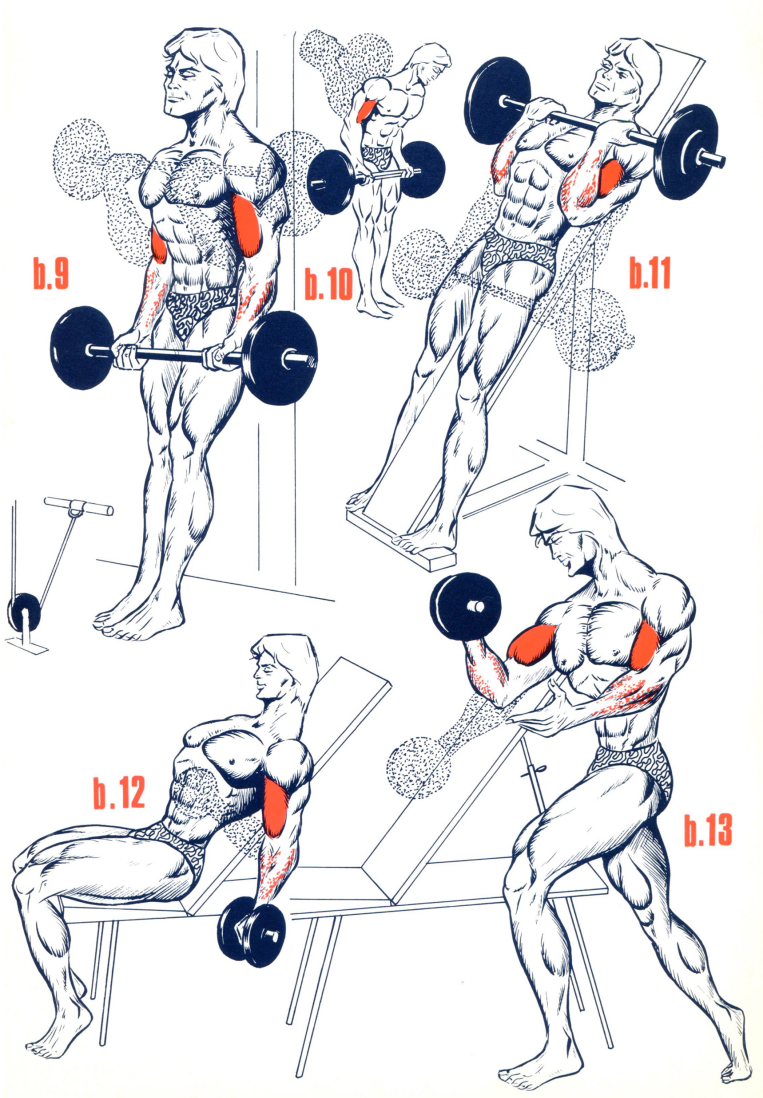
Do not abuse because it's not really



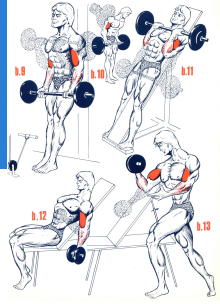
It's important to keep some track of you analyze

A sbatch script is an important metadata

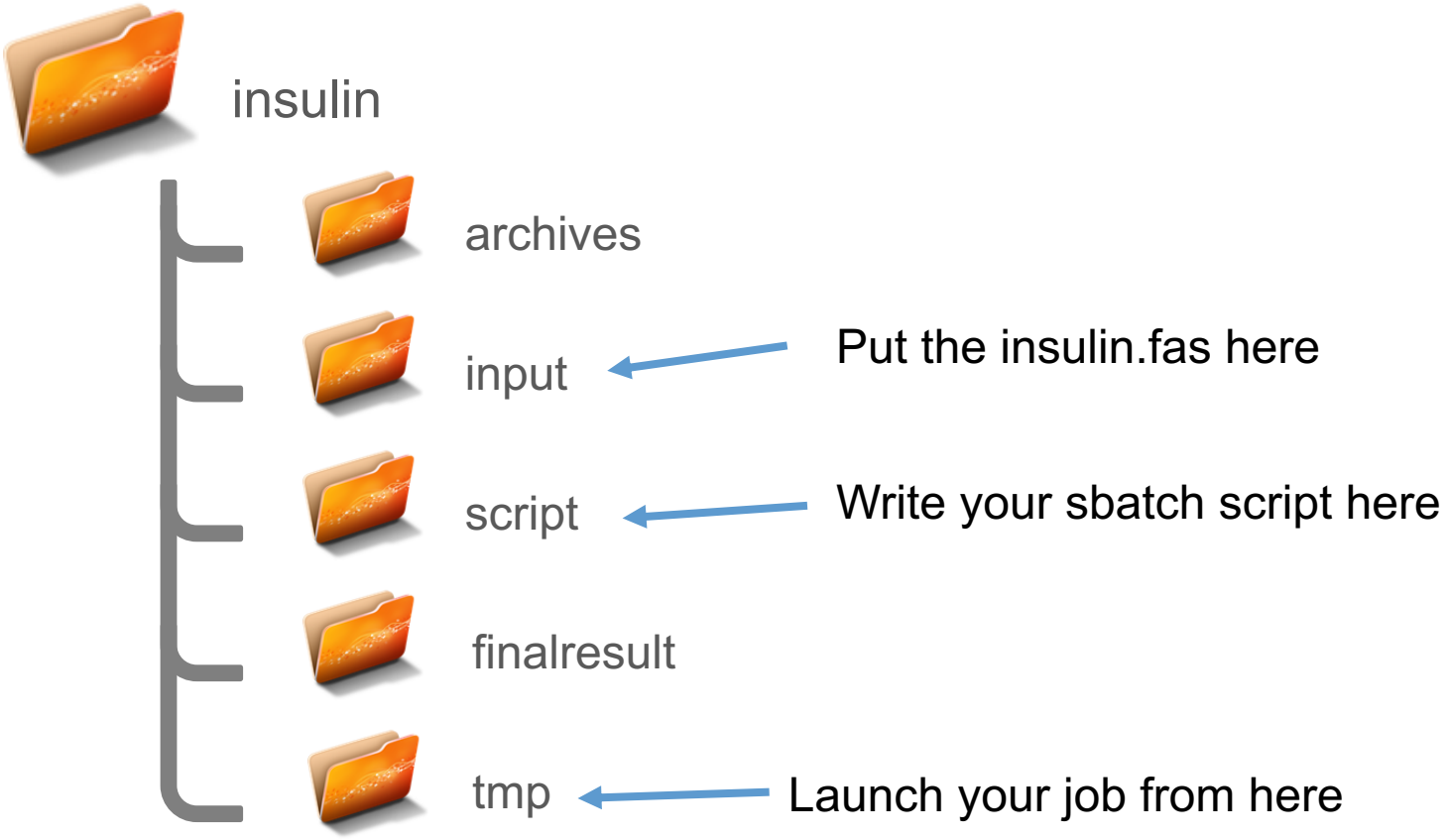
EXERCICE / EXAMPLES



Exercises/Examples

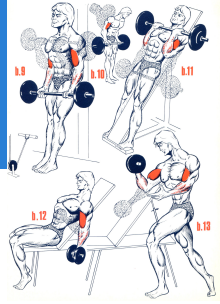


Search for sequence similarities using *blastn* on the fasta file *insulin.fas* again the database *nt*



Tips: keep in mind that your project directory is structured (input, script…)

Exercises/Examples



Search for sequence similarities using *blastn* on the fasta file *insulin.fas* again the database *nt*

Parameters:

-query insulin.fas

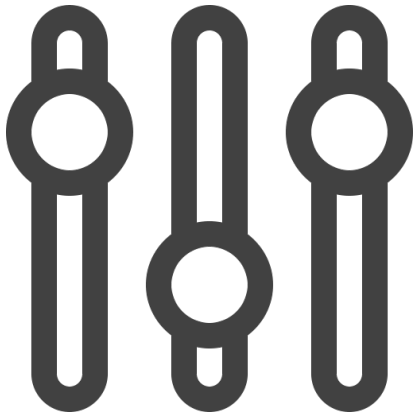
-outfmt 6

-evalue 1e-6

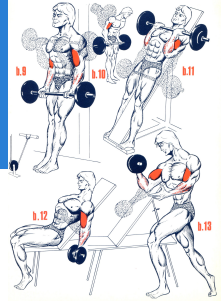
-max_target_seqs 5

-db /shared/bank/blast/all/nt

-out insulin_nt.blastn.tab



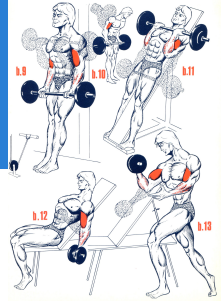
Exercises/Examples



Using:

1. srun
2. sbatch: Simple script
3. sbatch: Multithread script
4. sbatch: Job-array

1. srun

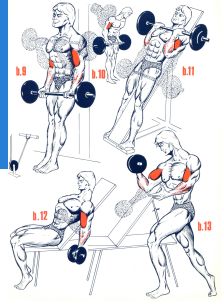


1. Launch a job on a node using `srun`

```
$ cp /tmp/insulin.fas .
```

```
$ blastn -version
...
$ module load blast/2.9.0
$ module list
...
$ blastn -version
...
$ cd /shared/projects/stage/stageXX/tp-cluster/...
$ blastn -help
...
$ srun blastn -query ...
```

2. sbatch: simple script



1. Edit a text file using `gedit` or `vim`
   - GUI: open a dedicated terminal for `gedit`

```
$ ssh -Y stageXX@slurm0.sb-roscoff.fr
[slurm0]$ cd /shared/projects/stage/stageXX/tp-cluster/
[slurm0]$ gedit script/blastn.sbatch
```

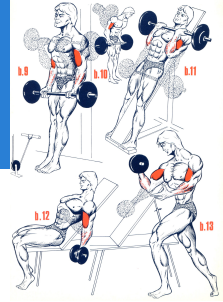
   - CLI: you can also use a pur CLI editor

```
$ vim script/blastn.sbatch
```

2. Add settings for SLURM

```
#!/bin/bash
#SBATCH --mail-user foo.bar@sb-roscoff.fr
#SBATCH --mail-type BEGIN,END
```

2. sbatch: simple script



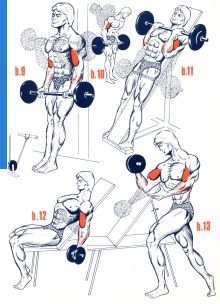
3. Append with the command line

```
#!/bin/bash
#SBATCH --mail-user foo.bar@sb-roscoff.fr
#SBATCH --mail-type BEGIN,END
#SBATCH --partition fast


INPUT="../input/insulin.fas"
OUTPUT="insulin.blast"
DATABASE="/db/blast/all/nt"

module load blast/2.9.0
blastn -query $INPUT -db $DATABASE -out $OUTPUT -outfmt 6 -evalue 1e-6 -max_target_seqs 5
```

4. Launch a `sbatch` request in the terminal

```
$ sbatch ../script/blastn.sbatch
  Submitted batch job 203738
```

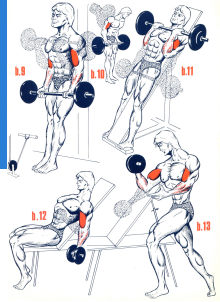
2. sbatch: simple script



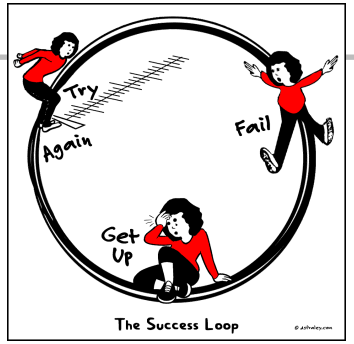
5. Monitor your job (quickly)

```
$ squeue -u stageXX
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            205559      fast blastn.s  stage10  R       1:16      1 n115
$ sacct
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
205576       blastn.sb+       fast      stage          1    RUNNING      0:0

$ sacct
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
205576       blastn.sb+       fast      stage          1  CANCELLED     0:15

$ cat slurm-205576.out
slurmstepd-n115: error: Job 205576 exceeded memory limit (4071148 > 2048000), being
killed
slurmstepd-n115: error: Exceeded job memory limit
slurmstepd-n115: error: *** JOB 205576 ON n115 CANCELLED AT 2020-11-11T14:48:29 ***
```

2. sbatch: simple script



3. Append with the command line

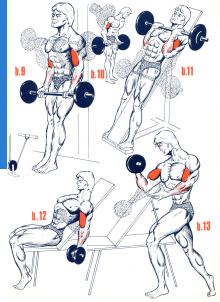



```
#!/bin/bash
#SBATCH --mail-user foo.bar@sb-roscoff.fr
#SBATCH --mail-type BEGIN,END
#SBATCH --partition fast
#SBATCH --mem 100G

INPUT="../input/insulin.fas"
OUTPUT="insulin.blast"
DATABASE="/db/blast/all/nt"

module load blast/2.9.0
blastn -query $INPUT -db $DATABASE -out $OUTPUT -outfmt 6 -evalue 1e-6 -max_target_seqs 5
```

4. Launch a `sbatch` request in the terminal

```
$ sbatch ../script/blastn.sbatch
  Submitted batch job 203739
```

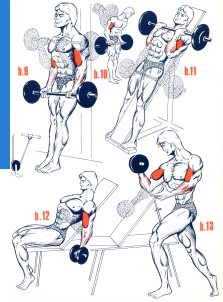
2. sbatch: simple script



5. Monitor your job (quickly)

```
$ sacct
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
205709       2-blastn.+       fast      stage          1  COMPLETED      0:0
205709.batch      batch                 stage          1  COMPLETED      0:0

$ alias sacctReq
alias sacctReq='sacct --
format=JobID,JobName,User%15,Partition,ReqCPUS,ReqMem,State,CPUTime,MaxVMSize%15,No
deList'

$ sacctReq -j 205709
       JobID    JobName           User  Partition  ReqCPUS     ReqMem      State    CPUTime       MaxVMSize        NodeList
------------ ---------- -------------- ---------- -------- ---------- ---------- ---------- --------------- ---------------
205709       2-blastn.+        stage10       fast        1      100Gn  COMPLETED   00:19:46                            n115
205709.batch      batch                                  1      100Gn  COMPLETED   00:19:46      81578052K            n115
```

3. sbatch: Multithread script



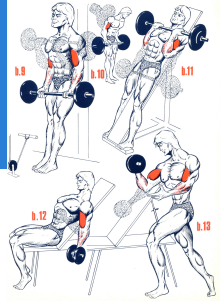
2. Add settings for SLURM
3. for `blastn`

```
#!/bin/bash
#SBATCH --mail-user foo.bar@sb-roscoff.fr
#SBATCH --mail-type BEGIN,END
#SBATCH --partition fast
#SBATCH --mem 100G
#SBATCH --cpus-per-task 4

INPUT="../input/insulin.fas"
OUTPUT="insulin_multi.blast"
DATABASE="/db/blast/all/nt"

module load blast/2.9.0
blastn -query $INPUT -db $DATABASE -out $OUTPUT -outfmt 6 -evalue 1e-6 -max_target_seqs 5
-num_threads $SLURM_CPUS_PER_TASK
```

4. Launch

```
$ sbatch ../script/blastn.sbatch
  Submitted batch job 203739
```

2. sbatch: simple script



5. Monitor your job (quickly)

```
$ squeue -u stage10
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            205602      fast 2-blastn  stage10  R       6:41      1 n115
            205614      fast 3-blastn  stage10  R       1:56      1 n82


$ alias squeue
alias squeue='squeue --format "%.18i %.9P %.8j %.8u %.2t %.10M %.5C %.8m %.6D %R" '
$ squeue -u stage10
 JOBID PARTITION     NAME     USER ST   TIME  CPUS MIN_MEMO  NODES NODELIST(REASON)
205602      fast 2-blastn  stage10  R   8:10     1     100G      1 n115
205614      fast 3-blastn  stage10  R   3:25     4     100G      1 n82
```

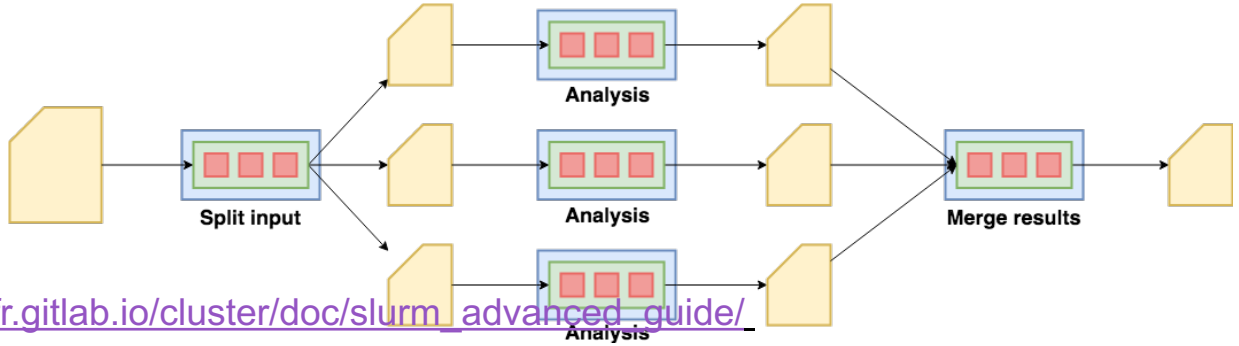
Job-array

Job arrays:

- offer a mechanism for submitting and managing collections of similar jobs quickly and easily;
- job arrays with millions of tasks can be submitted in milliseconds (subject to configured size limits).
- All jobs must have the same initial options (e.g. size, time limit, etc)

Purposes :

- When it possible, split the inputs in chunks



https://ifb-elixirfr.gitlab.io/cluster/doc/slurm_advanced_guide/

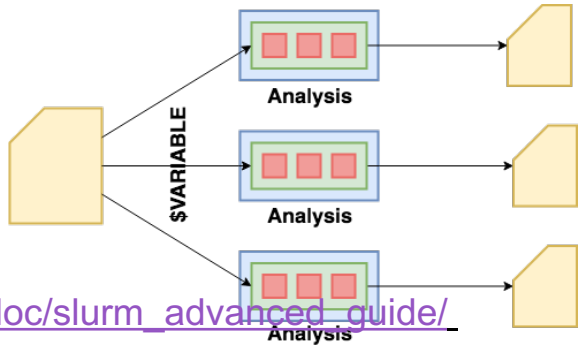
Job-array

Job arrays:

- offer a mechanism for submitting and managing collections of similar jobs quickly and easily;
- job arrays with millions of tasks can be submitted in milliseconds (subject to configured size limits).
- All jobs must have the same initial options (e.g. size, time limit, etc)

Purposes :

- Explore a parameter



https://ifb-elixirfr.gitlab.io/cluster/doc/slurm_advanced_guide/

4. sbatch: Job-array

Problem: a large number of jobs to run and they are largely identical in terms of the command to run.

For example, you may have 1000 data sets, and you want to run a single program on each of them.

Naive solution: generate 1000 shell scripts, and submit them to the cluster.

Best solution: on SLURM systems - array jobs. The advantages are:

→ You only have to write one shell script

One Script to rule them all, One Script to find them,
One Script to bring them all and in the darkness bind them

4. sbatch: Job-array

```
$ ls *.fas

488_albifrons_bret.fas      493_albifrons_bret.fas      577_praehirsuta_bret.fas
584_praehirsuta_bret.fas    594_hybride_norm.fas        703_praehirsuta_norm.fas
714_albifrons_norm.fas      724_praehirsuta_norm.fas    490_albifrons_bret.fas
570_praehirsuta_bret.fas    580_praehirsuta_bret.fas    587_hybride_norm.fas
595_hybride_norm.fas        707_praehirsuta_norm.fas    719_albifrons_norm.fas
```

1. Create the structure

```
#!/bin/bash
#SBATCH --cpus-per-task 4



INPUT="insulin.fas"



module load blast/2.9.0
blastn -query $INPUT -db /db/blast/all/nt -out ${INPUT}-nt.blast -outfmt 6 -num_threads
$SLURM_CPUS_PER_TASK
```

4. sbatch: Job-array

```
$ ls *.fas

488_albifrons_bret.fas      493_albifrons_bret.fas      577_praehirsuta_bret.fas
584_praehirsuta_bret.fas    594_hybride_norm.fas        703_praehirsuta_norm.fas
714_albifrons_norm.fas      724_praehirsuta_norm.fas    490_albifrons_bret.fas
570_praehirsuta_bret.fas    580_praehirsuta_bret.fas    587_hybride_norm.fas
595_hybride_norm.fas        707_praehirsuta_norm.fas    719_albifrons_norm.fas
```

2. Get the nth INPUT / SLURM_ARRAY_TASK_IDth

```
#!/bin/bash
#SBATCH --cpus-per-task 4



INPUT=$(ls *.fas | awk "NR==$SLURM_ARRAY_TASK_ID")



module load blast/2.9.0
blastn -query $INPUT -db /db/blast/all/nt -out ${INPUT}-nt.blast -outfmt 6 -num_threads
$SLURM_CPUS_PER_TASK
```

- NR = Number of Row
- $( ) means launch the command and get back the result

4. sbatch: Job-array

```
$ ls *.fas

488_albifrons_bret.fas      493_albifrons_bret.fas      577_praehirsuta_bret.fas
584_praehirsuta_bret.fas    594_hybride_norm.fas        703_praehirsuta_norm.fas
714_albifrons_norm.fas      724_praehirsuta_norm.fas    490_albifrons_bret.fas
570_praehirsuta_bret.fas    580_praehirsuta_bret.fas    587_hybride_norm.fas
595_hybride_norm.fas        707_praehirsuta_norm.fas    719_albifrons_norm.fas
```

2. Get the nth INPUT / SLURM_ARRAY_TASK_IDth

```
$ SLURM_ARRAY_TASK_ID=1 ; INPUT=$(ls *.fas | awk "NR==$SLURM_ARRAY_TASK_ID"); echo $INPUT
488_albifrons_bret.fas
$ SLURM_ARRAY_TASK_ID=2 ; INPUT=$(ls *.fas | awk "NR==$SLURM_ARRAY_TASK_ID"); echo $INPUT
493_albifrons_bret.fas
$ SLURM_ARRAY_TASK_ID=3 ; INPUT=$(ls *.fas | awk "NR==$SLURM_ARRAY_TASK_ID"); echo $INPUT
570_praehirsuta_bret.fas
$ ...
```

```
module load blast/2.9.0
blastn -query $INPUT -db /db/blast/all/nt -out ${INPUT}-nt.blast -outfmt 6 -num_threads
$SLURM_CPUS_PER_TASK
```

- NR = Number of Row
- $( ) means launch the command and get back the result

4. sbatch: Job-array

```
$ ls *.fas

488_albifrons_bret.fas      493_albifrons_bret.fas      577_praehirsuta_bret.fas
584_praehirsuta_bret.fas    594_hybride_norm.fas        703_praehirsuta_norm.fas
714_albifrons_norm.fas      724_praehirsuta_norm.fas    490_albifrons_bret.fas
570_praehirsuta_bret.fas    580_praehirsuta_bret.fas    587_hybride_norm.fas
595_hybride_norm.fas        707_praehirsuta_norm.fas    719_albifrons_norm.fas
```

2. Get the nth INPUT / SLURM_ARRAY_TASK_IDth

```
#!/bin/bash
#SBATCH --cpus-per-task 4



INPUT=$(ls *.fas | awk "NR==$SLURM_ARRAY_TASK_ID")



module load blast/2.9.0
blastn -query $INPUT -db /db/blast/all/nt -out ${INPUT}-nt.blast -outfmt 6 -num_threads
$SLURM_CPUS_PER_TASK
```

- NR = Number of Row
- $( ) means launch the command and get back the result

4. sbatch: Job-array

```
$ ls *.fas

488_albifrons_bret.fas      493_albifrons_bret.fas      577_praehirsuta_bret.fas
584_praehirsuta_bret.fas    594_hybride_norm.fas        703_praehirsuta_norm.fas
714_albifrons_norm.fas      724_praehirsuta_norm.fas    490_albifrons_bret.fas
570_praehirsuta_bret.fas    580_praehirsuta_bret.fas    587_hybride_norm.fas
595_hybride_norm.fas        707_praehirsuta_norm.fas    719_albifrons_norm.fas
```

3. Set the `SLURM_ARRAY_TASK_ID` range

```
#!/bin/bash
#SBATCH --cpus-per-task 4
#SBATCH --array 1-500


INPUT=$(ls *.fas | awk "NR==$SLURM_ARRAY_TASK_ID")


module load blast/2.9.0
blastn -query $INPUT -db /db/blast/all/nt -out ${INPUT}-nt.blast -outfmt 6 -num_threads
$SLURM_CPUS_PER_TASK
```

- `--array 1-1000`
- `--array 250-500`
- `--array 2,4,8,16`

4. sbatch: Job-array

```
$ squeue -u stage10
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
  56616_[4-500]      fast   blastn  stage10 PD       0:00      1
        56616_2      fast   blastn  stage10  R    01:14:23      1 n97
        56616_3      fast   blastn  stage10  R    01:14:23      1 n98
        56616_1      fast   blastn  stage10  R    01:15:18      1 n96
```

3. Set the `SLURM_ARRAY_TASK_ID` range

```
#!/bin/bash
#SBATCH --cpus-per-task 4
#SBATCH --array 1-500


INPUT=$(ls *.fas | awk "NR==$SLURM_ARRAY_TASK_ID")


module load blast/2.9.0
blastn -query $INPUT -db /db/blast/all/nt -out ${INPUT}-nt.blast -outfmt 6 -num_threads
$SLURM_CPUS_PER_TASK
```

- `--array 1-1000`
- `--array 250-500`
- `--array 2,4,8,16`

4. sbatch: Job-array

```
$ ls *.fas

488_albifrons_bret.fas      493_albifrons_bret.fas      577_praehirsuta_bret.fas
584_praehirsuta_bret.fas    594_hybride_norm.fas        703_praehirsuta_norm.fas
714_albifrons_norm.fas      724_praehirsuta_norm.fas    490_albifrons_bret.fas
570_praehirsuta_bret.fas    580_praehirsuta_bret.fas    587_hybride_norm.fas
595_hybride_norm.fas        707_praehirsuta_norm.fas    719_albifrons_norm.fas
```

3. Set the SLURM_ARRAY_TASK_ID range

```
#!/bin/bash
#SBATCH --cpus-per-task 4
#SBATCH --array 1-500


INPUT=$(ls *.fas | awk "NR==$SLURM_ARRAY_TASK_ID")


module load blast/2.9.0
blastn -query $INPUT -db /db/blast/all/nt -out ${INPUT}-nt.blast -outfmt 6 -num_threads
$SLURM_CPUS_PER_TASK
```

- `--array 1-1000`
- `--array 250-500`
- `--array 2,4,8,16`

4. sbatch: Job-array

```
$ ls *.fas

488_albifrons_bret.fas      493_albifrons_bret.fas      577_praehirsuta_bret.fas
584_praehirsuta_bret.fas    594_hybride_norm.fas        703_praehirsuta_norm.fas
714_albifrons_norm.fas      724_praehirsuta_norm.fas    490_albifrons_bret.fas
570_praehirsuta_bret.fas    580_praehirsuta_bret.fas    587_hybride_norm.fas
595_hybride_norm.fas        707_praehirsuta_norm.fas    719_albifrons_norm.fas
```

5. [optional] limit the number of running jobs

```
#!/bin/bash
#SBATCH --cpus-per-task 4
#SBATCH --array 1-500%100



INPUT=$(ls *.fas | awk "NR==$SLURM_ARRAY_TASK_ID")



module load blast/2.9.0
blastn -query $INPUT -db /db/blast/all/nt -out ${INPUT}-nt.blast -outfmt 6 -num_threads
$SLURM_CPUS_PER_TASK
```


The End