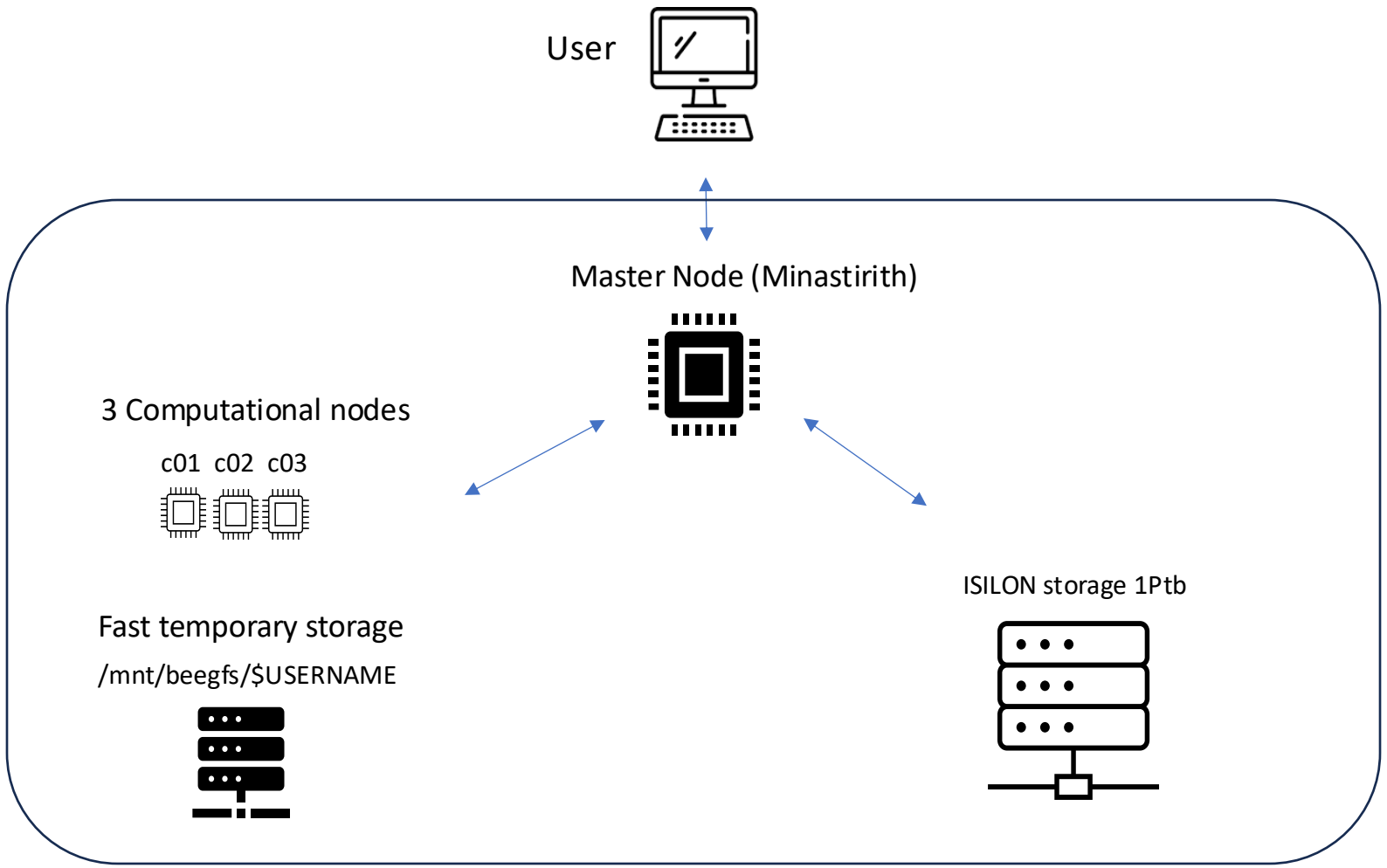
Computations with the IJC High-performance computing cluster

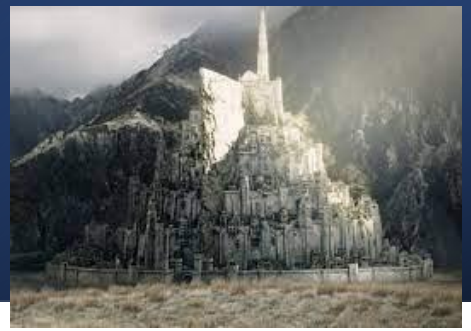
Angelika Merkel (Head of Bioinformatics Core Facility)
09/04/2025

Supported by IT & BIT

IJC HPC cluster



IJC cluster components



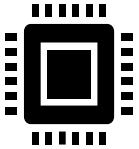
- Master Node (Minastirith)

  12 cores 128Gb Ram

  = connects user, computational nodes (+ flash storage), ISILON

  USE: to connect, submit slurm jobs

  -> Shared resource, do not run your jobs here!



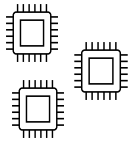
- Computational nodes

  c01 128 cores 1Tb Ram
  c02 128 cores 1Tb Ram
  c03 128 cores 1Tb Ram GPU

  = computing (job execution via SLURM)

  USE: to execute parallel or high-mem jobs

  -> Shared resource, use wisely!



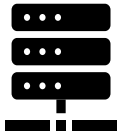
- Flash storage '/mnt/beegfs/'

  100Tb SSD memory

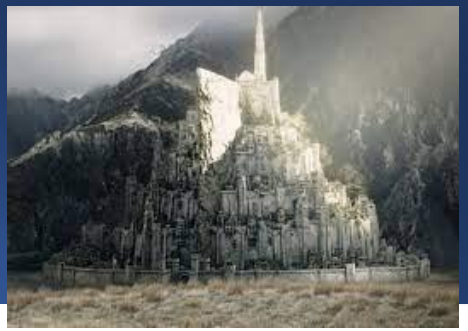
  = flash memory attached to computational nodes for fast read/write access

  USE: to temporarly store data

  -> Shared resource, use wisely!



Connecting to the cluster



You can connect to the cluster (master node = minastirith) in several ways:

1. From anywhere with a browser via VPN Portal: https://vpn.carrerasresearch.org (bookmark 'Minastirith')

2. From your machine (Linux/Mac) with a terminal:

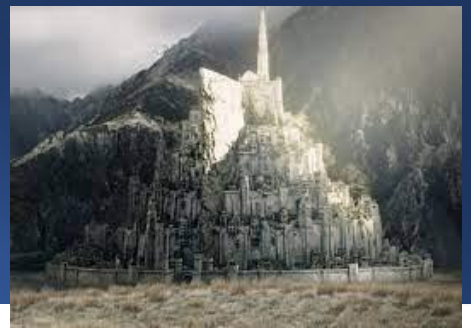
   $ ssh username@minastirith

3. From your machine using a ssh client (e.g. PUTTy for windows) connect directly to the host:

   minastirith

To connect from outside the IJC network you need to connect to the SSHlogin node first.

Enter minastirth



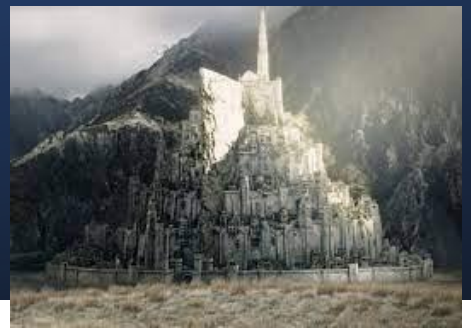
```
$ ssh amerkel@minastirith
```

```
[amerkel@IJC20249 ~]$ ssh minastirith
Last login: Mon Jun 10 17:06:04 2024 from 84.88.90.193

                    o8o                                        .    o8o           o8o      .   oooo
                                                             .o8                          .o8    888
 ooo. .oo.  .oo.   oooo  ooo. .oo.    .oooo.    .oooo.o    888   oooo  ooood8b  oooo    888    888
 `888P"Y88bP"Y88b  `888  `888P"Y88b  `P  )88b  d88(  "8  .o888oo `888  oooo d8b `888  .o888oo  888 .oo.
  888   888   888   888   888   888   .oP"888  `"Y88b.     888    888   888      888    888    888   888
  888   888   888   888   888   888  d8(  888  o.  )88b    888 .  888   888      888    888 .  888   888
 o888o o888o o888o o888o o888o o888o Y888 -8o  8 888P       888  o888o d888b    o888o    888  o888o o888o

 Master AMD EPYC 7272  12 cores 128Gb Ram *************
 Node01 AMD EPYC 7702 128 cores   1Tb Ram *************
 Node02 AMD EPYC 7702 128 cores   1Tb Ram *************
 Node03 AMD EPYC 7702 128 cores   1Tb Ram Nvidia A40 GPU



 Please remember:

 1.- Save your stuff before it gets deleted! <3

       BEEGFS data (/mnt/beegfs) will be periodically removed.
       Transfer your data to /ijc/LABS or /ijc/USERS !

       Example: $ rsync -rlv --checksum source_folder destination_folder

 2.- Do not launch jobs on the master node, use sbatch files or salloc. Interactive jobs on minastirith will be killed!

 3.- Check performance of your finished jobs!

     Example: $ seff [jobid]

 Thank you all!


Partition Name  Max Time  Default Time  Max CPU x Node  Max Mem x CPU  Def Mem x CPU  Max Mem x Node
==============  ========  ============  ==============  =============  =============  ==============
fast             2 hours            1h              64          4000M          2000M         256000M
highMem          1 Day              1h               8              -              -         800000M
hpc              3 Days             4h              80         200000M         4000M               -

Max run Cpu Per User: 48


show personal statistics
========================
module load reportseff
reportseff --since d=2 --format "+user,jobname,start,ncpus,reqmem" -u amerkel
[amerkel@minastirith ~]$
```

Working with the HPC

input → software → instructions → resources → output

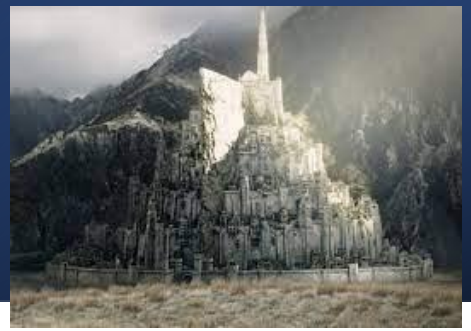
Using the software



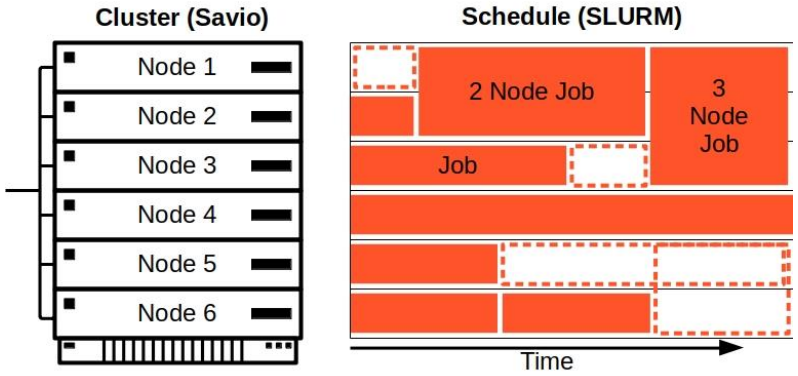
Available as modules

Description	Command
see all available modules	`module avail`
load/unload a module	`module load/unload`
see all loaded modules	`module list`
unload all modules	`module purge`

Executing a job



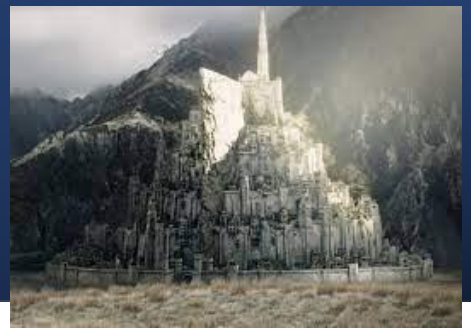
All tasks (jobs) executed on the cluster computational nodes are managed by SLURM (the scheduler)



https://docs-research-it.berkeley.edu/services/high-performance-computing/user-guide/

SLURM schedules each job based on available resources (CPU, memory, nodes, execution time, etc)

SLURM script

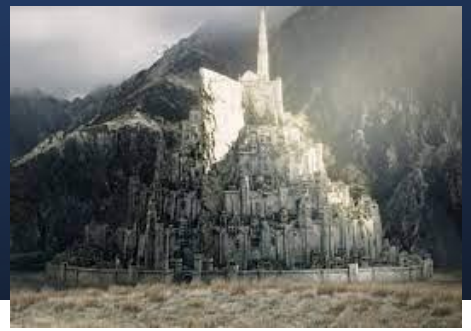


./my_clusterjob.sh # basic serial job

```
#!/bin/bash


# SLURM arguments
#SBATCH --job-name=job_serial     # Job name
#SBATCH --cpus-per-task=1         # Run on a single CPU
#SBATCH --mem=4gb                 # Job memory request
#SBATCH --time=00:10:00           # Time limit hrs:min:sec
#SBATCH --output=job_%j.log       # Standard output and error log


# load software

module load R


# message something

echo "Running R script on a single CPU core"

# run
Rscript myscript.r
```

%j = jobid

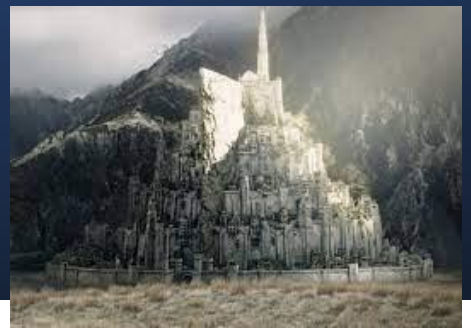
Slurm Commands



Command	Description
sbatch my_clusterjob.sh	Submit job for execution
squeue	Show the actual job queue
scancel jobid	Cancel a job
sacct –j jobid	Job accounting infos
seff jobid	Report on the efficiency of a job's cpu and memory utilization (after the job has finished)
salloc	Allocate resource for an interactive shell

```
module load reportseff
reportseff --since d=2 --format "+user,jobname,start,ncpus,reqmem" -u username
```

Deploy interactive shell



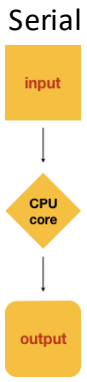
You can deploy an interactive shell (with specified resource) on a computational node for testing:

```
$ salloc –c 4
```

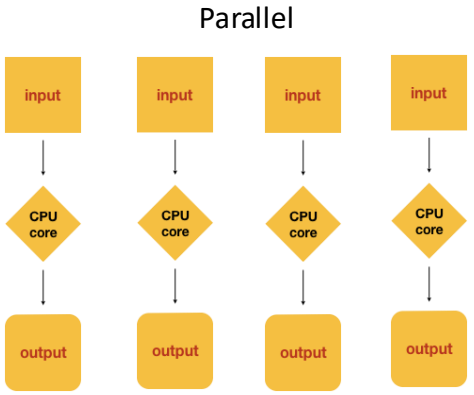
```
[amerkel@minastirith ~]$ salloc
salloc: Granted job allocation 3216713
salloc: Waiting for resource configuration
salloc: Nodes c02 are ready for job
[amerkel@c02 ~]$
```

Parallelization increases speed

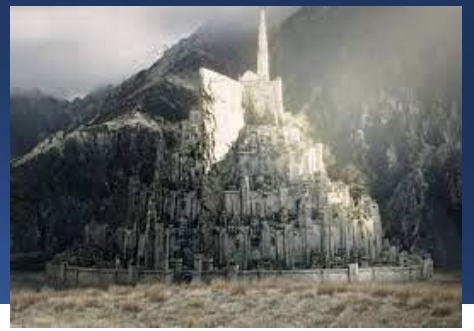
1. Single input + single task



2. Multiple input + Single task



Job arrays

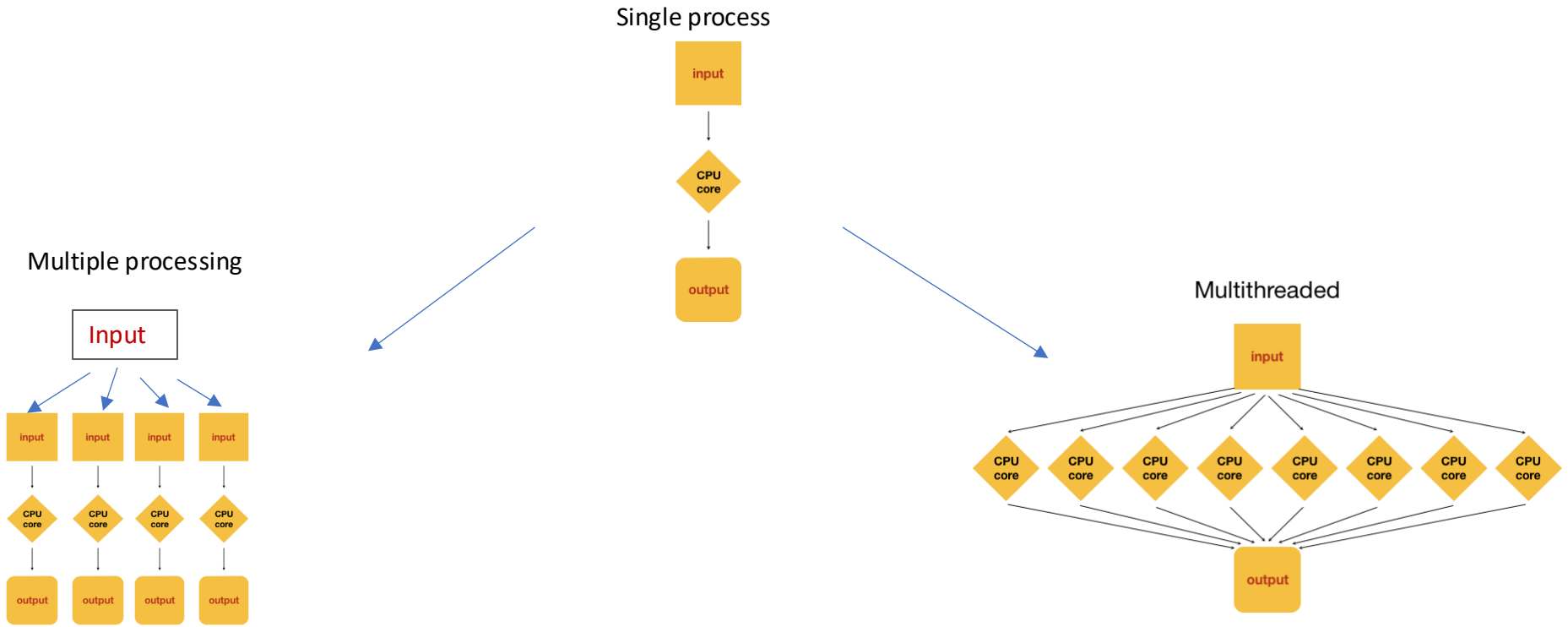


```
#!/bin/bash


# SLURM arguments
#SBATCH --job-name=job_array          # Job name
#SBATCH --ntasks=1                    # Run a single task
#SBATCH --mem=4gb                     # Job Memory
#SBATCH --time=00:05:00               # Time limit hrs:min:sec
#SBATCH --output=array_%A-%a.log      # Standard output and error log
#SBATCH --array=1-5                   # Array range


# message something
echo "This is task" $SLURM_ARRAY_TASK_ID

# run
infile=$(ls *.txt | sed -n ${SLURM_ARRAY_TASK_ID}p)      # multiple input files (*.txt)


./myscript.sh $infile                                    # run myscript for each input file
```

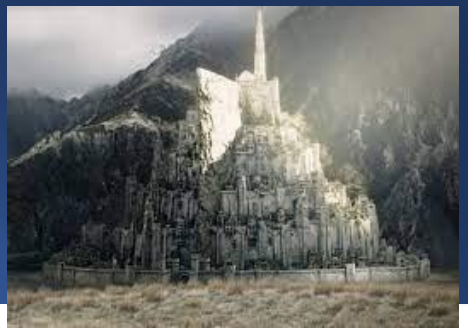
Job arrays allow to manage and submit similar jobs or execute the same job over multiple inputs (samples/data sets/parameter sets)

- Multiple jobs are created/submitted from one script, each with a different $SLURM_ARRAY_TASK_ID (=build-in variable)
- $SLURM_ARRAY_TASK_ID takes value specified with the array range

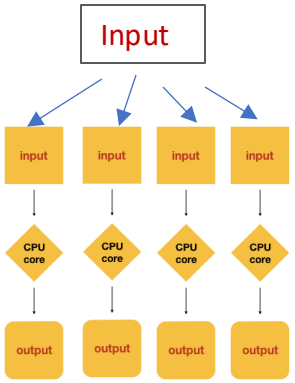
Parallelization increases speed



Multi-processing



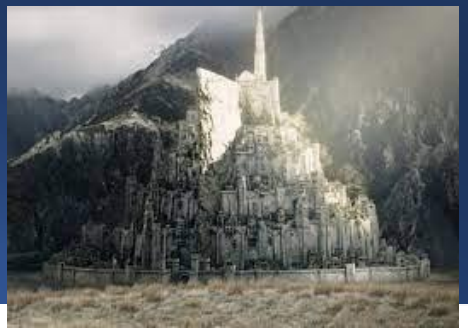
Multiple processes

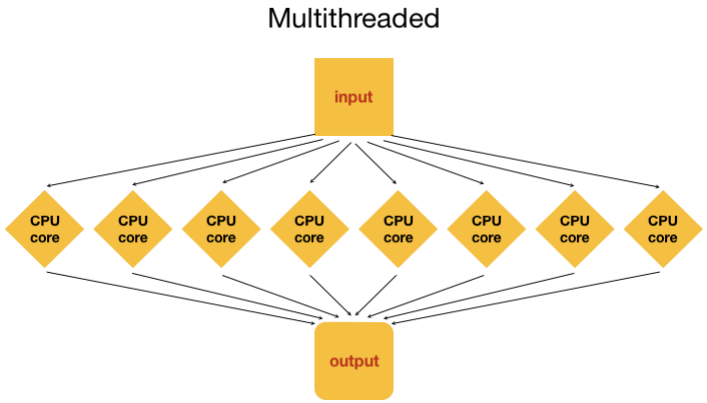


In **multi-processing** multiple processes are executed on multiple CPUs:

- #SBATCH –-ntask=number_of_processes

Multi-threading



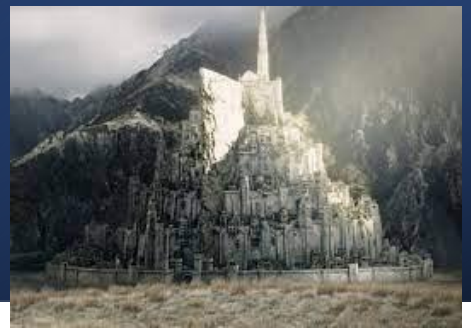


In **multi-threading** a single process is executed with multiple threads (e.g. multiple code segments):

1. Shared-memory multi-threading application (threaded, OpenMP, PTHREADS) can use multiple cpus but only on a single node:
   - #SBATCH ---nodes=1
   - #SBATCH –-ntask=1
   - #SBATCH –-cpus-per-task=number_of_threads

2. OpenMPI applications allow to share threads across nodes:
   - #SBATCH –-ntask=1
   - #SBATCH –cpus-per-task=number_of_threads

SLURM partitions (queues)



There are multiple partitions on the cluster depending to which jobs can be submitted depending on the required resources

#SBATCH --partition

Partition Name	Max Time	Default Time	Max CPU x Node	Max Mem x CPU	Def Mem x CPU	Max Mem x Node
fast	2 hours	1h	64	4000M	2000M	256000M
highMem	1 Day	1h	8	-	-	800000M
hpc	3 Days	4h	80	200000M	3000M	-

Max Cpu by job: 16

***NOTE: If the resource requirement is not specified, default value = max value**

Do's & Don’t's

DO

- only request resources that you need (more CPUs or memories don't make your job faster!)
- check if the resources you requested were efficiently used (use seff -j jobid or similar)
- check if your job should be submitted to a special queue
- submit resource intensive jobs at low peak times (over night or the weekend)
- break large jobs down into smaller ones
- optimize your code

DON’T

- ask for unnecessary resources (+ 10/15% time or memory are usually sufficient)
- Do not execute any resource intensive task on the master node, use an interactive shell instead
- forget to close your interactive session

How to get help

- contact **IT** for help with resource requirements, software and permissions
- contact **BIT** for help with coding and workflows
- check the **documentation** @:

  https://howto.carrerasresearch.org/

  https://vpn.carrerasresearch.org/ > wordpress
- check the BIT workshop pages @:

  https://ijcbit.github.io/Workshops/Trainings/HPC/Intro_to_IJC_HPC.html
- Teams: ijcbioinfo

Questions?