A Practical guide to HPC

mattia.pellegrino@unipr.it

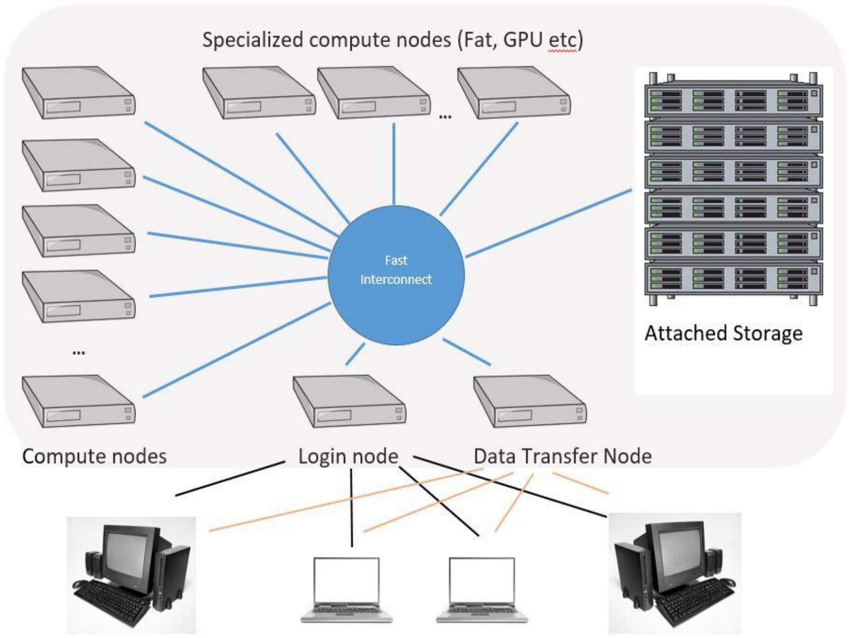
What is a cluster? (1)

- A **computer cluster** is a group of tightly coupled computers (**nodes**) that work together closely
- They can be seen as a **single computer**
- Clusters are commonly connected through **fast local area network**
- Clusters have evolved to support various applications:
    - **Ecommerce**
    - **High performance database** applications
- Clusters are usually deployed to improve **speed** and **reliability** over that provided by a single computer

What is a cluster? (2)

- In cluster computing each node (within a cluster) is an **independent system**
  - Has its own **operating system**
  - **Private** memory
  - Its own **file system** (in some cases)
- Programs or software run on clusters usually employ a procedure called "***message passing***" (memory is not shared)
- Cluster computing can also be used as a **low-cost** form of **parallel processing** for scientific applications

What is a cluster? (3)



History

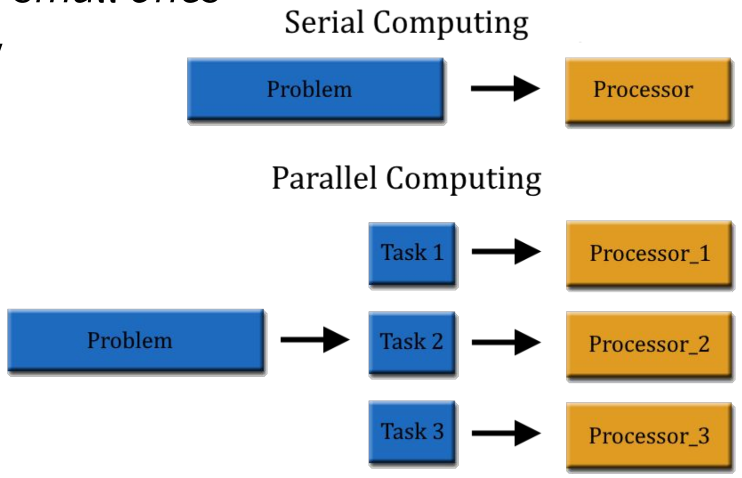
- The first clustering product was **ARCnet**, distributed and developed by Datapoint in **1977**
- Then in **1980 VAXcluster** produced a new product
- **Sun Microsystem**, **Microsoft** and other leading hardware and software companies offer clustering packages

Serial computing

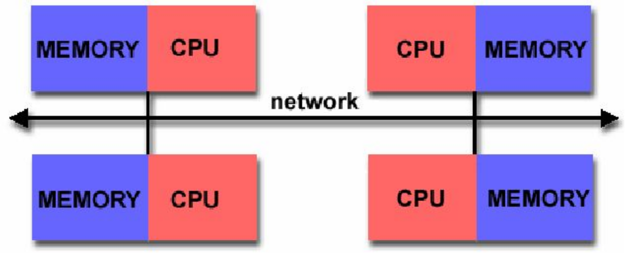
- **Serial computing**
  - Based on **Von Neumann architecture**, it have to process data and instructions respecting a sequential order
  - Limits:
    - Need **high speed** cpu
    - **Light** speed
    - **Heat**
    - **VLSI** (very large scale integration) is not infinite

Parallel computing

- **Parallel computing**
  - "*Big problems can be often divided into small ones*"
  - Problems can be solved simultaneously
  - Advantages:
    - Improve computing speed
    - Break memory limits
    - Exploit non-local resources



Distributed Memory



- Distributed memory requires a communication network for the information exchange:
  - Each **processor** has its own **local memory**.
  - Each **memory** has a **separate**, **independent** address space.
  - **Read**/**write** operations are local
  - To allow a task to **access remote data**, the programmer must explicitly **manage** the **communication** among tasks.
  - The corresponding programming model is called **message passing**

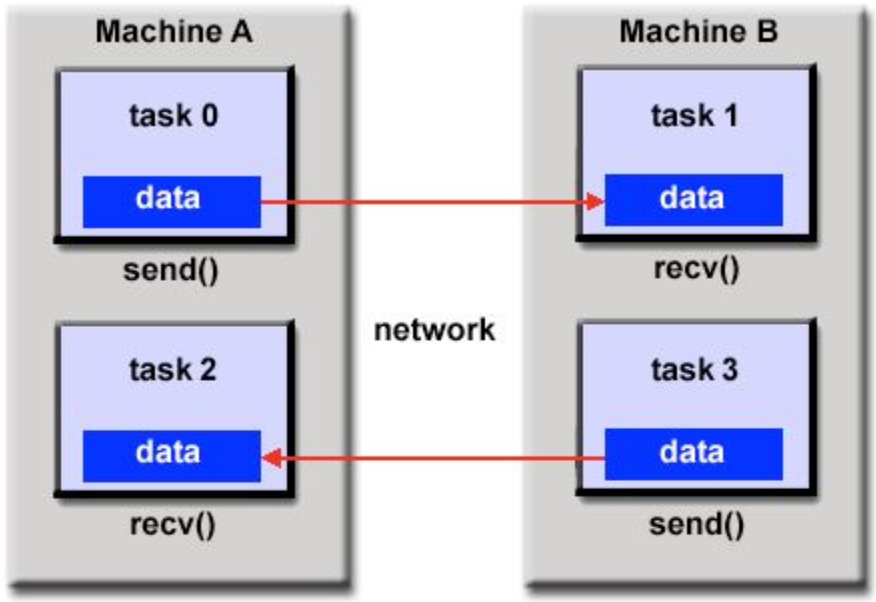
MPI

- **Message passing Interface**:
    - Communication protocol for **message exchanges**
    - There are **multiple** implementations of MPI
    - It is **not** a **language**
    - The standard has been defined through an **open process** by **community** of parallel computing vendor, computer scientists, nd application developers
    - It does not have a debugging facility and this make MPI **lightweight**
    - Several MPI implementations: **MPICH**, **Open MPI** (best overall), **Intel MPI Library**

MPI

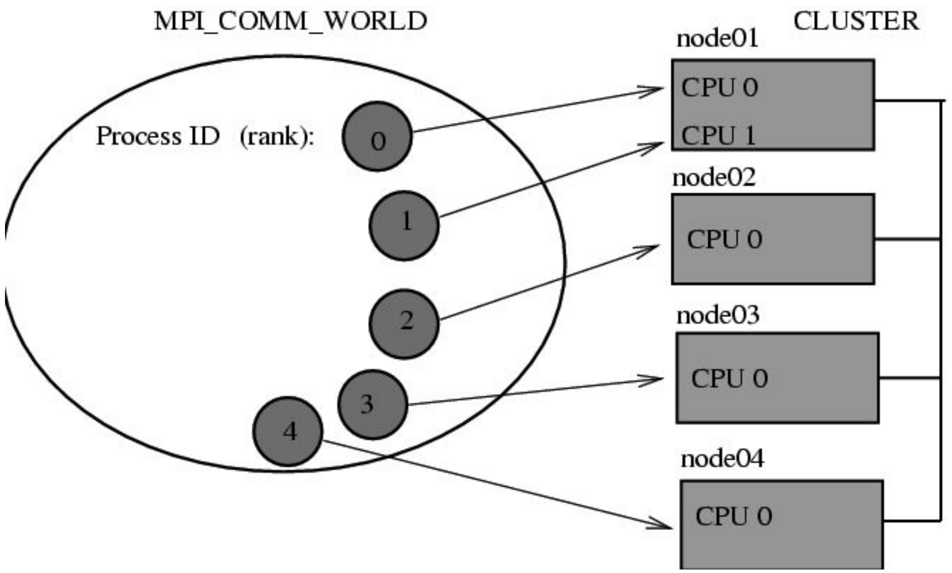
- Cooperation among processes is based on **explicit communication**
- A **sender** process and a **receiver** process exchange messages
- Each process is an **instance** of running sub-program.

- Each process is identified by an integer number, called **rank**, ranging from **0** to **n-1**, where n (size) is the total number of processes.



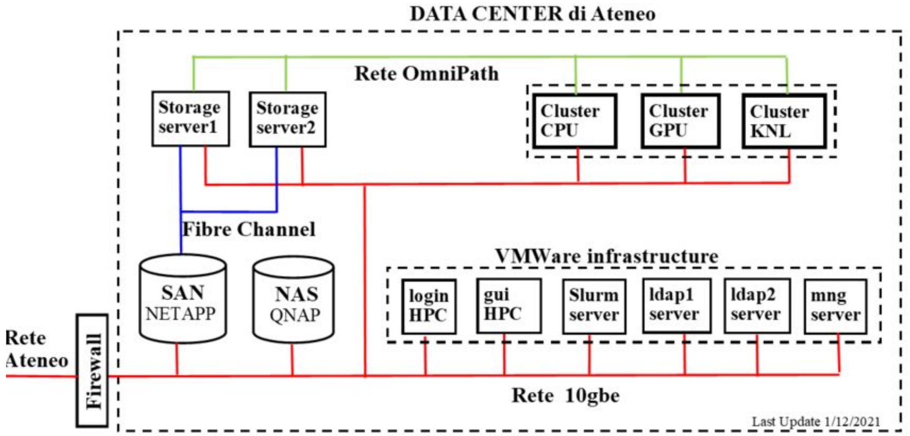
MPI - SPECIFICATIONS

- MPI allows you to create logical **groups** of processes
  - In each **group**, a process is identified by its **rank**.
- **Communicators** are **objects** that handle communication between processes.
  - **intra-communicator** handles processes within a single group
  - **inter-communicator** handles communication between two distinct groups.
- By default, there is a single group that contains all your processes, and the intra-communicator **MPI_COMM_WORLD**

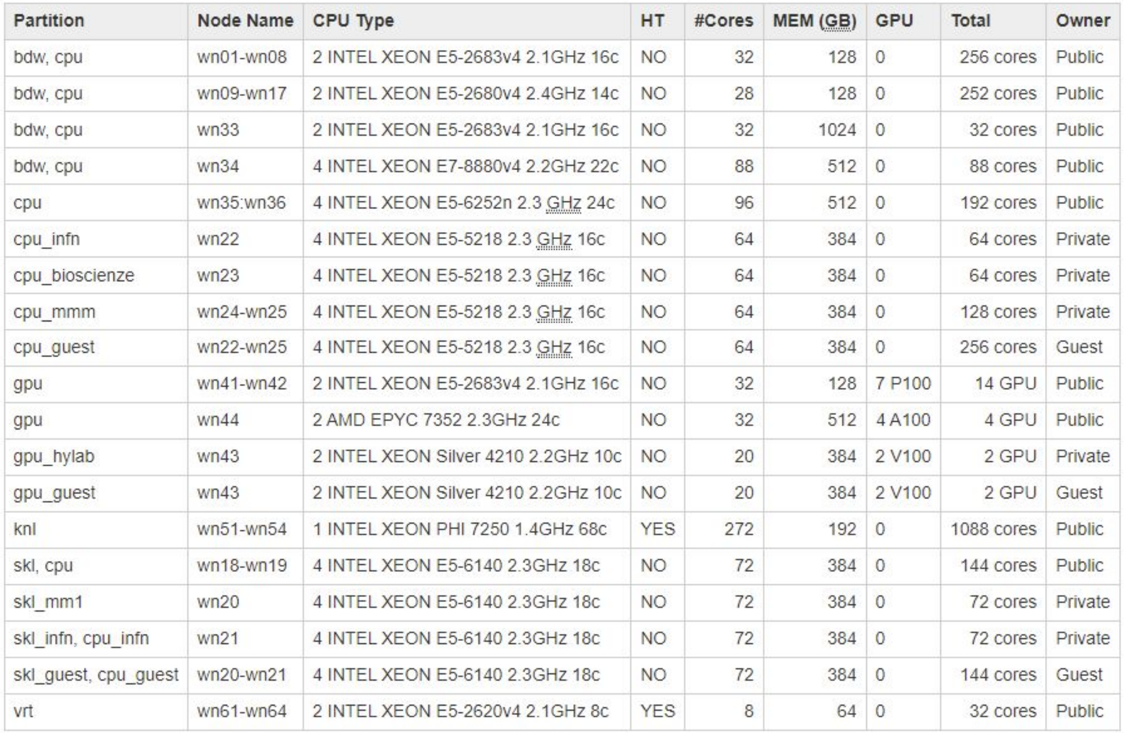


Unipr - HPC

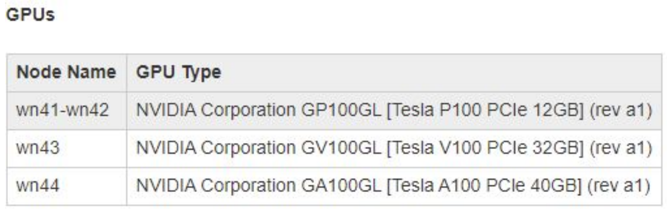
- **Website**:
  - https://www.hpc.unipr.it
- **Project Description:**
  - https://www.hpc.unipr.it/dokuwiki/doku.php?id=calcoloscientifico:progetto
- **User Guide:**
  - https://www.hpc.unipr.it/dokuwiki/doku.php?id=calcoloscientifico:userguide



Unipr - HPC - Nodes



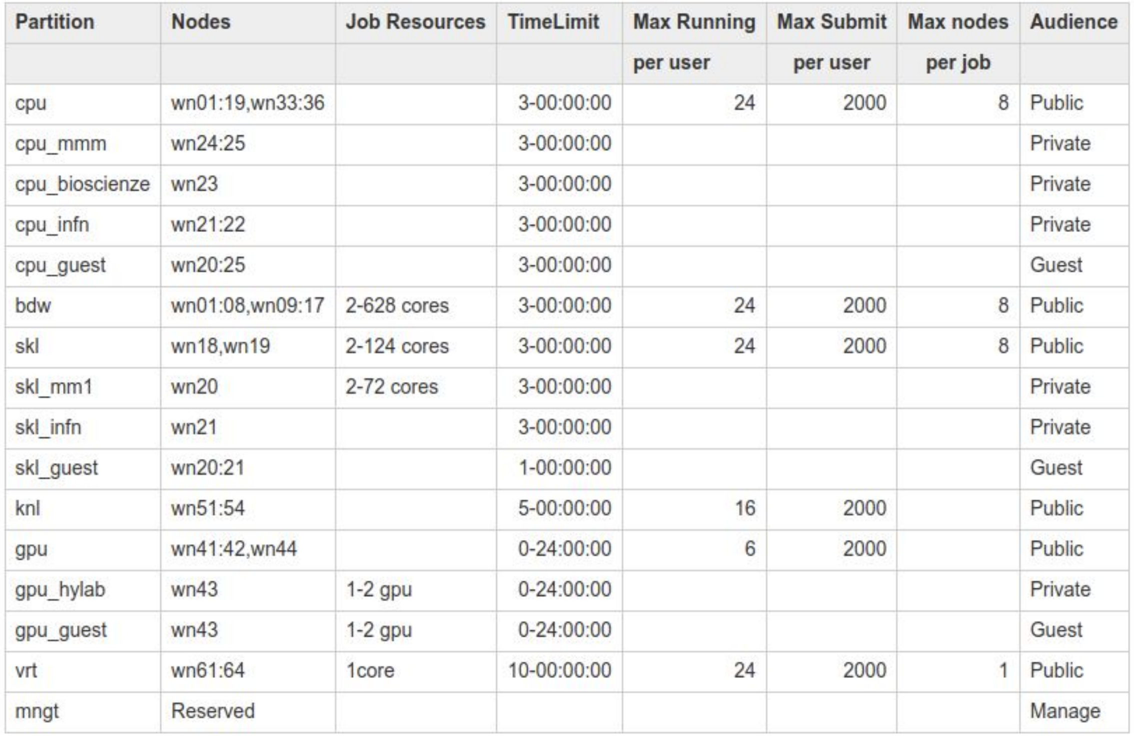
Partition	Node Name	CPU Type	HT	#Cores	MEM (GB)	GPU	Total	Owner
bdw, cpu	wn01-wn08	2 INTEL XEON E5-2683v4 2.1GHz 16c	NO	32	128	0	256 cores	Public
bdw, cpu	wn09-wn17	2 INTEL XEON E5-2680v4 2.4GHz 14c	NO	28	128	0	252 cores	Public
bdw, cpu	wn33	2 INTEL XEON E5-2683v4 2.1GHz 16c	NO	32	1024	0	32 cores	Public
bdw, cpu	wn34	4 INTEL XEON E7-8880v4 2.2GHz 22c	NO	88	512	0	88 cores	Public
cpu	wn35:wn36	4 INTEL XEON E5-6252n 2.3 GHz 24c	NO	96	512	0	192 cores	Public
cpu_infn	wn22	4 INTEL XEON E5-5218 2.3 GHz 16c	NO	64	384	0	64 cores	Private
cpu_bioscienze	wn23	4 INTEL XEON E5-5218 2.3 GHz 16c	NO	64	384	0	64 cores	Private
cpu_mmm	wn24-wn25	4 INTEL XEON E5-5218 2.3 GHz 16c	NO	64	384	0	128 cores	Private
cpu_guest	wn22-wn25	4 INTEL XEON E5-5218 2.3 GHz 16c	NO	64	384	0	256 cores	Guest
gpu	wn41-wn42	2 INTEL XEON E5-2683v4 2.1GHz 16c	NO	32	128	7 P100	14 GPU	Public
gpu	wn44	2 AMD EPYC 7352 2.3GHz 24c	NO	32	512	4 A100	4 GPU	Public
gpu_hylab	wn43	2 INTEL XEON Silver 4210 2.2GHz 10c	NO	20	384	2 V100	2 GPU	Private
gpu_guest	wn43	2 INTEL XEON Silver 4210 2.2GHz 10c	NO	20	384	2 V100	2 GPU	Guest
knl	wn51-wn54	1 INTEL XEON PHI 7250 1.4GHz 68c	YES	272	192	0	1088 cores	Public
skl, cpu	wn18-wn19	4 INTEL XEON E5-6140 2.3GHz 18c	NO	72	384	0	144 cores	Public
skl_mm1	wn20	4 INTEL XEON E5-6140 2.3GHz 18c	NO	72	384	0	72 cores	Private
skl_infn, cpu_infn	wn21	4 INTEL XEON E5-6140 2.3GHz 18c	NO	72	384	0	72 cores	Private
skl_guest, cpu_guest	wn20-wn21	4 INTEL XEON E5-6140 2.3GHz 18c	NO	72	384	0	144 cores	Guest
vrt	wn61-wn64	2 INTEL XEON E5-2620v4 2.1GHz 8c	YES	8	64	0	32 cores	Public



GPUs

Node Name	GPU Type
wn41-wn42	NVIDIA Corporation GP100GL [Tesla P100 PCIe 12GB] (rev a1)
wn43	NVIDIA Corporation GV100GL [Tesla V100 PCIe 32GB] (rev a1)
wn44	NVIDIA Corporation GA100GL [Tesla A100 PCIe 40GB] (rev a1)

Unipr - HPC - Limits



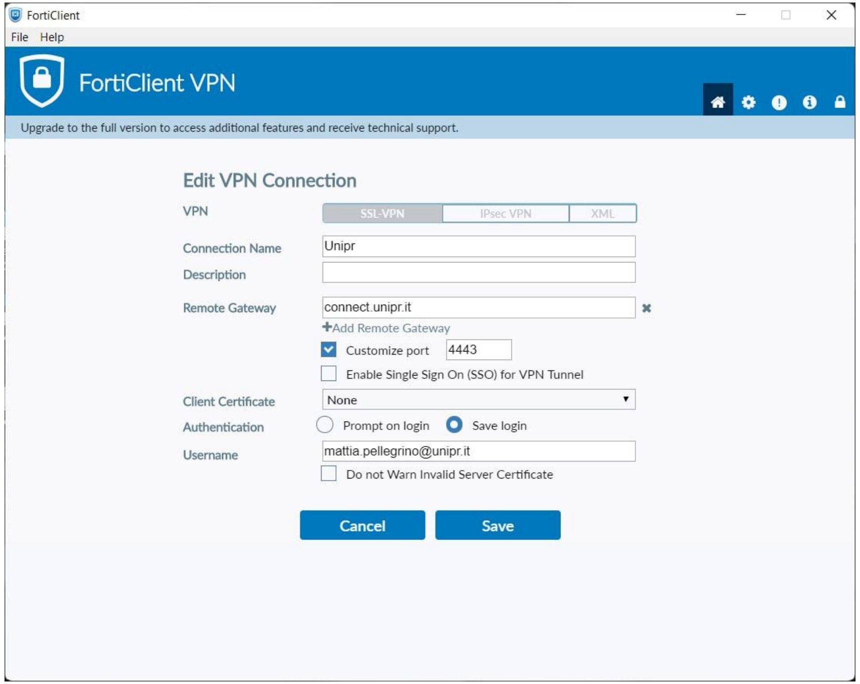
Partition	Nodes	Job Resources	TimeLimit	Max Running	Max Submit	Max nodes	Audience
				per user	per user	per job	
cpu	wn01:19,wn33:36		3-00:00:00	24	2000	8	Public
cpu_mmm	wn24:25		3-00:00:00				Private
cpu_bioscienze	wn23		3-00:00:00				Private
cpu_infn	wn21:22		3-00:00:00				Private
cpu_guest	wn20:25		3-00:00:00				Guest
bdw	wn01:08,wn09:17	2-628 cores	3-00:00:00	24	2000	8	Public
skl	wn18,wn19	2-124 cores	3-00:00:00	24	2000	8	Public
skl_mm1	wn20	2-72 cores	3-00:00:00				Private
skl_infn	wn21		3-00:00:00				Private
skl_guest	wn20:21		1-00:00:00				Guest
knl	wn51:54		5-00:00:00	16	2000		Public
gpu	wn41:42,wn44		0-24:00:00	6	2000		Public
gpu_hylab	wn43	1-2 gpu	0-24:00:00				Private
gpu_guest	wn43	1-2 gpu	0-24:00:00				Guest
vrt	wn61:64	1core	10-00:00:00	24	2000	1	Public
mngt	Reserved						Manage

SLURM

- Slurm is an open source, fault-tolerant, and highly scalable **cluster management** and **job scheduling system** for large and small Linux clusters
- SLURM manages user jobs with the following key characteristics:
    - set of requested resources:
        - X number of computing resources: **nodes** (including all their CPUs and cores) or **CPUs** (including all their cores) or cores
        - X number of accelerators (**GPUs**)
        - X **amount** of **memory**: either per node or per (logical) CPU
        - X the (wall)**time** needed for the user's tasks to complete their work
        - a set of **constraints** limiting jobs to nodes with specific features
        - a requested **node partition** (job queue)

PRACTICAL GUIDE

- First of all, you need to be connected to UNIPR wireless network, instead you must use a VPN in order to connect to the HPC facility



FortiClient

File Help

FortiClient VPN

Upgrade to the full version to access additional features and receive technical support.

Edit VPN Connection

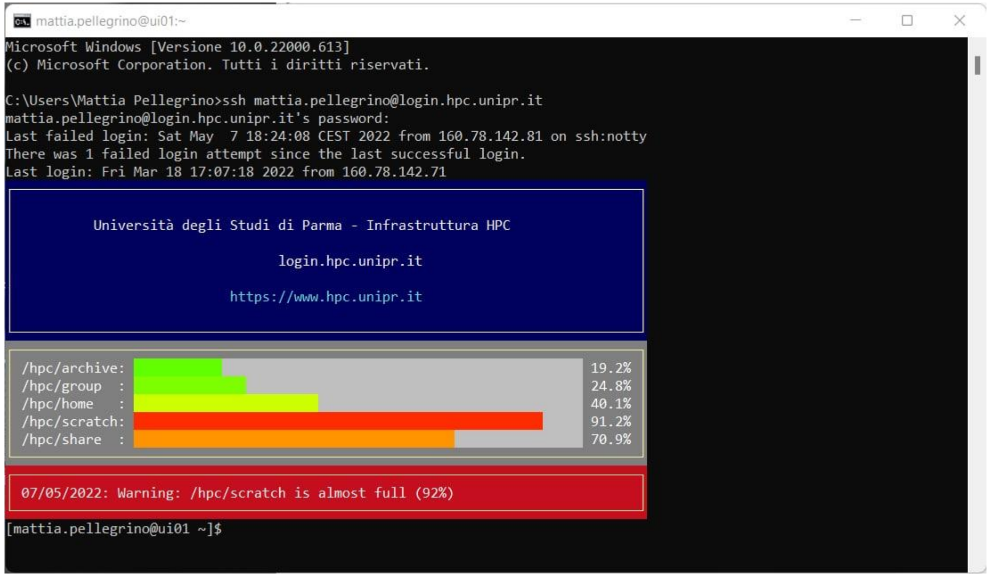
VPN	[✓ SSL-VPN] IPsec VPN XML
Connection Name	Unipr
Description	
Remote Gateway	connect.unipr.it
	+Add Remote Gateway
	✓ Customize port 4443
	☐ Enable Single Sign On (SSO) for VPN Tunnel
Client Certificate	None
Authentication	Prompt on login [✓ Save login]
Username	mattia.pellegrino@unipr.it
	☐ Do not Warn Invalid Server Certificate

Cancel Save

- **Steps**:
  - Download **FortiClient**: https://www.fortinet.com/it
  - Use this parameters to setting a new VPN connection:
    - Host: **connect.unipr.it**
    - Port: **4443**
  - Use you UNIPR credentials to login

PRACTICAL GUIDE

- In order to connect to the HPC Cluster, you must use an SSH protocol connection



```
mattia.pellegrino@ui01:~
Microsoft Windows [Versione 10.0.22000.613]
(c) Microsoft Corporation. Tutti i diritti riservati.

C:\Users\Mattia Pellegrino>ssh mattia.pellegrino@login.hpc.unipr.it
mattia.pellegrino@login.hpc.unipr.it's password:
Last failed login: Sat May  7 18:24:08 CEST 2022 from 160.78.142.81 on ssh:notty
There was 1 failed login attempt since the last successful login.
Last login: Fri Mar 18 17:07:18 2022 from 160.78.142.71

          Università degli Studi di Parma - Infrastruttura HPC

                         login.hpc.unipr.it

                      https://www.hpc.unipr.it

 /hpc/archive:                                                      19.2%
 /hpc/group  :                                                      24.8%
 /hpc/home   :                                                      40.1%
 /hpc/scratch:                                                      91.2%
 /hpc/share  :                                                      70.9%

 07/05/2022: Warning: /hpc/scratch is almost full (92%)

[mattia.pellegrino@ui01 ~]$
```

- **Steps**:
  - Open the system terminal
  - **ssh <nomeutente>@login.hpc.unipr.it**
  - Password: your institutional password

FILE TRANSFER

- **SSH** is the only protocol for external communication and can also be used for file transfer.

- If you use a Unix-like client (Linux, MacOS X) you can use the command **scp** or **sftp**.

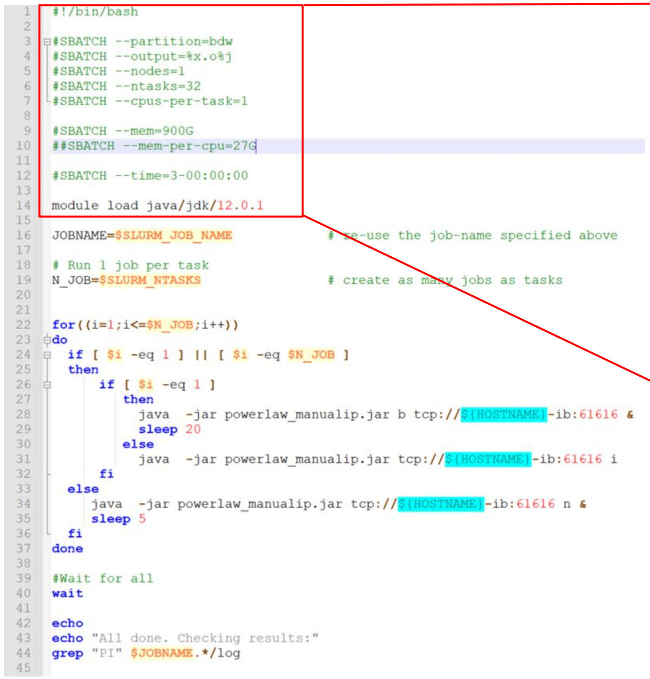
- On Windows systems, the most used tool is WinSCP (https://winscp.net/eng/docs/introduction).

scp example: **scp /path/to/local/file remote_user@remote_host:/path/to/remote/file**

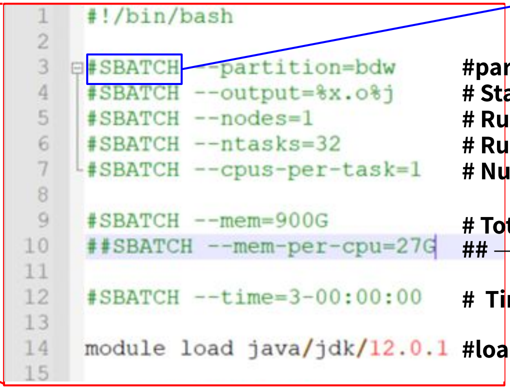
```
scp foo.file mattia.pellegrino@login.hpc.unipr.it:/hpc/group/G_SOWIDE/10-05-2022/python/NN/foo.file
```

PRACTICAL GUIDE

- To correctly launch a job on hpc you have to write an .sh script. This script is required to use SLURM functionalities



```bash
#!/bin/bash

#SBATCH --partition=bdw
#SBATCH --output=%x.o%j
#SBATCH --nodes=1
#SBATCH --ntasks=32
#SBATCH --cpus-per-task=1

#SBATCH --mem=900G
##SBATCH --mem-per-cpu=27G

#SBATCH --time=3-00:00:00

module load java/jdk/12.0.1

JOBNAME=$SLURM_JOB_NAME                          # re-use the job-name specified above

# Run 1 job per task
N_JOB=$SLURM_NTASKS                              # create as many jobs as tasks


for((i=1;i<=$N_JOB;i++))
do
  if [ $i -eq 1 ] || [ $i -eq $N_JOB ]
  then
        if [ $i -eq 1 ]
           then
             java  -jar powerlaw_manualip.jar b tcp://${HOSTNAME}-ib:61616 &
             sleep 20
           else
             java  -jar powerlaw_manualip.jar tcp://${HOSTNAME}-ib:61616 i
        fi
  else
      java  -jar powerlaw_manualip.jar tcp://${HOSTNAME}-ib:61616 n &
      sleep 5
  fi
done

#Wait for all
wait

echo
echo "All done. Checking results:"
grep "PI" $JOBNAME.*/log
```



```bash
#!/bin/bash

#SBATCH --partition=bdw
#SBATCH --output=%x.o%j
#SBATCH --nodes=1
#SBATCH --ntasks=32
#SBATCH --cpus-per-task=1

#SBATCH --mem=900G
##SBATCH --mem-per-cpu=27G

#SBATCH --time=3-00:00:00

module load java/jdk/12.0.1
```

This is not a comment!

#partition name
# Standard output and error log
# Run all processes on a single node
# Run a single task
# Number of CPU cores per task

# Total memory limit
## → comment

# Time limit hrs:min:sec

#load the desire module
module avail to see all module supported by the HPC

How to launch a script: sbatch <nomescript>.sh

Queue: squeue | grep <partition_name>
squeue | grep <username>
squeue | grep <job_id>

Delete a job: scancel <job_id>

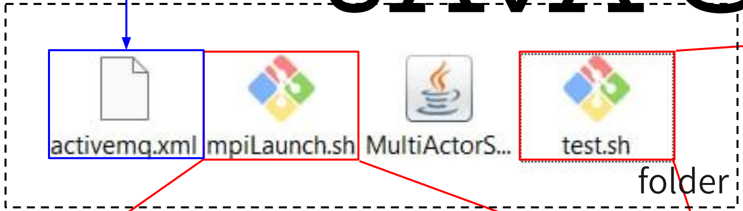
Job's details: scontrol show jobid <job_id>

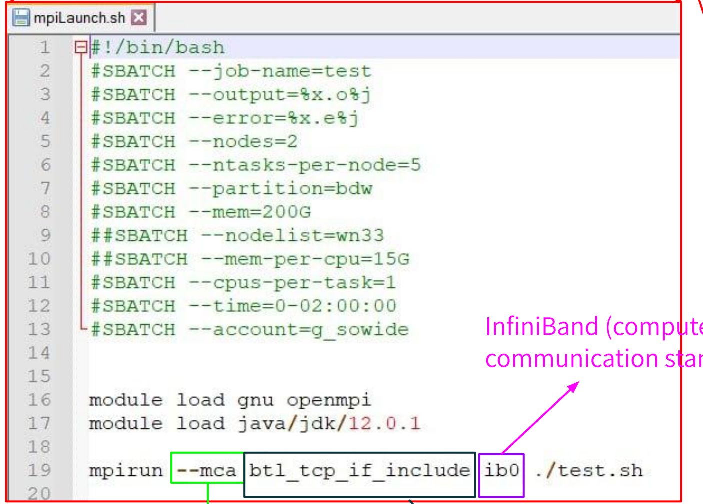
ACTODES

- Java **framework**
- Support the **actor** paradigm:
  - Simple and independent **agents**
  - Based on **message queue** and **exchange**
- It can be used in **distributed architecture**

JAVA ON HPC (actodes)

To ignore time limit!



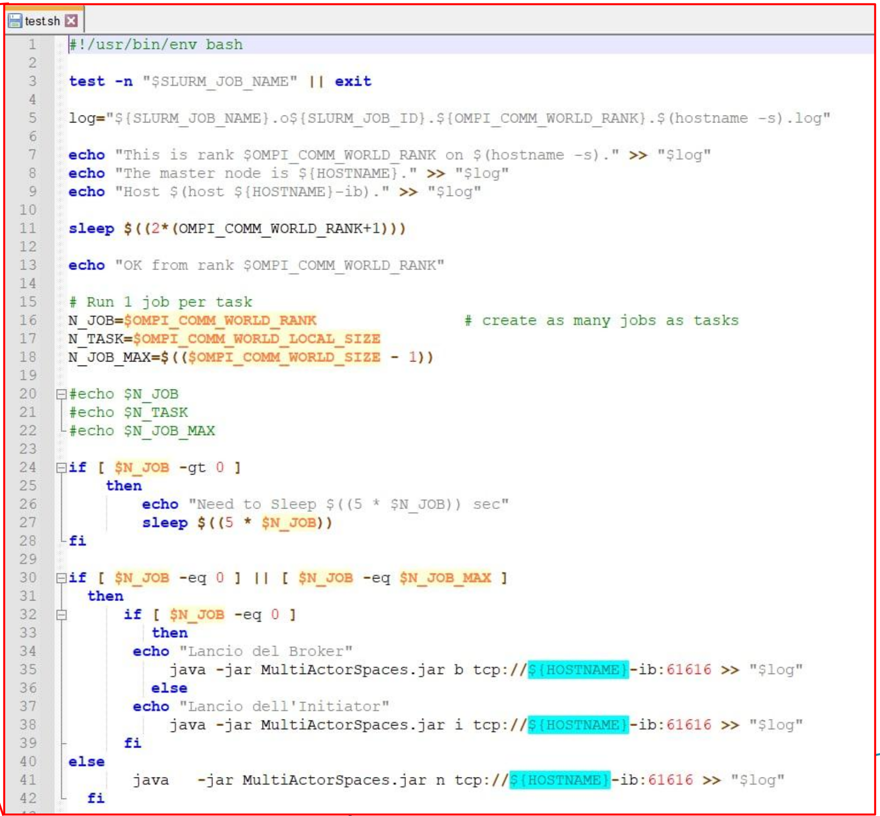

mpiLaunch.sh

```
#!/bin/bash
#SBATCH --job-name=test
#SBATCH --output=%x.o%j
#SBATCH --error=%x.e%j
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=5
#SBATCH --partition=bdw
#SBATCH --mem=200G
##SBATCH --nodelist=wn33
##SBATCH --mem-per-cpu=15G
#SBATCH --cpus-per-task=1
#SBATCH --time=0-02:00:00
#SBATCH --account=g_sowide


module load gnu openmpi
module load java/jdk/12.0.1

mpirun --mca btl_tcp_if_include ib0 ./test.sh
```

InfiniBand (computer networking communication standard)

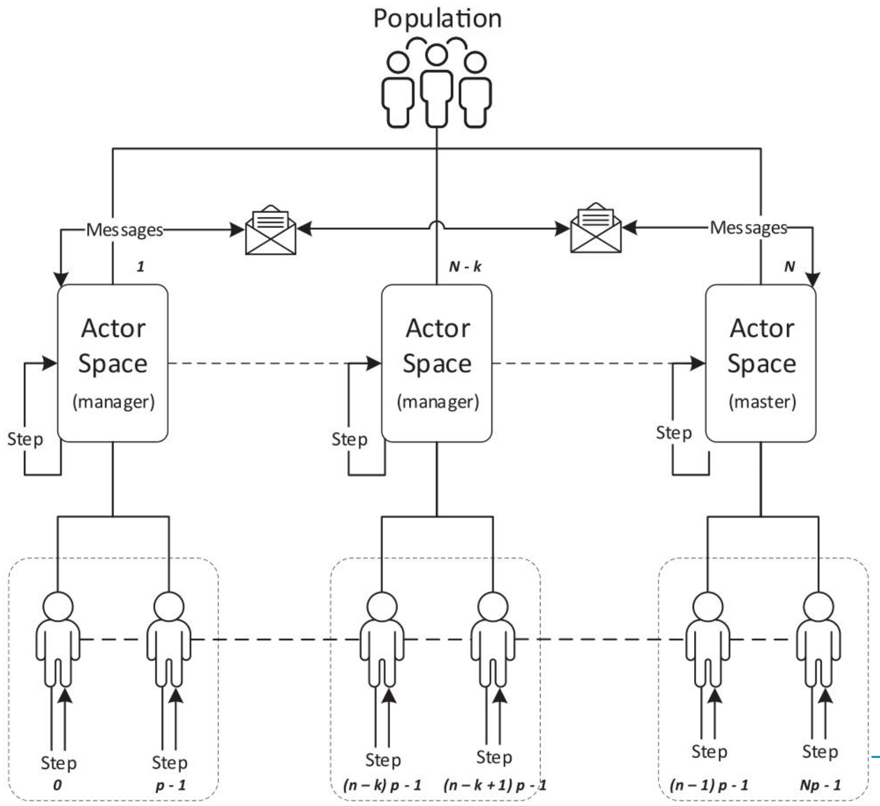
Modular component architecture

Use TCP protocol

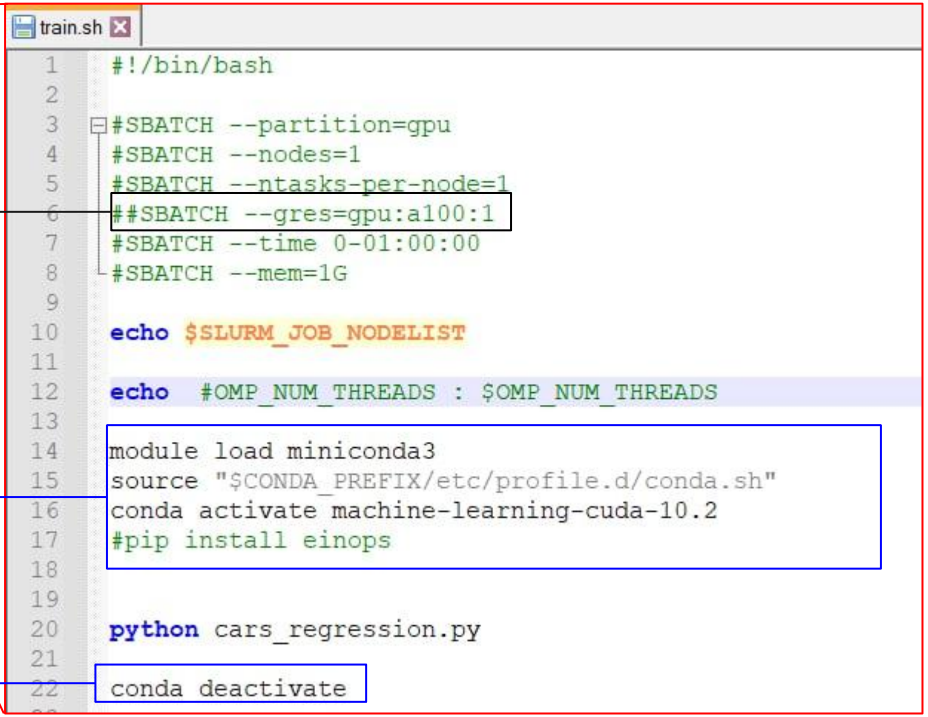

test.sh

```
#!/usr/bin/env bash

test -n "$SLURM_JOB_NAME" || exit

log="${SLURM_JOB_NAME}.o${SLURM_JOB_ID}.${OMPI_COMM_WORLD_RANK}.$(hostname -s).log"

echo "This is rank $OMPI_COMM_WORLD_RANK on $(hostname -s)." >> "$log"
echo "The master node is ${HOSTNAME}." >> "$log"
echo "Host $(host ${HOSTNAME}-ib)." >> "$log"

sleep $((2*(OMPI_COMM_WORLD_RANK+1)))

echo "OK from rank $OMPI_COMM_WORLD_RANK"

# Run 1 job per task
N_JOB=$OMPI_COMM_WORLD_RANK                # create as many jobs as tasks
N_TASK=$OMPI_COMM_WORLD_LOCAL_SIZE
N_JOB_MAX=$(($OMPI_COMM_WORLD_SIZE - 1))

#echo $N_JOB
#echo $N_TASK
#echo $N_JOB_MAX

if [ $N_JOB -gt 0 ]
    then
        echo "Need to Sleep $((5 * $N_JOB)) sec"
        sleep $((5 * $N_JOB))
fi

if [ $N_JOB -eq 0 ] || [ $N_JOB -eq $N_JOB_MAX ]
  then
      if [ $N_JOB -eq 0 ]
         then
     echo "Lancio del Broker"
         java -jar MultiActorSpaces.jar b tcp://${HOSTNAME}-ib:61616 >> "$log"
         else
     echo "Lancio dell'Initiator"
         java -jar MultiActorSpaces.jar i tcp://${HOSTNAME}-ib:61616 >> "$log"
      fi
else
       java   -jar MultiActorSpaces.jar n tcp://${HOSTNAME}-ib:61616 >> "$log"
  fi
```

ACTODES ON HPC



PYTHON for ML



```
#!/bin/bash

#SBATCH --partition=gpu
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
##SBATCH --gres=gpu:a100:1
#SBATCH --time 0-01:00:00
#SBATCH --mem=1G

echo $SLURM_JOB_NODELIST

echo  #OMP_NUM_THREADS : $OMP_NUM_THREADS

module load miniconda3
source "$CONDA_PREFIX/etc/profile.d/conda.sh"
conda activate machine-learning-cuda-10.2
#pip install einops


python cars_regression.py

conda deactivate
```

You can select a specific GPU

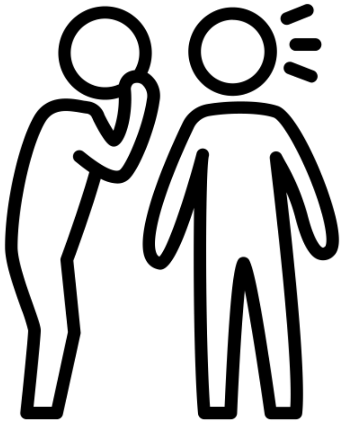
Load ML resources

Release ML resources

GOSSIP ALGORITHM

Exercise

- Systems-oriented **computational and communication paradigm** distributed on a **large scale** with high dynamic characteristics
- Probabilistic approach
- Main features:
    - **Simplicity**
    - **Scalability**
    - **Efficiency**
    - **Robustness**



EXERCISE

A Gossip Algorith

- Modify the source code available at the following address: https://github.com/sowide/HPC_practice

- Each agent involved in the simulation can be: **infected or healthy**

- Agents meet each other **randomly**

- Every time an **interaction** occurs, there is a chance that an **infected** agent **will infect** a **healthy** one

EXERCISE

- Classes and function to complete:
    - **Phases.class**
    - **PersonManager.class**:
        - BuildPopulation(), infectionInPartition(), setinfectedpeople(),saveMeasure()
    - **Manager.class**
        - messageDataHandler()

- When the simulator is ready (**test it locally with 100K agents**) launch the simulator on the HPC **with 1M of agents**, utilizing **8,16,24,32** cores. Finally, measure the temporal performance
- **Extra**: Add 1 or more personalized phases (you're free to decide which fases you want)

- Deliver source code: https://shorturl.at/bfBJR [name_surname.zip]

EXERCISE

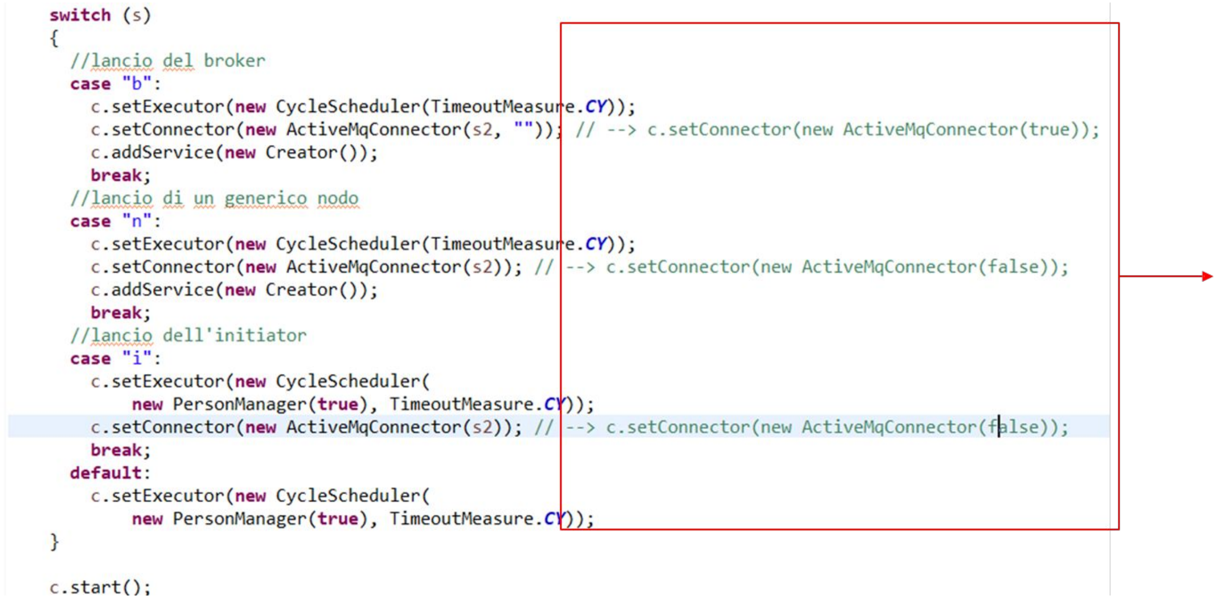
Login

- **Login and take reservation**:
  - ssh <nomeutente>@login.hpc.unipr.it
  - ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
  - cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
  - chmod 0600 ~/.ssh/authorized_keys
  - newgrp T_2022_SISTEMI_DISTRIBUITI

  - **Aggiungere allo script mpiLunch.sh**:
    - #SBATCH --reservation=t_2022_sistemi_distribuiti-20230418
    - #SBATCH --account=t_2022_sistemi_distribuiti

Local Execution

Initiator Class



```
switch (s)
{
  //lancio del broker
  case "b":
    c.setExecutor(new CycleScheduler(TimeoutMeasure.CY));
    c.setConnector(new ActiveMqConnector(s2, "")); // --> c.setConnector(new ActiveMqConnector(true));
    c.addService(new Creator());
    break;
  //lancio di un generico nodo
  case "n":
    c.setExecutor(new CycleScheduler(TimeoutMeasure.CY));
    c.setConnector(new ActiveMqConnector(s2)); // --> c.setConnector(new ActiveMqConnector(false));
    c.addService(new Creator());
    break;
  //lancio dell'initiator
  case "i":
    c.setExecutor(new CycleScheduler(
        new PersonManager(true), TimeoutMeasure.CY));
    c.setConnector(new ActiveMqConnector(s2)); // --> c.setConnector(new ActiveMqConnector(false));
    break;
  default:
    c.setExecutor(new CycleScheduler(
        new PersonManager(true), TimeoutMeasure.CY));
}

c.start();
```

- **s2** is the string that contains the IP address of the broker: **tcp://x.x.x.x:port** (es. tcp://127.0.0.1:61616)
- To run the simulator on local machine you can also change the object argument **ActiveMqConnector**
- To launch the simulator correctly you need to:
  - Launch a broker instance **(b)**
  - Launch one or more generic instances **(n)**
  - Launch an initiator instance to start the simulation (master) **(i)**