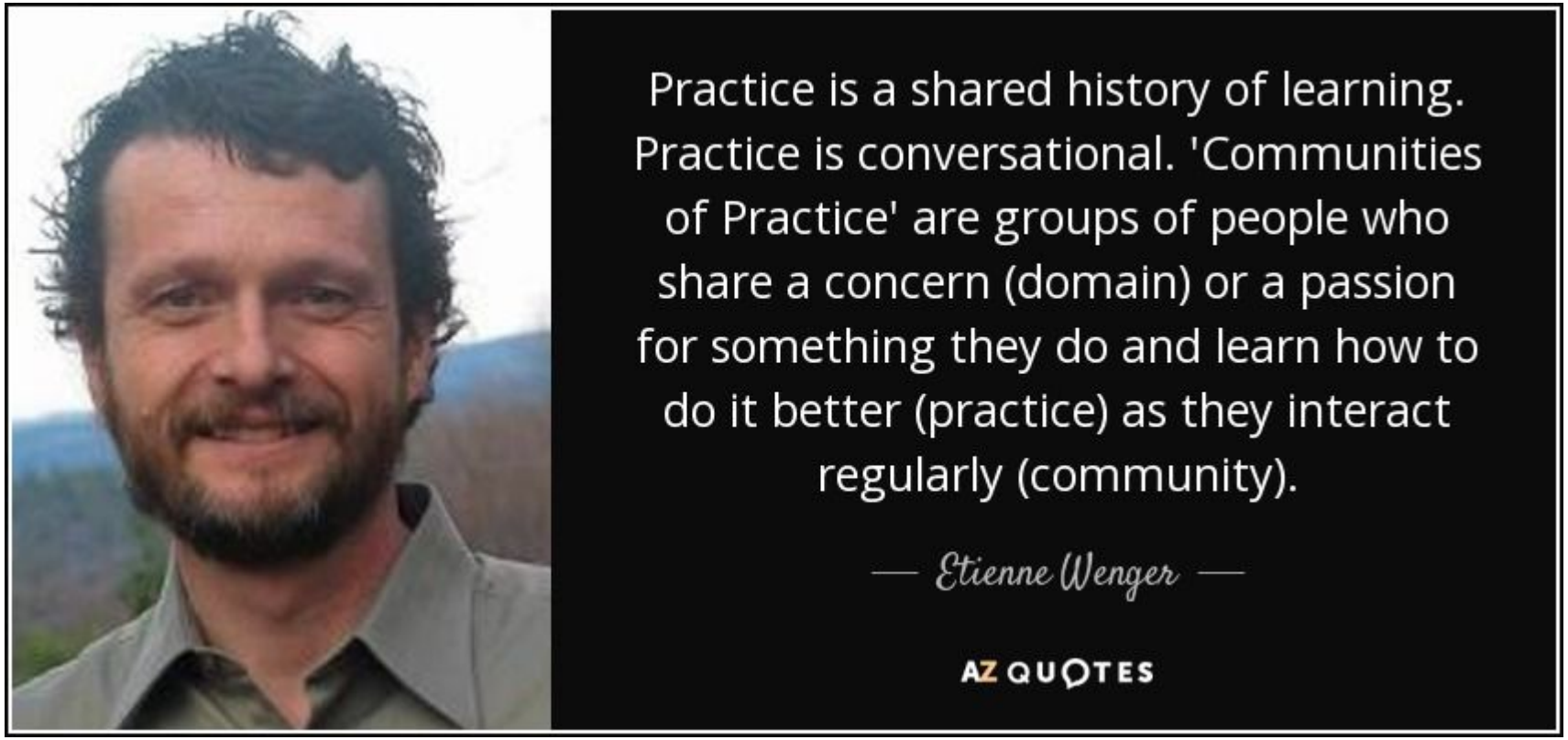
Introducción a ambientes HPC: SLURM

Jaime Ibarra Nuño

Octubre 2024

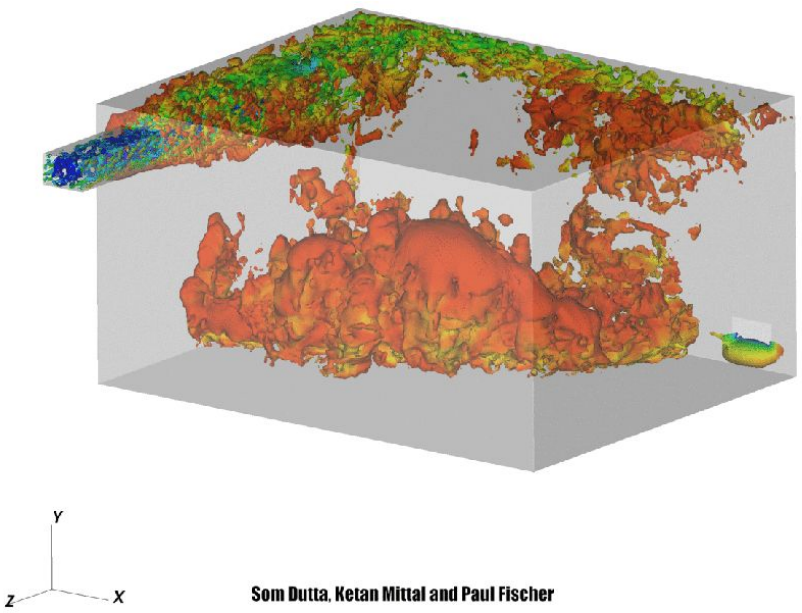

Practice is a shared history of learning. Practice is conversational. 'Communities of Practice' are groups of people who share a concern (domain) or a passion for something they do and learn how to do it better (practice) as they interact regularly (community).
— Etienne Wenger —
AZ QUOTES

Contenido

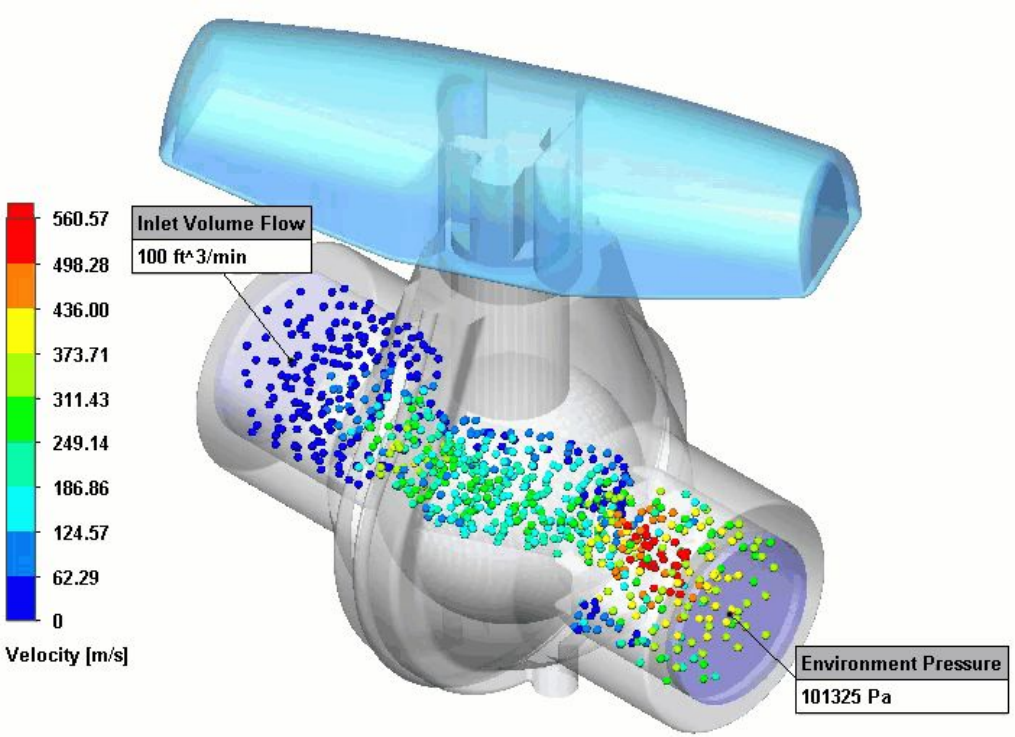
- Introducción
- Conexión remota ssh/vpn
- Nodo login / working node
- Paralelización
- Ambientes: Modulos
- SLURM
- Envío de trabajos (tiempo real): srun
- Monitoreo y cancelación de trabajos
- Envió de trabajos vía script

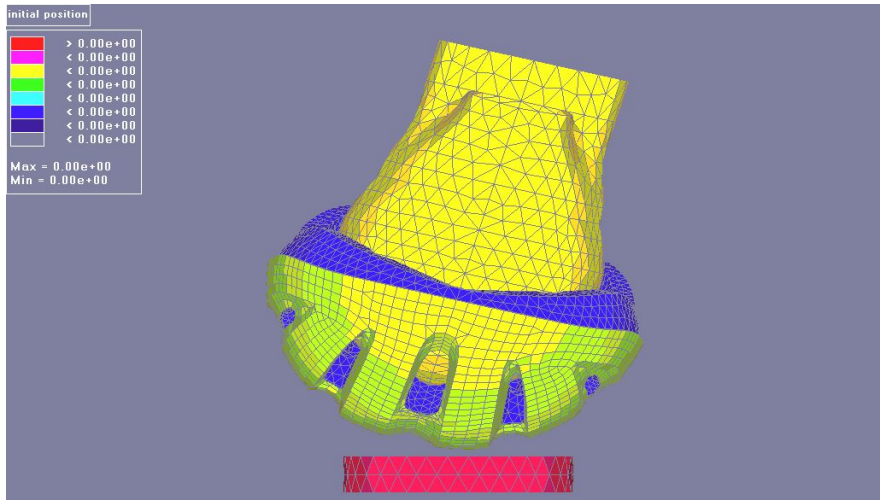
Supercómputo (High Performance Computing)

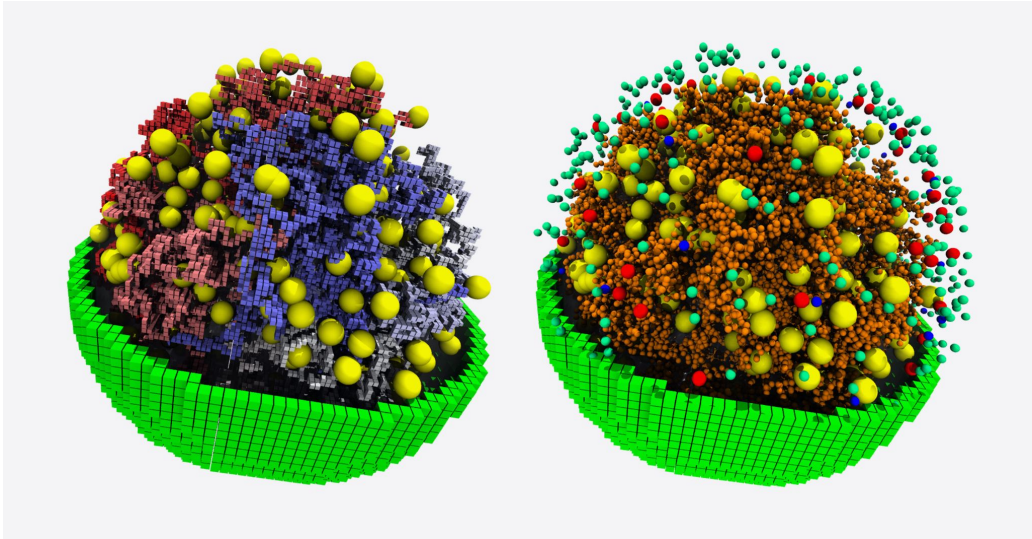
El nombre supercómputo, también llamado cómputo de alto rendimiento (*HPC* en inglés) se refiere al uso de las computadoras interconectadas y utilizan técnicas informáticas avanzadas para resolver problemas complejos que requieran procesar gran cantidad de datos. Algo imposible de realizar en una computadora de escritorio.



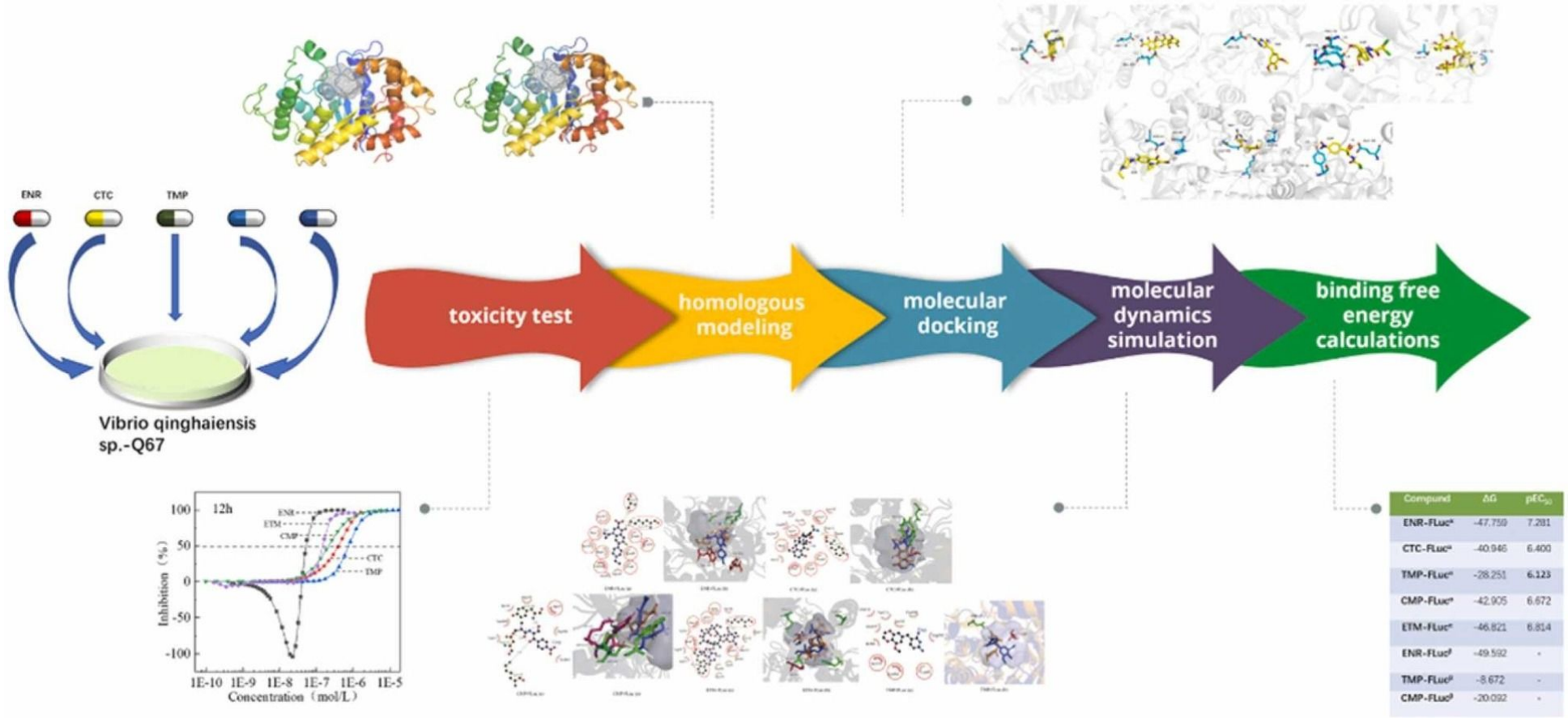
A supercomputer rendering of airflow in an empty hospital room, where the vertical velocity of air is high enough to keep the virus-laden aerosols in suspension.
Image Credit: IBM







As the simulation of the minimal cell JCVI-syn3A grows and divides, the model tracks the movements and interactions of the cell's components, including its ribosomes (yellow spheres) and specific membrane complexes and proteins (red, blue and green spheres), all wrapped up inside its cell membrane (green cubes).



Compund	ΔG	pEC_{50}
ENR-Fluc[a]	-47.759	7.281
CTC-Fluc[a]	-40.946	6.400
TMP-Fluc[a]	-28.251	6.123
CMP-Fluc[a]	-42.905	6.672
ETM-Fluc[a]	-46.821	6.814
ENR-Fluc[b]	-49.592	-
TMP-Fluc[b]	-8.672	-
CMP-Fluc[b]	-20.092	-

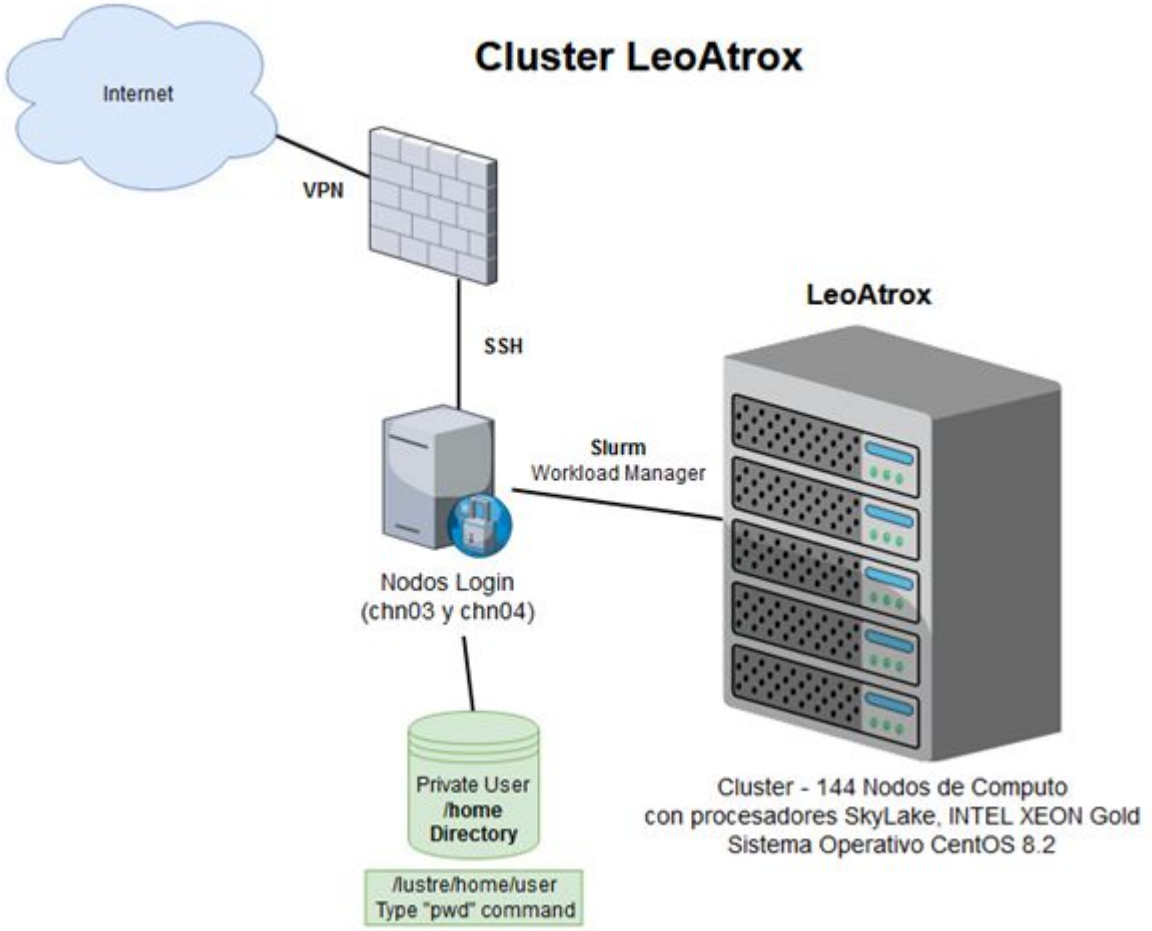
Partes de clúster HPC

Consiste en varios servidores de cómputo conectados a una red.

A cada servidor de cómputo se le denomina **nodo**.

Aquí, cada nodo de cómputo tiene **núcleos** (cores).

Cores son los encargados de realizar los procesos y tareas.

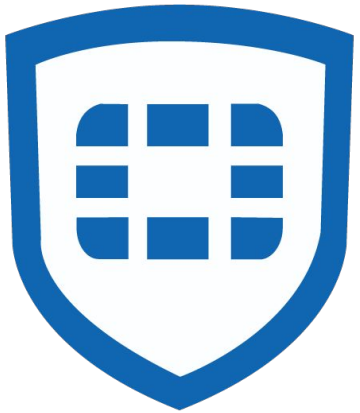


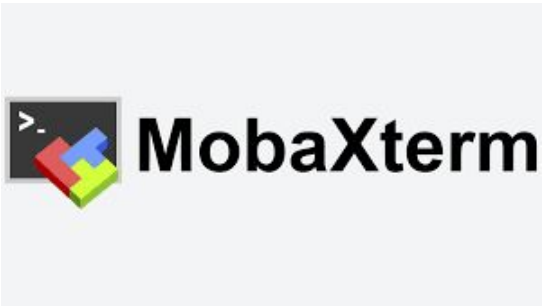
Protocolo SSH

Descargar forticlient:
Linux
https://www.fortinet.com/support/product-downloads
Windows
https://shorturl.at/nL245
MAC FortiVPN 6
http://148.202.15.34/files/

Descargar mobaXterm

Windows
https://mobaxterm.mobatek.net/download-home-edition.html





Openfortivpn https://github.com/adrienverge/openfortivpn

Conexión protocolo ssh

```
$ ssh cursoXX@login1.cads.udg.mx
```

PuTTY

PowerShell

MobaXterm

bitvise

BASH
THE BOURNE-AGAIN SHELL

```
rsync -vp ~/archivo.txt  [usuario]@login1.cads.udg.mx:/lustre/home/usuario/

rsync -avnr   [usuario]@login1.cads.udg.mx:/lustre/home/usuario/archivo.txt
/home/local/
```

bitvise

Filezilla

MobaXterm

Conectarse a una sesión remota

Información de la VPN

- **Username:** cursos@cads.udg.mx
- **Contraseña:** UDG.leo2024
- **Gateway:** vpn.cads.udg.mx
- **Puerto:** 443

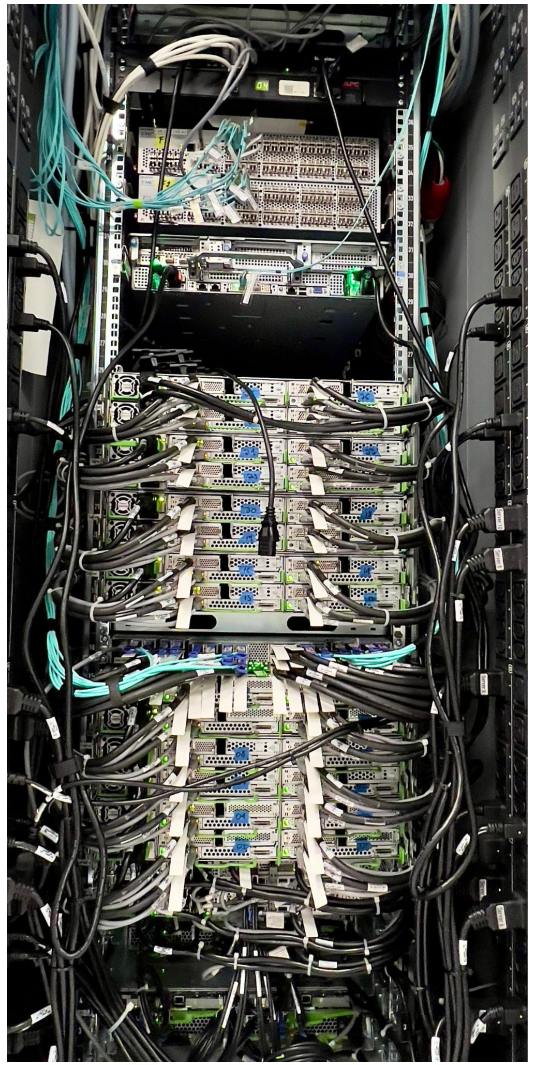
Información para ssh

- **Dirección IP:** login1.cads.udg.mx
- **Usuario/Hostname:** curso[1-20]
- **Contraseña:** CADS.2024
- **Puerto:** 22

```
[unix]$ ssh curso1@login1.cads.udg.mx
```

Leo Átrox

- 140 nodos de cómputo (CPU)
  - Dos procesadores Intel Xeon Gold 16 cores c/u
- 4 nodo login
  - fat nodes
- 2 nodos GPU
  - Nvidia Tesla P100 con 2 procesadores intel Xeon c/u
- 4 Intel Xeon Phi



(Simple Linux Utility Resource Manager) SLURM

- Gestor de recursos
- Administra los recursos de las particiones.
- Gestiona las tareas en ejecución y en espera en el clúster.
- Reserva recursos compartidos.
- Ejecución de tareas hasta por 6 días (Leo Átrox)

Estado de particiones

```
$ sinfo
```

```
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
q1*          up 7-00:00:00      5    mix cn[011,015,023,082,097]
q1*          up 7-00:00:00     42  alloc cn[001-010,012-014,016-022,
q1*          up 7-00:00:00      2   idle cn[095-096]
q3           up 8-00:00:00     20   idle cn[029-040,042-049]
gpu          up 15-00:00:0      2   idle nvd[01-02]
matlab       up 8-00:00:00      4   idle cnf[141-144]
```

- *mix*: parcialmente en uso
- *idle*: disponible
- *alloc*: no disponible

LEO ÁTROX
CENTRO DE ANÁLISIS DE DATOS Y SUPERCÓMPUTO

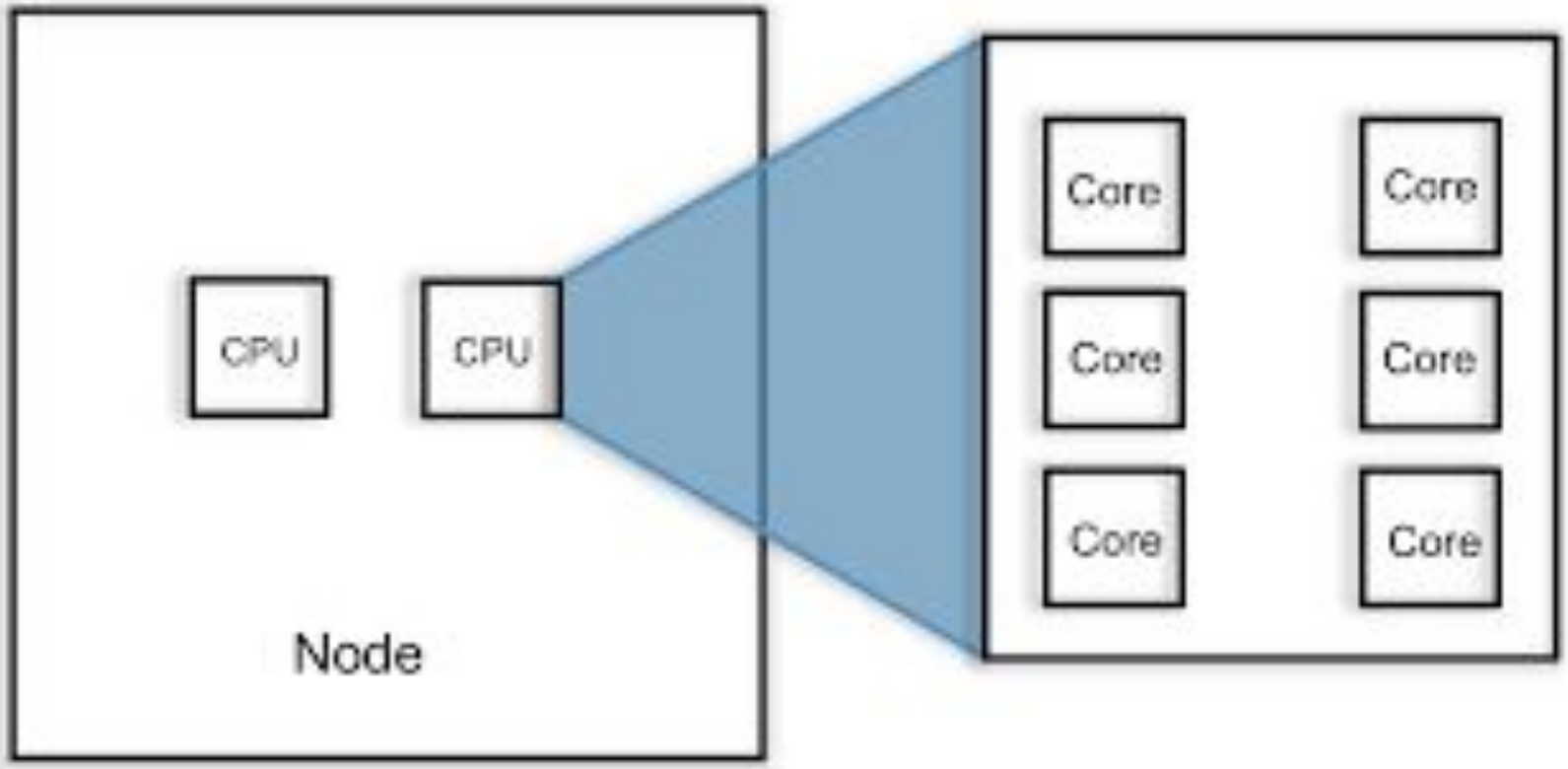
Nodo login:

- chn03

Nodos de trabajo:

- test cn[115-116]
- q1 cn[001–114]
- gpu nvd[01-02]
- matlab cnf[141-144]

Nodo & Cores



https://iam.auckland.ac.nz/profile/SAML2/Redirect/SSO?execution=e2s1

Nodo login

```
[usuario@chn03 ~]$ hostname
chn03.leoatrox.cads.udg.mx
```

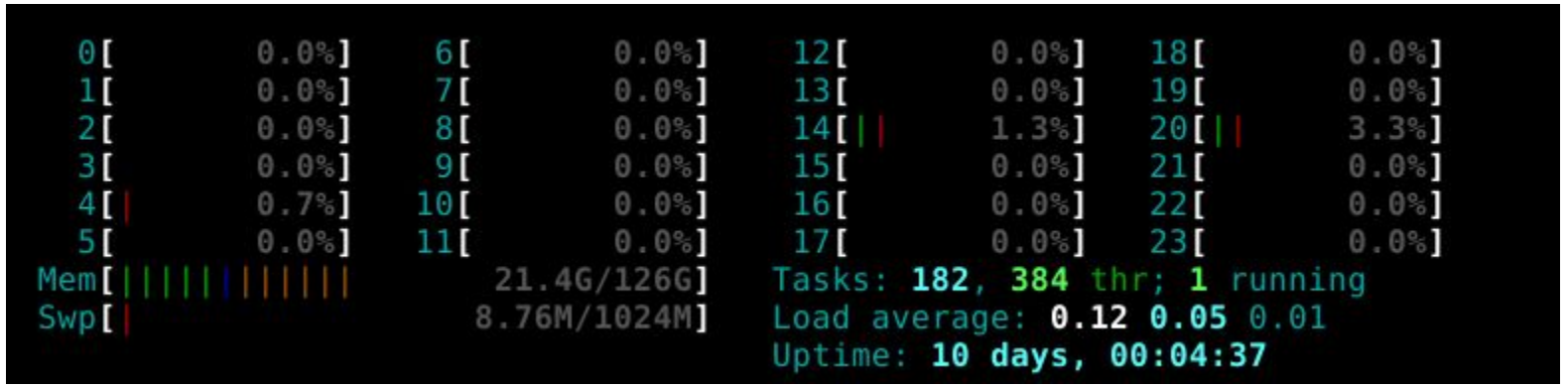
La ruta de trabajo (power working directory, PWD)

```
[usuario@chn03 ~]$ pwd
/lustre/home/usuario
```

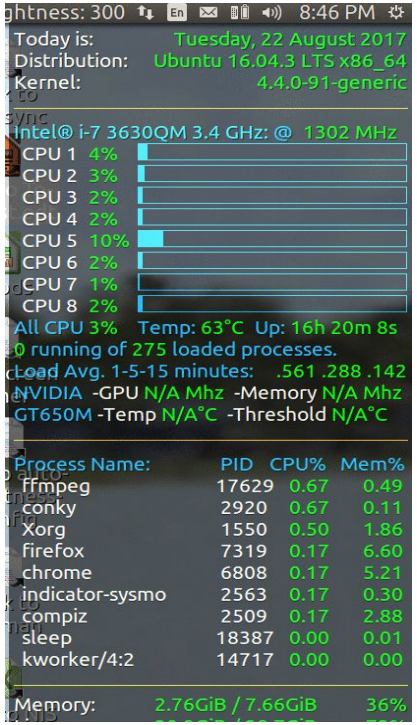
Núcleos (*cores*)

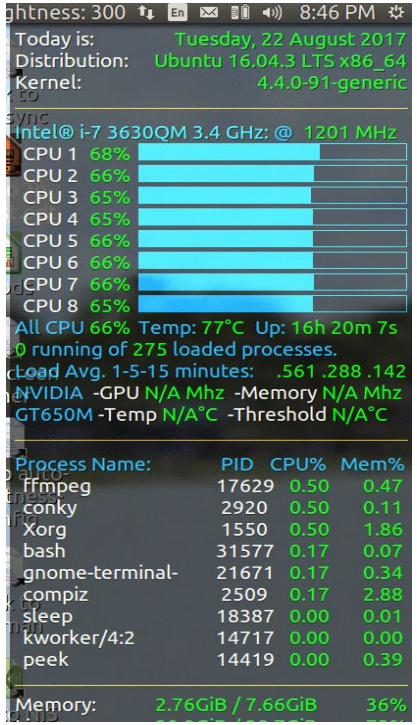
Comando para monitorear o ver el uso de recursos

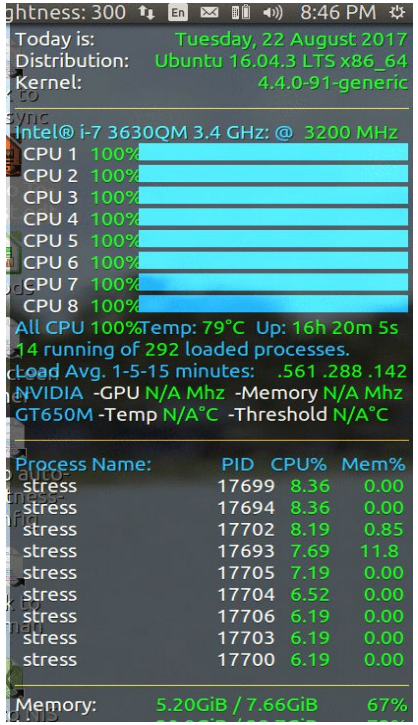
```
$ htop
```



```
$ ssh <node_trabajo>
```

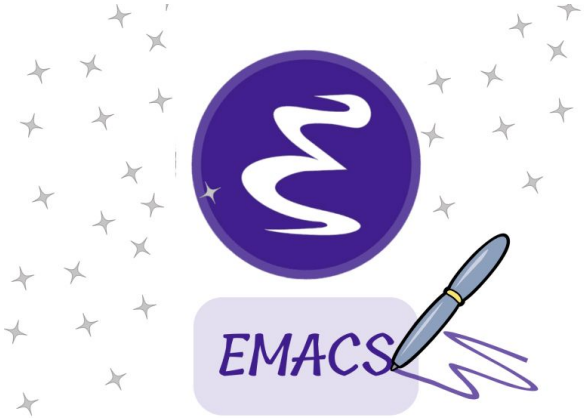






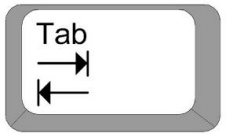
https://askubuntu.com/questions/948854/how-do-i-stress-test-cpu-and-ram-at-the-same-time

Herramientas en Leo Átrox



Tip:

- tecla tab para autocomplementar
- Doble tab muestra lista de opciones



Interface remota gráfica

CODE
Visual Studio Code

MATLAB®

jupyter

VSCodium

Ambiente

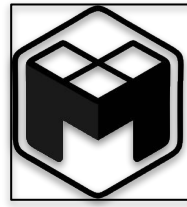
En un clúster los recursos son compartidos: se tiene una gran variedad de programas y versiones. Para poder trabajar es necesario crear un ambiente de trabajo.

Algunas soluciones para crear ambientes de trabajos son:

- modulos
- contenedores (singularity/docker)
- anaconda/miniconda

SINGULARITYCE

CONDA®



ENVIRONMENT MODULES

Modulos

Visualizar programas instalados:

```
$ module av
```

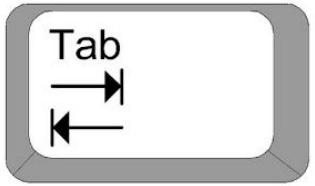
```
---------- /lustre/spack/share/spack/modules/linux-centos8-broadwell ----------
   alsa-lib-1.2.3.2-gcc-10.2.0-poaz4qc
   arpack-ng-3.7.0-gcc-10.2.0-lhrowop
   autoconf-2.69-gcc-10.2.0-u62dfgf
   autoconf-2.69-gcc-8.3.1-ttljhrz
   autoconf-archive-2019.01.06-gcc-10.2.0-ro2zesf
   autoconf-archive-2019.01.06-gcc-8.3.1-sw7ehtu
   autodock-vina-1_1_2-gcc-10.2.0-lvpl2qb
   automake-1.16.2-gcc-10.2.0-656ahcs
   automake-1.16.2-gcc-8.3.1-h7da7nn
```

Ejemplo

```
$ python
$ module load python- #(press tab 2 veces)
$ python-3.8.6-gcc-10.2.0-snp3iil
$ python --version
$ python
>> exit()
$ module list
$ module purge
```

Tip: usar la tecla Tab para autocompletar comandos o rutas.

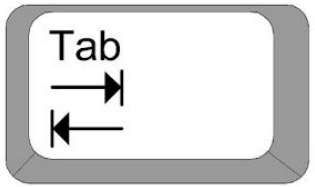
- presionar una vez para autocomplementar
- presionar más de una vez para mostrar lista de opciones.



Comando (ml)	Función	Ejemplo
module av	muestra la lista de programas instalados	`$ module av`
module load	cargar programas	`$ module load julia/1.6.3`
module list	muestra programas cargados	`$ module list`
module unload	para quitar un modulo disponibles (específico)	`$ module unload julia/1.6.3`
module purge	limpia todos los modulos cargados	`$ module purge`

Tip: usar la tecla tab para autocomplementar comandos o programas.
Ejemplo: escribir las primeras letras y luego:

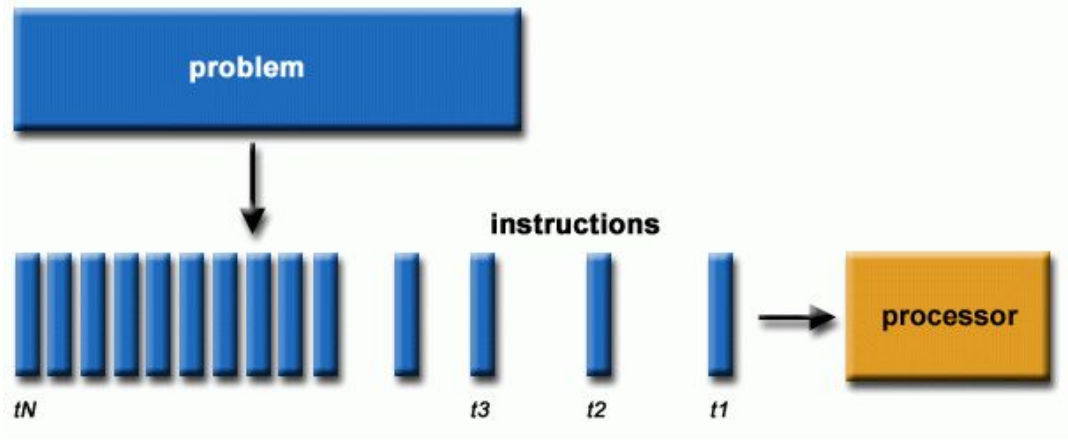
- presionar una vez para autocompletar
- presionar más de una vez para mostrar lista de opciones.

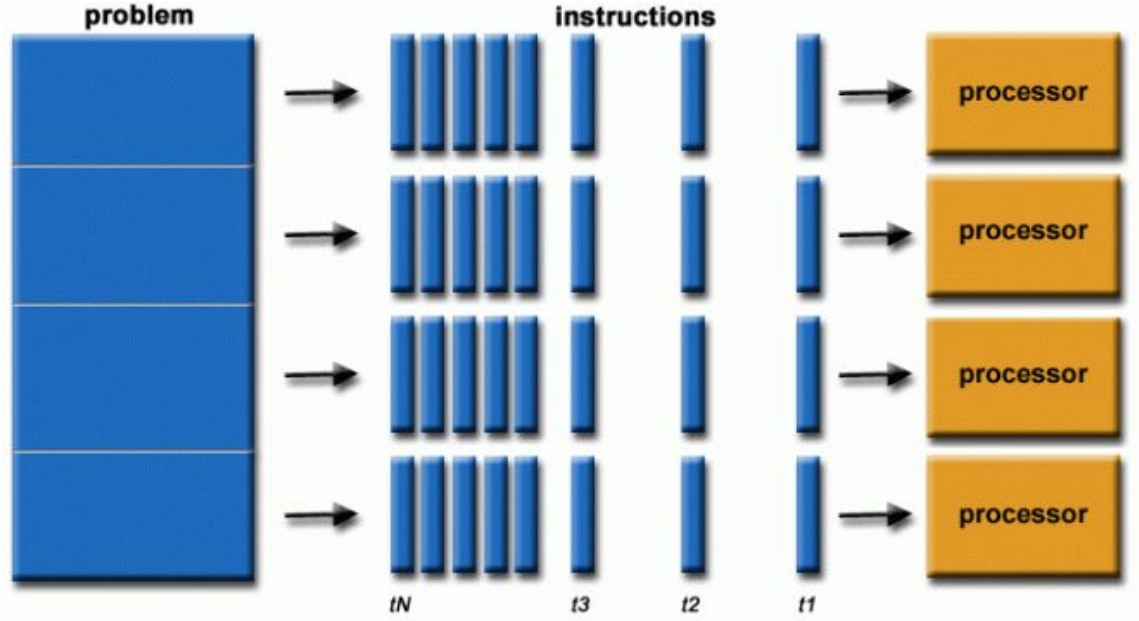


Ejercicio: Ambiente bash

1. ¿Nombres de las particiones tengo acceso?
2. Mencionar mínimo un nodo que este:
   a. disponible
   b. ocupado
   c. parcialmente en uso
3. Modulos
   a. Cuantas versiones de Python están instalados?
   b. cargar 2 módulos: ejemplo: julia, Python, Emacs, Singularity
   c. ver lista de modulos cargados
   d. limpiar todos los modulos.

Comando (ml)	Función
module av	muestra la lista de programas instalados
module load	cargar programas
module list	muestra programas cargados
module unload	para quitar un modulo disponibles (específico)
module purge	limpia todos los modulos cargados





Serial

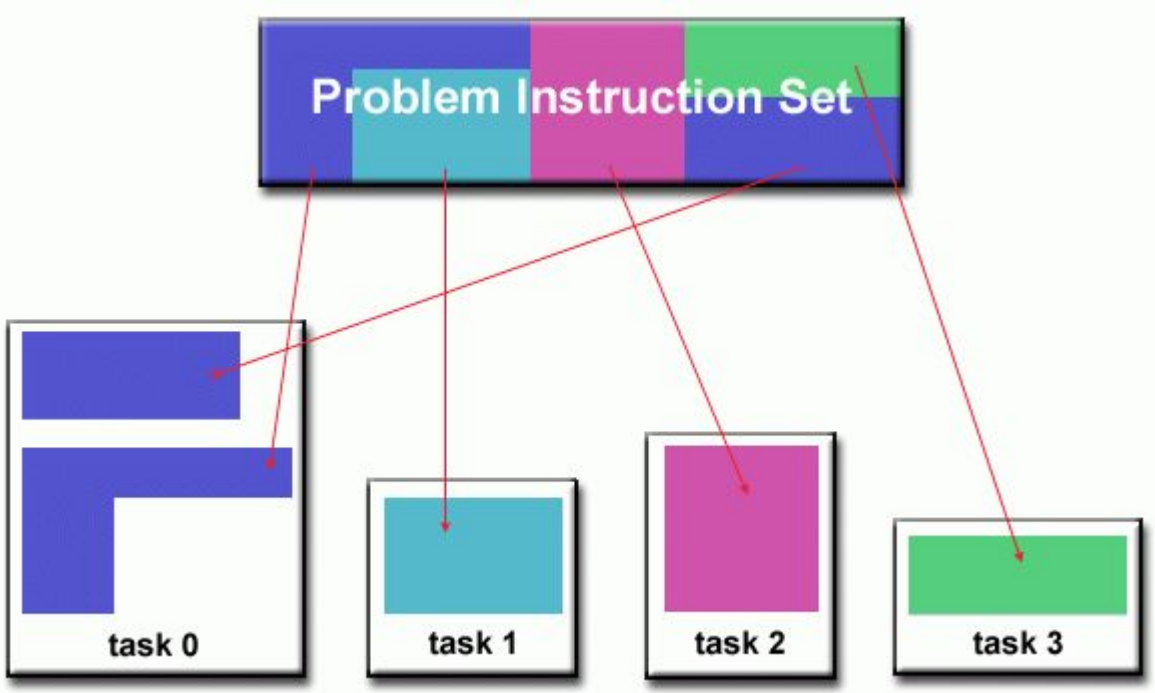
Paralelo

https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial

Computación paralela

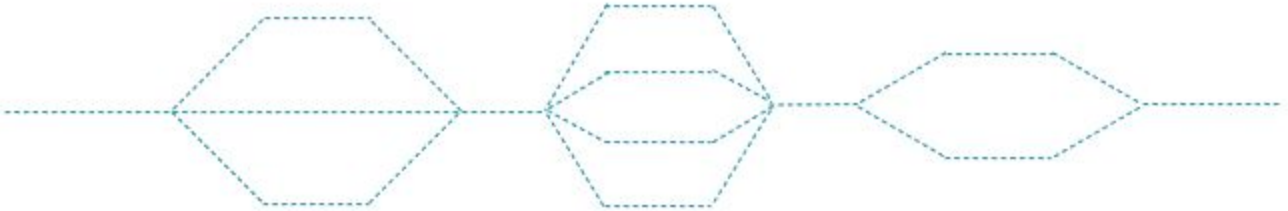
Es el uso simultáneo de múltiples recursos computacionales para resolver un problema:

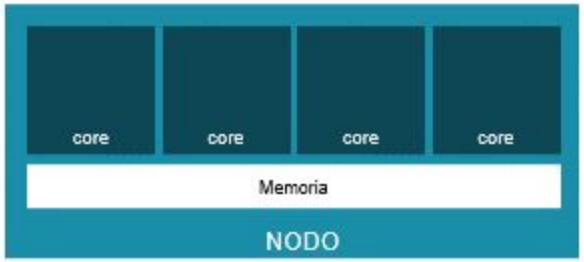
- Cada parte se descompone en una serie de instrucciones.
- Las instrucciones de cada parte se ejecutan simultáneamente en diferentes procesadores.



https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial

Modelo de programación por hilos





Work - steal

slurm
workload manager

Comandos útiles de SLURM

Parámetro	Uso en script	Acción
-J	-J prueba	asignación de nombre a la tarea
-p	-p q1	indica la partición a utilizar
-n	-n 2	número de tareas o procesos
-c	-c 4	CPUs por tarea
-N	-N 2	número de nodos
- -ntasks-per-node	- -ntasks-per-node=2	número de tareas por nodo
- -mem-per-cpu	- -mem-per-cpu=2300	memoria por CPU
-o	-o salida_%j.out	archivo de salida
-e	-e error_%j.out	archivo de errores

Ejecutar trabajos tiempo real

Comando **hostname**: nos permite conocer el nombre del equipo o IP.
En Leo Átrox nos ayuda a conocer el nodo donde se ejecuta una tarea.
Ejemplo:

```
$ hostname
chn03.leatrox.cads.udg.mx
```

```
$ srun -p q1 hostname
cn080.leatrox.cads.udg.mx
```

Demostración

```
$ srun -p gpu hostname
```

```
$ srun -p q1 -N 2 hostname
```

```
$ srun -p q1 -n 2 hostname
```

```
$ srun -p q1 -n 40 hostname
```

```
$ srun -p q1 -N 2 --ntasks-per-node=2 hostname
```

Ejercicios

1. ¿Qué ocurre si no especifico la partición en la que quiero ejecutar mi comando?
2. Ejecute el comando **hostname** en la partición **q1** con **srun**:
   a. Con un unico proceso.
   b. Con dos procesos iguales.
   c. Con dos procesos en distintos nodos.
3. En la partición **q1**
   a. ¿Cuántos cores puedo reservar por proceso? ¿Por qué ese número?
   b. ¿Qué ocurre si reservo más cores de los disponibles?

Uso en script	Acción
-J prueba	asignación de nombre a la tarea
-p q1	indica la partición a utilizar
-n 2	número de tareas o procesos
-c 4	CPUs por tarea
-N 2	número de nodos
- -ntasks-per-node=2	número de tareas por nodo
- -mem-per-cpu=2300	memoria por CPU
-o salida_%j.out	archivo de salida
-e error_%j.out	archivo de errores

Solución

1. hostname
2. srun
   a. srun -p q1 hostname
   b. srun -p q1 -n 2 hostname
   c. srun -p q1 -- ntasks-per-node 2 hostname

Script: númbers.slurm

```
#!/bin/bash

#SBATCH -p q1                    # Nombre de la particion
#SBATCH -J numbers               # Nombre del trabajo
#SBATCH -n 1                     # Numero de núcleos(cores)
#SBATCH -t 0-02:00               # Duración (D-HH:MM)
#SBATCH -o salida_%j.txt         # Salidas o impresiones
#SBATCH -e error_%j.txt          # Errores o warning

for i in {1..1000000}; do

echo $RANDOM >> SomeRandomNumbers.txt

done
```

https://github.com/jinleon/SLURM/tree/63b13607296ab7b65c80bc1fe8f5a5980a6132f8/basic/

Ejecución de trabajos con script

```
$ sbatch numbers.slurm
$ ls
$ cat SomeRandomNumbers.txt
$ head SomeRandomNumbers.txt
$ tail SomeRandomNumbers.txt
```

```
$ sacct -X
```

```
(base) [joelgl@chn03 ~]$ sacct -X
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
254596               comm.py        q1     joelgl       1188  COMPLETED      0:0
```

Monitorear y cancelar tareas: sleep.slurm

```
#!/bin/bash

#SBATCH -p q1                    # Nombre de la particion
#SBATCH -J sleep                 # Nombre del trabajo
#SBATCH -n 1                     # Numero de nucleos(cores)
#SBATCH -t 0-02:00               # Duracion (D-HH:MM)
#SBATCH -o salida_%j.out         # Salidas o impresiones
#SBATCH -e error_%j.err          # Errores o warnings

sleep 180

hostname
```

Comandos:

```
$ sbatch sleep.slurm
Submitted batch job 376793
$ squeue -u $USER
$ watch squeue -u $USER
crt+c (salir)
$ scancel 376793
$ squeue -u $USER
```

script error.slurm

```
#!bin/bash

#SBATCH -p q1                       # Nombre de la particion
#SBATCH -J Sing                      # Nombre del trabajo
#SBATCH -n 1                        # Numero de nucleos(cores)
#SBATCH -t 0-02:00                  # Duracion (D-HH:MM)
#SBATCH -o salida_%j.txt            # Salidas o impresiones
#SBATCH -e ejemplo_error.txt      # Errores o warnings

module load singularity-3.6.4-gcc-8.3.1-qsfb5jh

singularity run lolcow.sif
sleep 180
hostname
```

```
$ sbatch singularity.slurm
Submitted batch job 376811
$ cat ejemplo_error.txt
```

Cargar módulos: python.slurm

```
#!/bin/bash

#SBATCH -p q1                    # Nombre de la particion
#SBATCH -J python                # Nombre del trabajo
#SBATCH -n 1                     # Numero de nucleos(cores)
#SBATCH -t 0-02:00               # Duracion (D-HH:MM)
#SBATCH -o salida_%j.txt         # Salidas o impresiones
#SBATCH -e error_%j.txt          # Errores o warnings

module load python-3.8.6-gcc-8.3.1-7uu5gyz

python hola.py
```

Script hola.py

```
quote = (10*("*")+ ("\n")  + "Hola Mundo \n"+ 10*("*")+("\n"))

#print("hola mundo")

with open('mundo.txt', 'w', encoding='utf-8') as f:
    f.write(quote)
```

```
$ sbatch python.slurm
Submitted batch job 376793
$ cat mundo.txt
```

Gracias!!