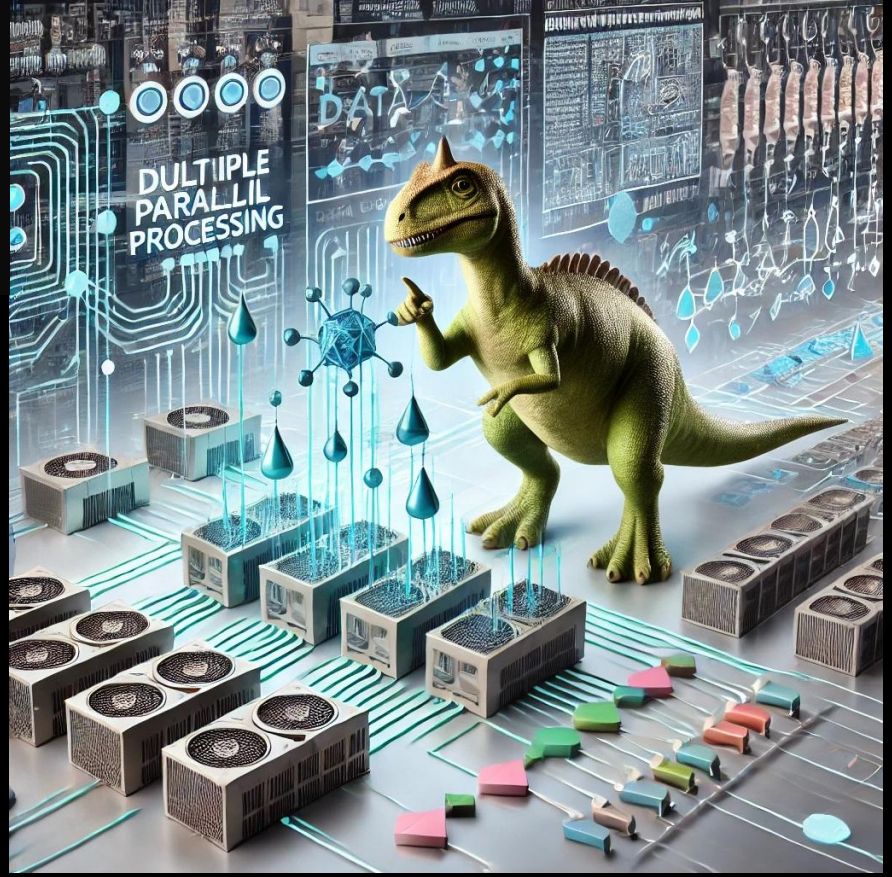
GCU
GRUPO COLOSUNGS
computación paralela



Lic. Martha Semken - Lic. Mariano Vargas
Tutores: Ignacio Tula, Zoe Sacks, Gabriel Althaparro

Workshop: día 2

Repasando:

Cómputo paralelo consiste en resolver problemas dividiéndolos en partes que se ejecutan simultáneamente.

- Por qué es importante:
  - Reducción del tiempo de ejecución.
  - Escalabilidad: Aprovechar los recursos de hardware modernos (multinúcleo y clústeres).
  - Ejemplos de aplicaciones: Modelos climáticos, simulaciones físicas, inteligencia artificial, etc.

Recordando la clasificación

- **Memoria compartida** (usado por …): Todos los núcleos acceden a una memoria común.
- **Memoria distribuida** (usado por OpenMPI): Cada nodo tiene su propia memoria y los nodos se comunican mediante paso de mensajes.
- **Programación híbrida**: No llegamos a verlo en este workshop

Arquitectura Beowulf: Maestro/esclavo

Cómo funciona un clúster con arquitectura maestro/esclavos:

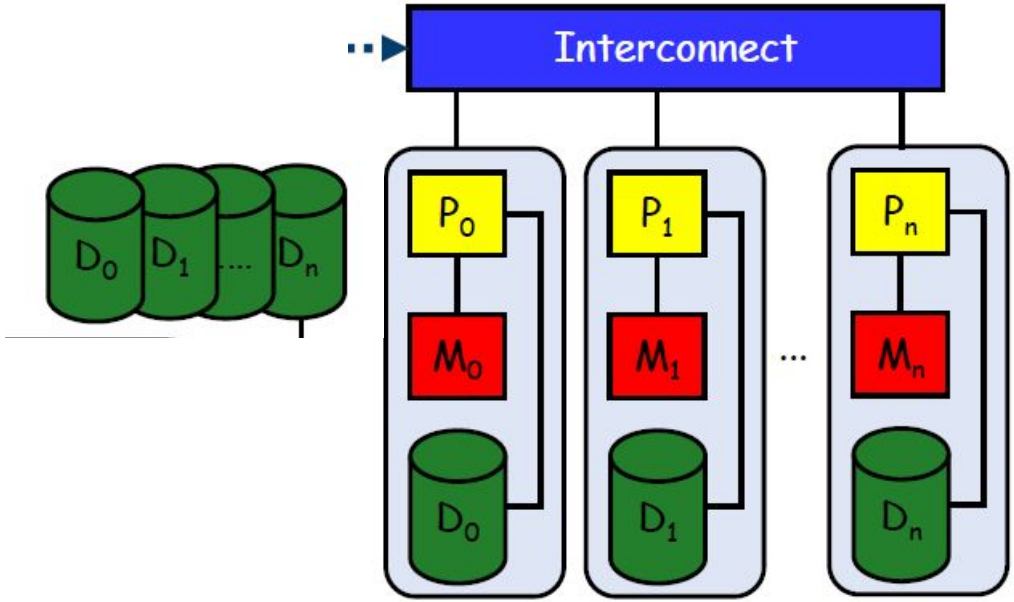
1. **Nodo maestro**:

- Administra los recursos del clúster.
- Coordina y distribuye las tareas a los nodos esclavos.
- Corre servicios importantes como SLURM Controller y, a menudo, NFS (sistema de archivos compartido).

2. **Nodos esclavos:**

- Realizan el trabajo asignado por el maestro.
- Ejecutan las tareas paralelas y reportan el progreso.

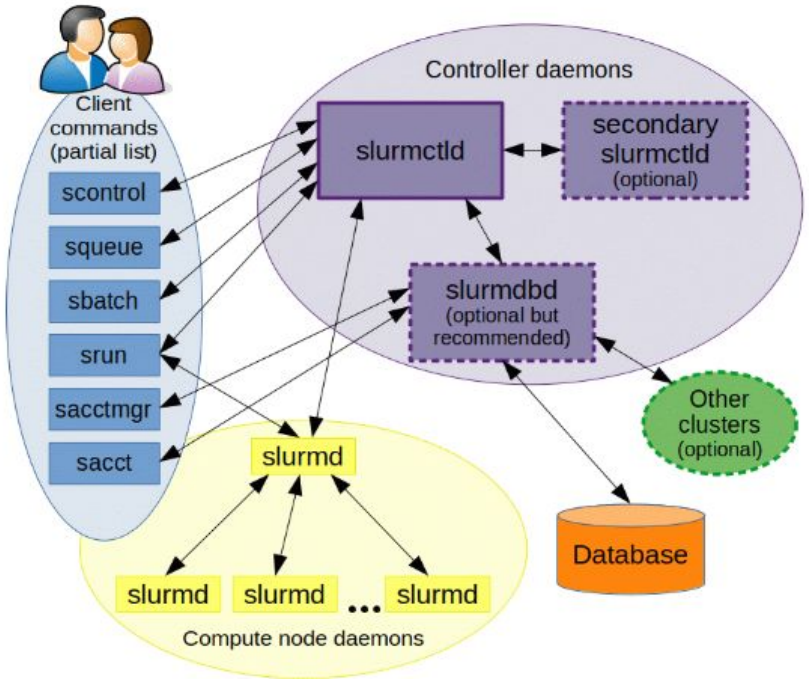
Arquitectura Maestro/esclavo



Arquitectura actual

¿Qué es Slurm?

- Simple Linux Utility for Resource Management.
- Administra recursos de clústeres y asigna trabajos de manera eficiente.



Slurm: Elementos clave

1. **Cola de trabajos**:

- Los usuarios envían trabajos (scripts) a la cola.
- SLURM decide cuándo y dónde ejecutar cada trabajo según los recursos disponibles.

2. **Componentes principales**:

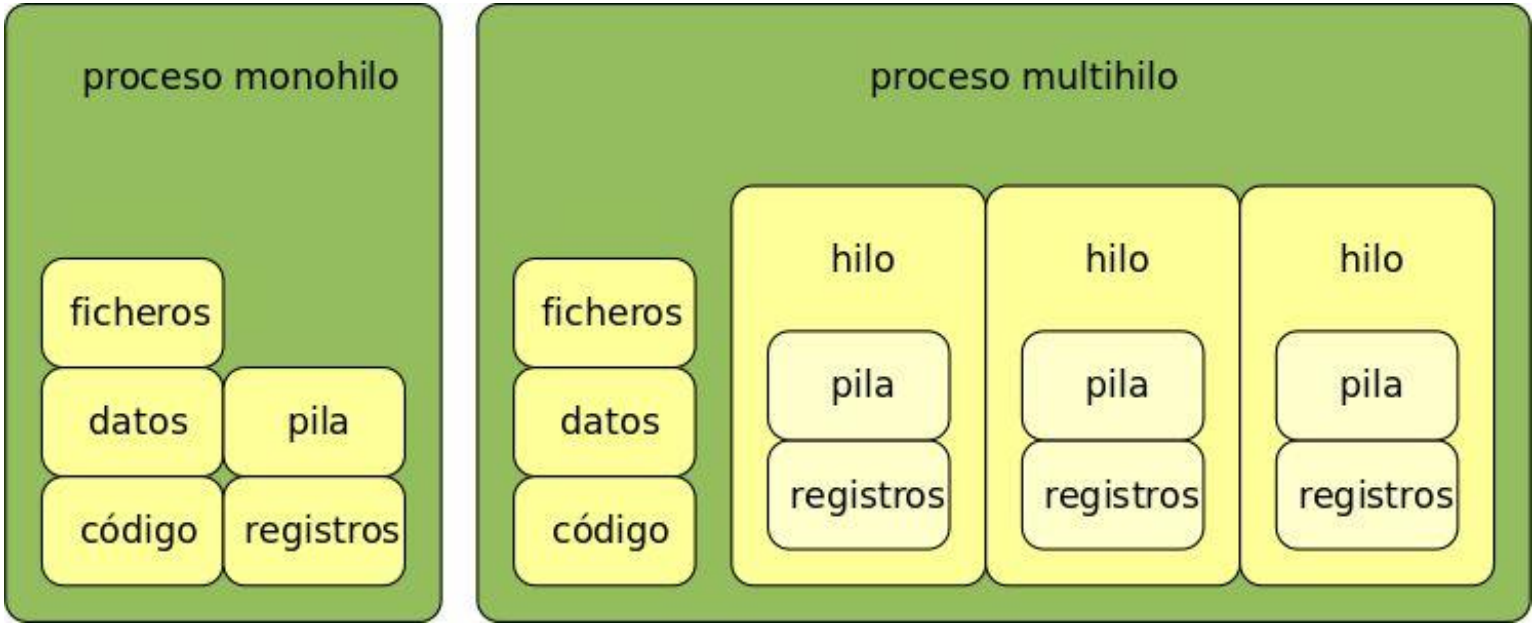
- slurmctld (SLURM Controller): Se ejecuta en el maestro.
- slurmd (SLURM Daemon): Se ejecuta en los nodos esclavos y ejecuta los trabajos asignados.
- squeue: Comando para listar trabajos en la cola.
- sinfo: Muestra información del clúster.
- scancel: ¿Qué hace **wuser36**?

Workflow básico en SLURM:

Estructura básica de un script SLURM:

```
#!/bin/bash
#SBATCH --job-name=mi_trabajo
#SBATCH --nodes=2
#SBATCH --ntasks=4
#SBATCH --time=00:10:00
#SBATCH --output=salida_%j.txt
cd $SLURM_SUBMIT_DIR
srun ./mi_programa
```

Recordando SOR



La Ley de Amdahl, pero es un chiste

Un programador paralelo está explicando a su amigo la Ley de Amdahl y le dice:

"Mientras más hilos agregas, más rápido debería ir el programa, pero aún así, ¡el secuencial siempre arruina todo!" A lo que su amigo responde:

"Ah, entonces el programa es como la fila del supermercado: por más cajas que abran, siempre hay alguien lento que lo arruina todo."

```
#!/bin/bash
#SBATCH --job-name=openmp_test          # Nombre del trabajo
#SBATCH --nodes=1                       # Número de nodos
#SBATCH --ntasks=1                      # Una sola tarea
#SBATCH --cpus-per-task=4               # 4 CPUs disponibles para la tarea
#SBATCH --time=00:10:00                 # Tiempo máximo de ejecución
#SBATCH --output=output_%j.txt          # Archivo de salida
# Exportar el número de hilos como OMP_NUM_THREADS
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
# Ejecutar el programa OpenMP
./mi_programa_openmp
```