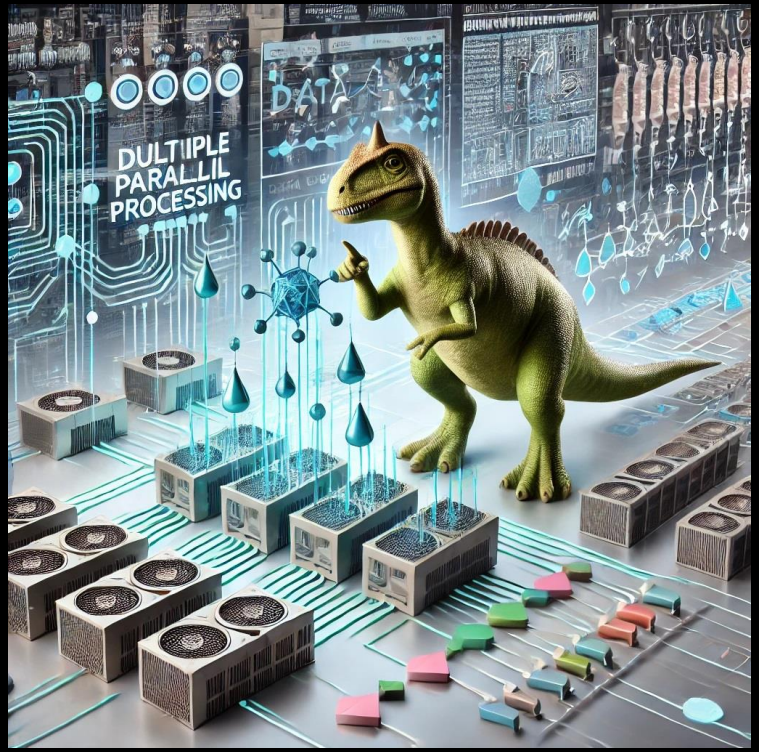
UNIVERSIDAD NACIONAL DE HURLINGHAM

Instituto de Tecnología e Ingeniería



Día 2

curso: 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.

Arquitectura Beowulf: Maestro/esclavo

Cómo funciona un clúster con arquitectura maestro/esclavos:

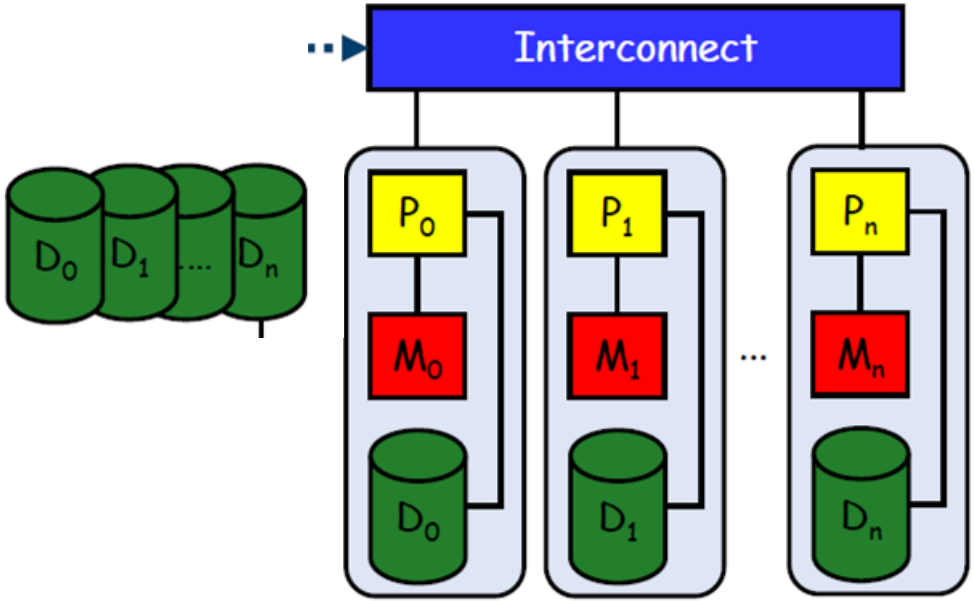
1. **Nodo maestro**:

- 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

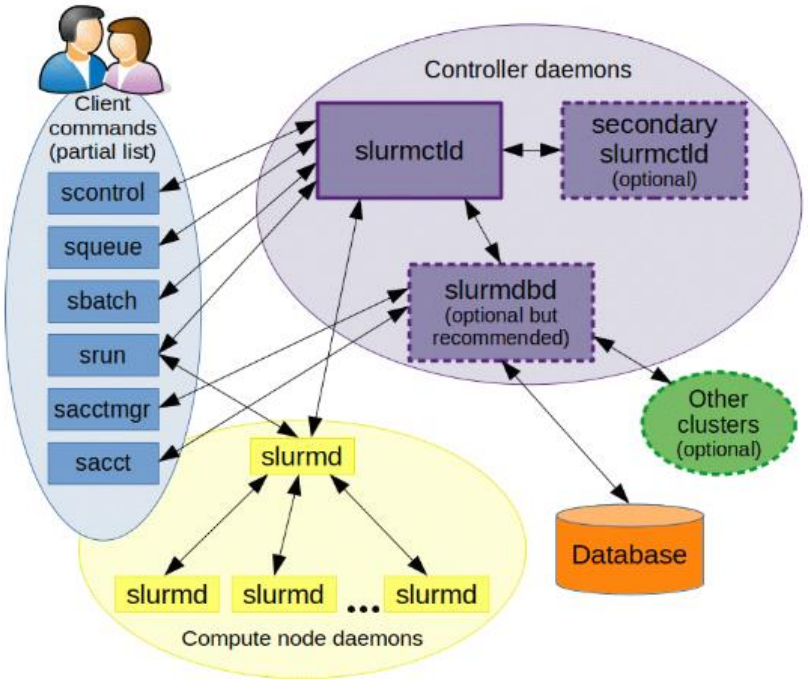


Arquitectura actual

¿Qué es Slurm?

Simple Linux Utility for Resource Management.

- Administra recursos de clústeres y asigna trabajos de manera eficiente.
- Encolar trabajos.
- Monitorear el estado del clúster.



¿Qué es Slurm?

Conceptos básicos

- **Nodo**: máquina física dentro del clúster.
- **Job**: trabajo enviado al clúster.
- **Partición**: conjunto de nodos agrupados en una cola.
- **Tarea** (task): unidad de ejecución (proceso/hilo).

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. **comandos básicos**:

- sinfo → ver estado del clúster.
- srun → ejecutar un trabajo interactivo.
- squeue → consultar la cola de trabajos.
- scancel <job_id> → cancelar un trabajo.

Workflow básico en SLURM:

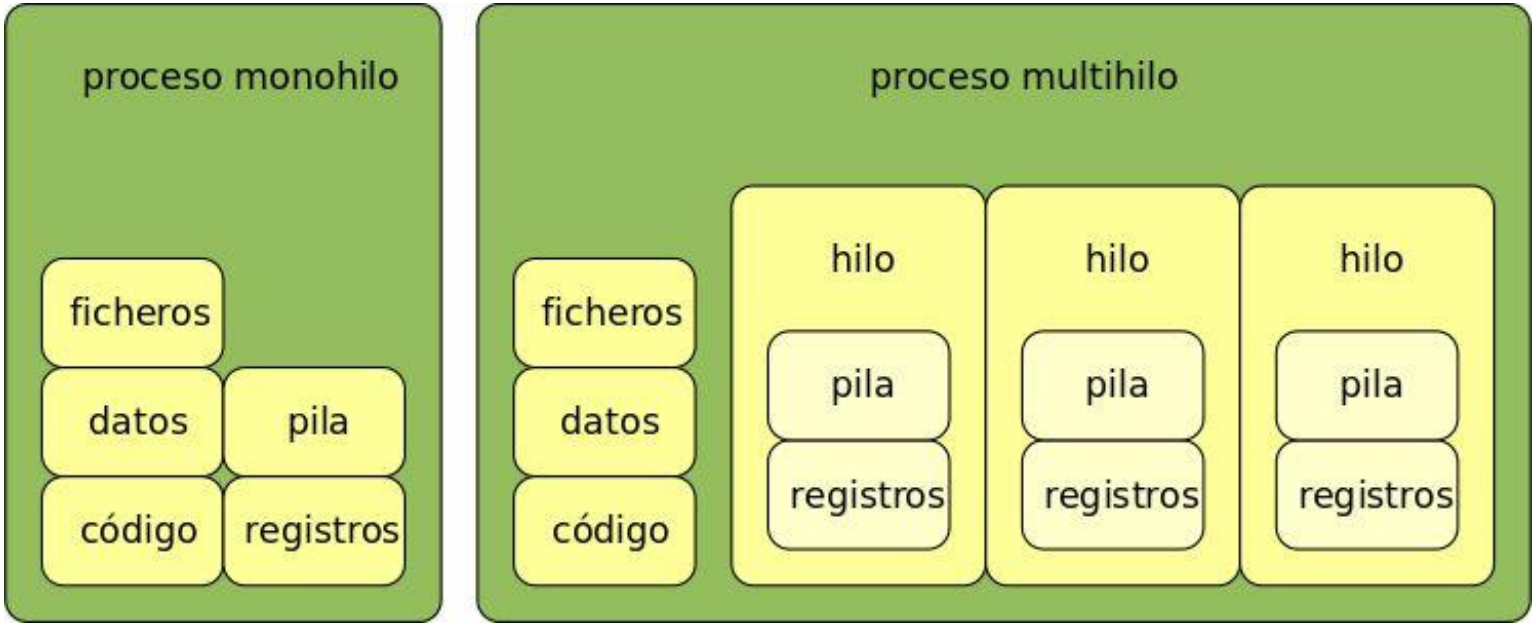
Estructura básica de un script SLURM:

```
#!/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
```

Se envía el job al cluster

- sbatch mi_job.sh
- El job queda encolado y se puede chequear con:
- squeue
- cuando el job termina, se habrá generado el archivo de salida con los resultado de la ejecución.

Recordando Sistemas operativos



```
#!/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
```

Comandos Slurm

🔍 Consulta de información

Comando	Uso	Ejemplo
`sinfo`	Ver el estado de las particiones y nodos del clúster.	`sinfo`
`squeue`	Ver trabajos en ejecución y en cola.	`squeue`
`scontrol show job <job_id>`	Ver información detallada de un job.	`scontrol show job 1234`
`scontrol show node <nodo>`	Ver información de un nodo.	`scontrol show node nodo3`
`sacct`	Historial de jobs terminados con estadísticas.	`sacct -j 1234`
`sstat -j <job_id>`	Estadísticas en tiempo real de un job.	`sstat -j 1234`

Comandos Slurm

🚀 Ejecución de trabajos

Comando	Uso	Ejemplo
`srun`	Ejecutar un trabajo interactivo.	`srun --nodes=1 --ntasks=4 hostname`
`sbatch <script>`	Enviar un script batch al clúster.	`sbatch mi_job.slurm`
`salloc`	Reservar recursos de forma interactiva.	`salloc --nodes=1 --ntasks=4 --time=00:10:00`

Comandos Slurm

⚙️ Gestión y control

Comando	Uso	Ejemplo
`scancel <job_id>`	Cancelar un job en ejecución o en cola.	`scancel 1234`
`scontrol hold <job_id>`	Congelar un job (queda en espera).	`scontrol hold 1234`
`scontrol release <job_id>`	Liberar un job que estaba en hold.	`scontrol release 1234`
`scontrol update NodeName=<nodo> State=DOWN`	Marcar un nodo como caído.	`scontrol update NodeName=nodo3 State=DOWN`
`scontrol update NodeName=<nodo> State=RESUME`	Volver a habilitar un nodo.	`scontrol update NodeName=nodo3 State=RESUME`

Comandos Slurm

📊 Reportes y prioridades

Comando	Uso	Ejemplo
`sprio`	Mostrar las prioridades de los jobs en cola.	`sprio`
`sreport`	Reportes de uso de recursos.	`sreport cluster utilization`

La Ley de Amdahl

Imaginemos que queremos cocinar una comida:

Picamos los ingredientes, esto lo pueden hacer varias personas en paralelo. Pero hervir el agua solo lo puede hacer uno, y todos tienen que esperar.

Aunque tengamos 10 personas ayudando, el tiempo de hervir el agua **no se puede acelerar**.

La Ley de Amdahl es un principio de la informática que predice la aceleración teórica máxima de un sistema al optimizar un componente, determinando que la mejora general del rendimiento está limitada por la fracción de tiempo que ese componente optimizado puede utilizarse. En esencia, establece que no se puede acelerar un sistema indefinidamente al añadir más recursos si una parte significativa de la tarea es intrínsecamente secuencial.