Introducción a Slurm

Alfonso Pardo

alfonso.pardo@ciemat.es

¿Quiénes somos?

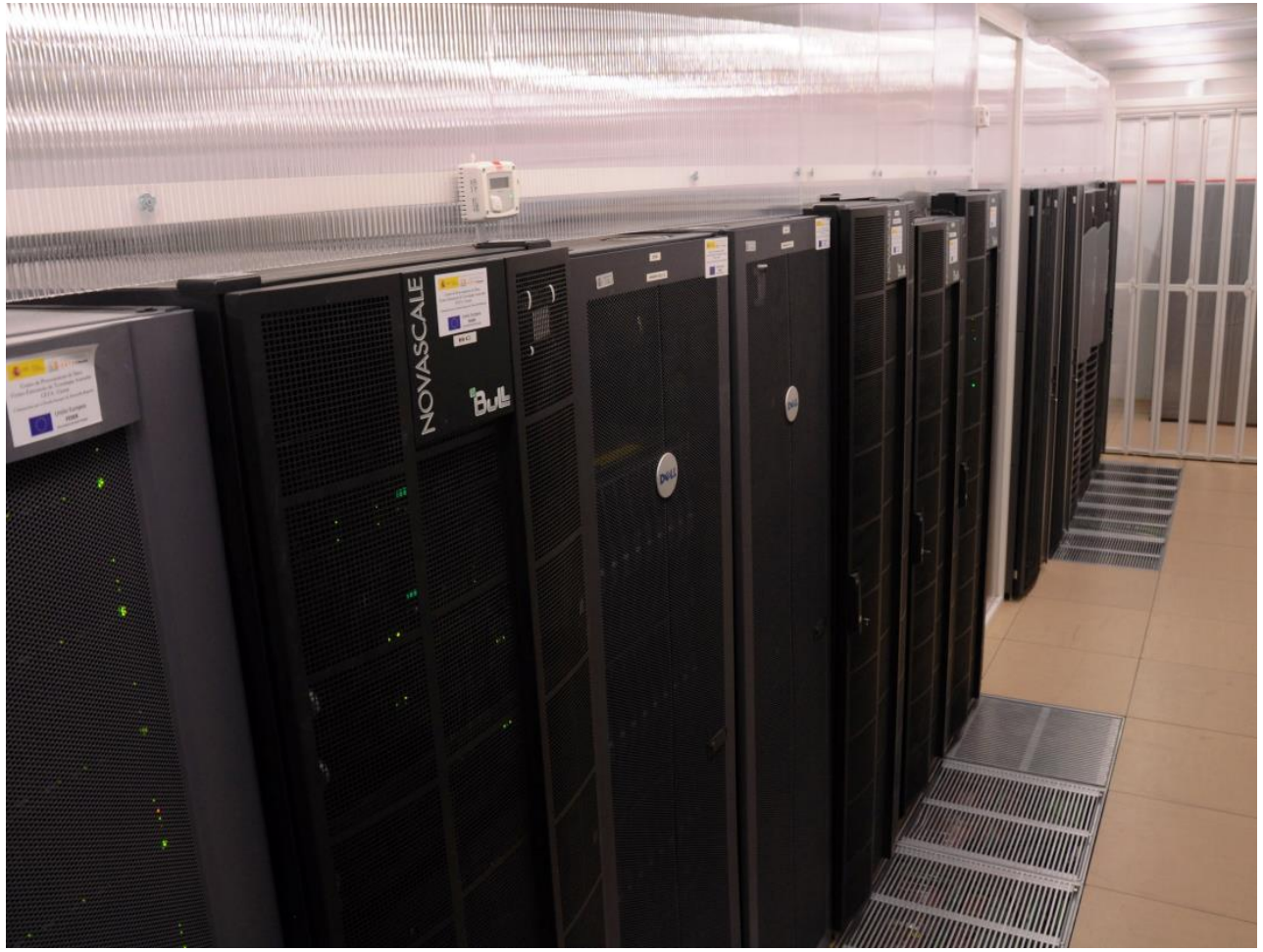
¿Quienes somos?

¿Quiénes somos?

- Centro de datos del CIEMAT (MICINN), iniciativa conjunta con la Junta de Extremadura.
- Institución pública, financiada por PGE y FEDER.
- Misión: Consolidar y diseminar eCiencia y TICs, especialmente Grid y eInfrastructuras.
- Ofrecer nuestros recursos: Grid, Cloud y HPC (GPU, clusters, ...).
- Contribuir a la expansión efectiva de eCiencia.
- Facilitar el uso de los recursos (nuevos sitios y nuevas aplicaciones).

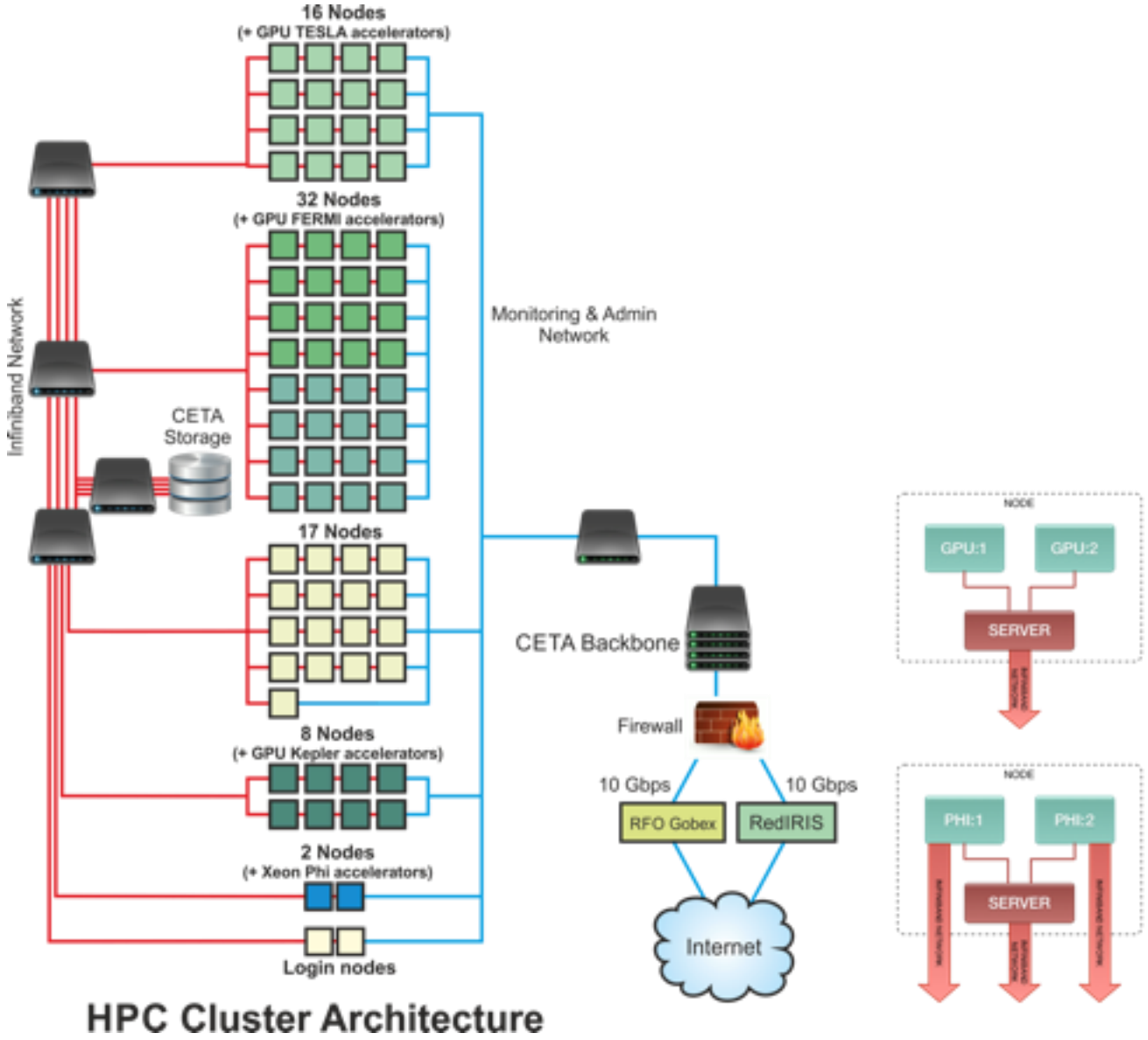
Nuestra infraestructura HPC/HTC

- **Computo:**
  - **Para grid (HTC).**
  - **Para CPU/GPGPU (HPC)**
- **Almacenamiento:**
  - **Lustre**
- **Red:**
  - **Ethernet**
  - **Infiniband**
  - **Fibre channel**
- **Free cooling**
- **Redundacia**



Supercomputador

- **HPC:**
  - **GPGPU:**
    - 17 nodos x 2 TESLA
    - 32 nodos x 2 FERMI
    - 8 nodos KEPLER
  - **PHI:**
    - 2 nodos x 2 PHI
  - **CPU:**
    - 48 nodos

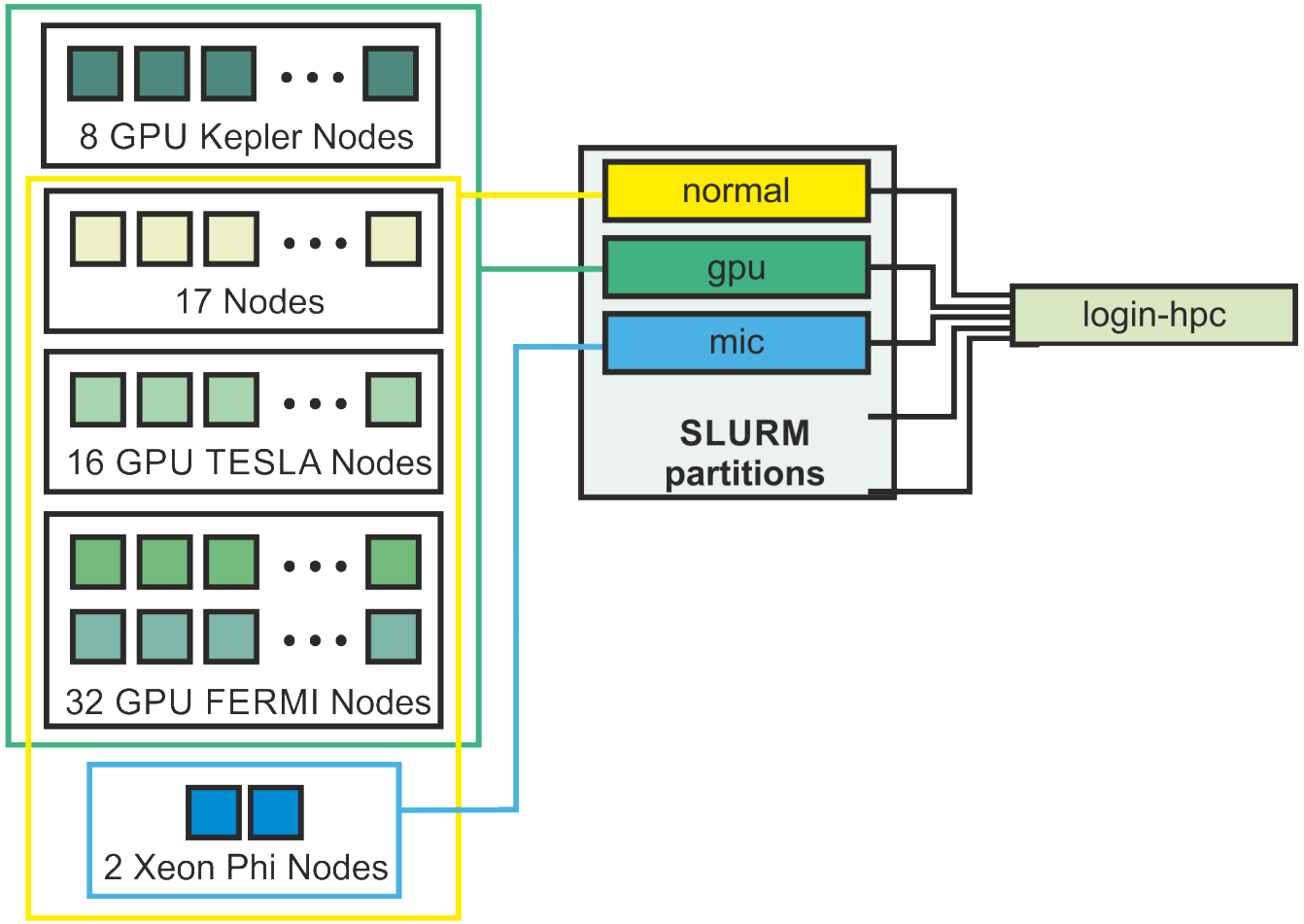


Supercomputador

- **Almacenamiento:**
  - 168TB+110TB+240TB
  - 2 librerías de cintas / HSM
  - CEPH (en progreso)

- **Network**
  - 10 Gb Ethernet
  - Conexión a RedIris Nova (10Gb/s)
  - Conexión Infiniband en el cluster (40/50Gbs)
  - Almacenamiento lustre por fibra (8Gbs) y SAS (6Gbs)

Supercomputador



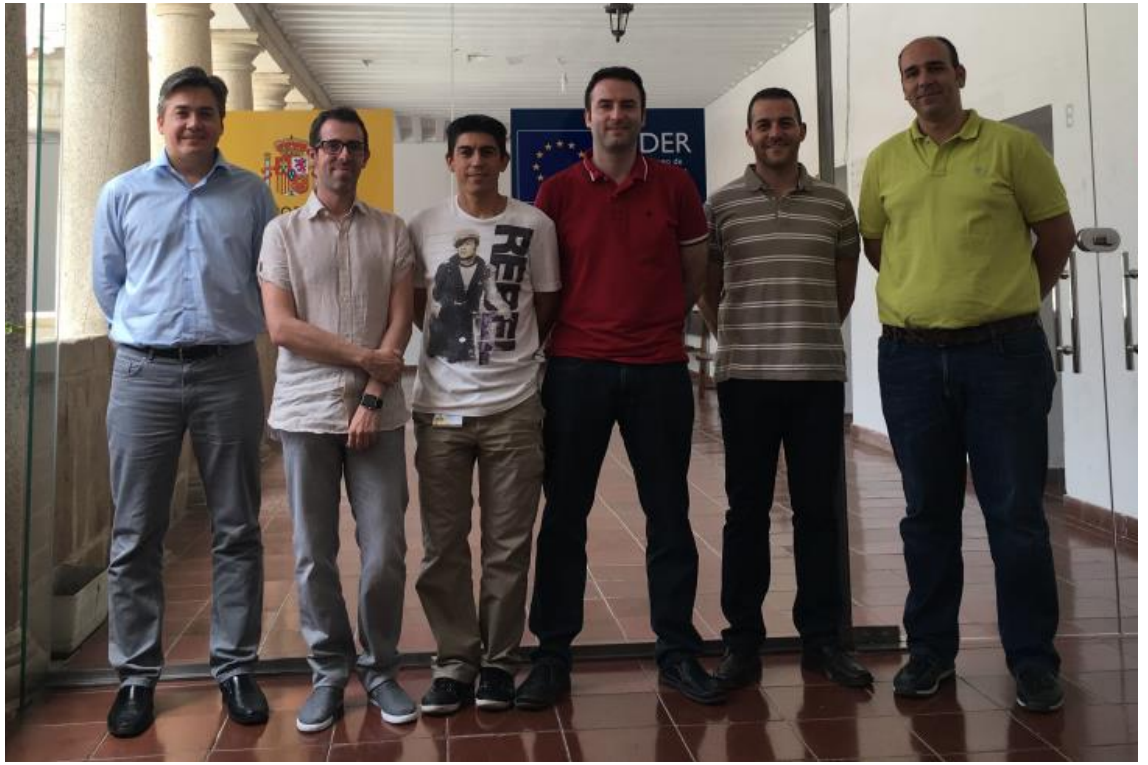
Particiones (colas):

- normal: arquitectura x86 sin aceleradores, solo CPU.
- gpu: arquitectura x86 con aceleradores: Tesla, Fermi y Kepler.
- mic: arquitectura x86 con acelerador Xeon PHI.

Supercomputador

- SO:
  - Centos 7.3
- Gestor de colas:
  - SLURM
- Conectividad:
  - Infiniband FDR+QDR (40/50Gbps en baja latencia)
  - Ethernet 1Gbs
- Almacenamiento:
  - Lustre
- Backup:
  - Bacula (Robot de cintas)

- Personal muy cualificado ☺



¿Que es Slurm?

- Gestor de colas mas utilizados (60% del Top500)
- Gratuito y open source
- Gran comunidad, grupo de usuarios
- Permite una gestión de trabajos completa:
  - Calidad de servicio
  - Colas (particiones)
  - Limites de recursos por usuario/grupo
  - Parametros de calidad de servicio
  - Reservas dinámicas
  - …

Introducción a SLURM

- Prioridades
  - Cuando hay mas trabajos que nodos hay que establecer un método justo que determine cual es el próximo trabajo en ser despachado.
  - SLURM realiza cálculos periódicamente basándose en un algoritmo de gestión de prioridades en base a barios parámetros configurables (por el administrador):
    - Tamaño del trabajo (numero de nodos/cores).
    - Duración de los Jobs (se especifica a la hora de encolarlo).
    - Edad del trabajo en la cola
    - Numero de trabajos en cola

Introducción a SLURM

- ¿Como acceder?
  - Acceso por SSH al nodo de login.
  - Desde el nodo de login se ejecutan todos los comandos.

- Almacenamiento
  - El espacio de trabajo esa compartido a todos los nodos en un sistema de ficheros paralelo y distribuido

- Librerias
  - Cada usuario necesita distintas librerías
  - Cada librería puede tener distintas versiones o “sabores”.
  - Cada librería puede tener distintas dependencias.
  - Cada librería puede tener distintas rutas de instalación.
  - Solucion → ”environment modules”
    - Permite gestionar diferentes versiones o sabores de un mismo compilador o librería en una misma instalación del sistema operativo.

Introducción a SLURM

- Uso de módulos o librerías:
  - Como carga librerías mediante “módulos”
  - Permite cargar una librería y sus dependencias:
    - Carga openmpi → gcc
  - Permite varias versiones del mismo software.
  - No genera conflictos entre versiones.
  - El software está centralizado.
  - Se cargan las variables de entorno del módulo.
  - Se puede especificar versión para un mismo modulo. En el caso que se omita se carga la mas actual o la que este etiquetada por defecto.
  - Los módulos pueden tener etiquetas si son especiales o tienen alguna compilación especial:
    - (g) → compilada para GPU

Uso de librerías

- Para cargar un módulo:
    - module avail → muestra todos los nodos disponibles
    - module load <nombre_modulo> → Carga un modulo
    - module list → lista todos los módulos cargados
    - module unload → quita un módulo previamente cargado
    - module purge → elimina todos los modulos cargados

- Ejemplos:
    - module load intel → Carga las librerías mas actual de intel y variables de entorno.
    - module load gcc/6.2.0 → Carga la librería gcc version 6.2 y sus variables de entorno.

Comandos y envío de trabajos

- Comandos útiles:
  - sinfo -> obtiene información de estado del supercomputador.
  - squeue -> obtiene estado de las colas/particiones.
  - srun –> ejecuta un trabajo simple.
  - sbatch -> ejecuta un script.
  - scancel -> cancela un trabajo lanzado.
  - sacct -> obtiene estadísticas de uso del supercomputador.
- Parámetros:
  - -N X → uso de “X” nodos
  - -n X → lanza “X” instancias del trabajos
  - -p X → usa la partición “X”
  - --gres=X:Y → reserva para usar el Y recursos X
  - -t D-HH:MM:SS –> Tiempo máximo de ejecución de un trabajo en días, horas, minutos y segundos.

Comandos y envío de trabajos

- Envío de trabajos:
  - Uso básico:
    - sbatch <parametros> <script>
  - El script contiene:
    - Variables de entorno
    - Llamada al programa
  - Para este curso se han reservado nodos para no tener que esperar a otros usuario:
    - Parametro: --reservation=curso_ciemat00_13

- Obtención de resultados:
  - Una vez ejecutado el trabajo, si ha finalizado con éxito obtendremos dos ficheros:
    - slurm-XXXXXX.out → Contiene los resultados de la ejecución del trabajos lanzado.
    - slurm-XXXXXX.err → Contiene los errores de la ejecución del trabajo lanzado.

Comandos y envío de trabajos

- Parámetros mas usuales para envió de trabajos:

Parámetro	Abreviatura	Descripción
--partition=	-p	Particion a la que enviaremos el trabajo
--node=	-N	Numero de nodos requeridos
--ntask=	-n	Numero de tareas a enviar
--gres=	--gres	Uso de recurso: gpu,mic
--time=	-t	Tiempo máximo de la ejecución
--mem=		Memoria en megabits que usara nuestro trabajo
--job-name=	-J	Nombre de nuestro trabajo
--error=	-e	Fichero de salida de error
--output=	-o	Fichero de salida de resultados

Mas información:

- https://slurm.schedmd.com/sbatch.html
- man sbatch

Comandos y envío de trabajos

- Estado del cluster:
  - Uso básico:
    - sinfo

```
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
normal*      up   infinite     15   resv bc[10-14],bd[16-25]
normal*      up   infinite     62  alloc bc[03-09,15-18],bd[01-14,26-28],be[01-02],dap[101-104,201-204,301-304,401-404,501-504,601-604,701-704,801-804]
normal*      up   infinite      5   idle bd[15,29-32]
mic          up   infinite      2  alloc be[01-02]
gpu          up   infinite     15   resv bc[10-14],bd[16-25]
gpu          up   infinite      2    mix da[09,11]
gpu          up   infinite     28  alloc bc[03-09,15-18],bd[01-14,26-28]
gpu          up   infinite     11   idle bd[15,29-32],da[10,12-16]
```

  - Uso detallado:
    - avail nodes

```
NODELIST                                NODES  CPUS   MEMORY AVAIL_FEATURES                                                    GRES
bd[17-32]                               16     12     22000  Infiniband_QDR,CPU_Westmere,E5649,2.53GHz,24GB_RAM,gpu_fermi_M2075        gpu:fermi2070:2
bc[03-18]                               16     8      22000  Infiniband_QDR,CPU_Nehalem,E5520,2.26GHz,24GB_RAM,gpu_tesla_C1060        gpu:tesla:2
bd[01-16]                               16     8      22000  Infiniband_QDR,CPU_Nehalem,E5520,2.26GHz,24GB_RAM,gpu_tesla_C2050        gpu:fermi2050:2
be[01-02]                               2      12     30000  Infiniband_QDR,CPU_Sandy,E5-2620,2.00GHz,32GB_RAM,phi                   mic:2,rapl:1
dap[101-104,201-204,301-304,401-404,501-504 32     12     30000  Infiniband_QDR,CPU_Sandy,E5-2620,2.00GHz,32GB_RAM                       rapl:1
da[09-16]                               8      24     62000  Infiniband_FDR,CPU_Haswell,E5-2680v3,2.50GHz,64GB_RAM,gpu_kepler_K80     gpu:kepler:2,rapl:1
```

Comandos y envío de trabajos

- Estado de las colas:
  - Uso básico:
    - squeue
  - Solo mis trabajos
    - squeue -u nombre_usuario
  - Uso detallado
    - squeue -l

```
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             37806    normal barrido.   apardo PD       0:00      1 (Resources)
             37807    normal barrido.   apardo PD       0:00      1 (Priority)
             37808    normal barrido.   apardo PD       0:00      1 (Priority)
             37809    normal barrido.   apardo PD       0:00      1 (Priority)
             37810    normal barrido.   apardo PD       0:00      1 (Priority)
             37801    normal barrido.   apardo  R       0:02      1 bd31
             37802    normal barrido.   apardo  R       0:02      1 bd29
             37803    normal barrido.   apardo  R       0:02      1 bd30
             37804    normal barrido.   apardo  R       0:02      1 bd32
             37805    normal barrido.   apardo  R       0:02      1 bd15
```

Comandos y envío de trabajos

- Estadísticas de uso:
  - Uso básico:
    - sacct
  - Si no especificas tiempo solo muestra el día actual.
  - Para crear un rango de tiempo de estatificas usar los parámetros “-S” y “-E” para determinar el inicio (S) y fin (E).
    - Emplearemos la notación: YYY-MM-DD
    - sacct –S 2017-01-01 –E 2017-10-09

  - O ver el estado concreto de un trabajo con “-j numero_trabajo”

```
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
24445        mpicuda.sh        gpu     cursos          1  COMPLETED      0:0
24445.batch       batch                cursos          1  COMPLETED      0:0
24445.0         mpicuda                cursos          1  COMPLETED      0:0
26092        deviceQue+        gpu     cursos          1  COMPLETED      0:0
26092.batch       batch                cursos          1  COMPLETED      0:0
```

Comandos y envío de trabajos

Code	State	Description
CA	CANCELLED	Trabajo cancelado por el usuario o el administrador.
CD	COMPLETED	El trabajo ha finalizado en todos los nodos
CF	CONFIGURING	El trabajo ha ocupado los recursos y esta esperando que estos terminen de iniciarse.
CG	COMPLETING	El trabajo finalizado, se esta esperando por algún nodo.
F	FAILED	El trabajo ha finalizado con una condición distinta de 0 (éxito)
NF	NODE_FAIL	El trabajo ha finalizado porque algun nodo ha fallado.
PD	PENDING	El trabajo esa pendiente de los recursos.
R	RUNNING	Trabajo en ejecución.
S	SUSPENDED	El trabajo ha encontrado recursos, pero la ejecucion no se ha completado.
TO	TIMEOUT	El trabajo ha alcanza el tiempo limite..

Comandos y envío de trabajos

- Estado de las particiones:
  - Uso:
    - avail partitions

```
PARTITION AVAIL  TIMELIMIT   NODES(A/I/O/T)  NODELIST
normal*      up   infinite      62/20/0/82  bc[03-18],bd[01-32],be[01-02],dap[101-104,201-204,301-304,401-404,501-504,601-604,701-704,801-804]
mic          up   infinite         2/0/0/2  be[01-02]
gpu          up   infinite      30/26/0/56  bc[03-18],bd[01-32],da[09-16]
```

Prioridades

- Determina en que posición de la cola se situara un trabajo para ser atendido.
- Factores que determinan prioridad de tarea
  - Edad - la cantidad de tiempo que el trabajo ha estado esperando en la cola
  - Tamaño de la tarea - número de nodos solicitados por el trabajo
  - Partición - prioridad para una partición determinada
  - Mientras más tareas se ejecuten en el cluster, menor será la prioridad.
  - Mientras menos tareas se ejecuten en el clúster, más alta es la prioridad.

Comandos y envío de trabajos

- Ejemplo:
  - Script (hostname.sh):

```
#!/bin/sh
hostname
```

  - Ejecucion:
    - sbatch -n 2 --reservation=curso_ciemat00_13 -t 0-00:01:00 hostname.sh
  - Resultado?

Comandos y envío de trabajos

- Ejemplo:
  - Script (holamundo.sh):

    ```
    #!/bin/bash
    ./holamundo_app.sh
    ```

  - Script (holamundo_app.sh):

    ```
    #!/bin/bash
    yo=$(hostname)
    echo “Hola mundo desde $yo”
    ```

  - Ejecución:
    - chmod +x holamundo_app.sh
    - sbatch -n 2 --reservation=curso_ciemat00_13 -t 0-00:01:00 ./holamundo.sh
  - Resultado?

Comandos y envío de trabajos

- Slurm permite enviar y administrar miles de trabajos similares de una sola vez.
  - Todos los trabajos deben tener las mismas condiciones iniciales.
  - Provee de variables para controlar la ejecución de los jobs.

```
#!/bin/bash
#SBATCH --job-name=sleep-test
#SBATCH --partition=normal
#SBATCH -n 1
#SBATCH --output=st_%j.out
#SBATCH --error=st_%j.err
#SBATCH --array=1-10
#SBATCH -t 0-00:05:00

./ejecutable entrada_$SLURM_ARRAY_TASK_ID
```

Sesión practica