Notas de campo desde el frente de soporte para Slurm

Alejandro Sánchez
SchedMD

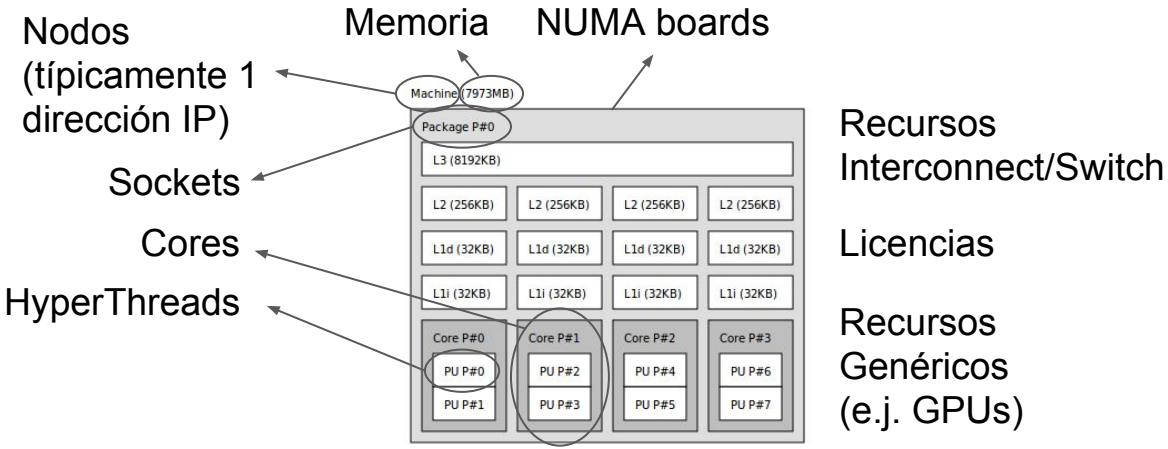
HPC AdminTech 2018

Breve introducción a Slurm

- Gestor de recursos y planificador
- Código abierto (GPL v2, disponible en GitHub)
- Altamente configurable y escalable
- Amigable para administradores de sistemas
- Seguro, a prueba de fallos y portable
- Comunidad global activa
- Usado en muchos de los sistemas más grandes del mundo

Propósito de un gestor de recursos

- Asignar recursos en un clúster

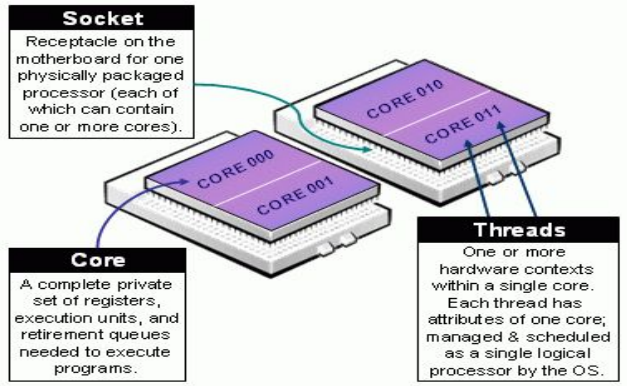


Recursos
Interconnect/Switch

Licencias

Recursos
Genéricos
(e.j. GPUs)

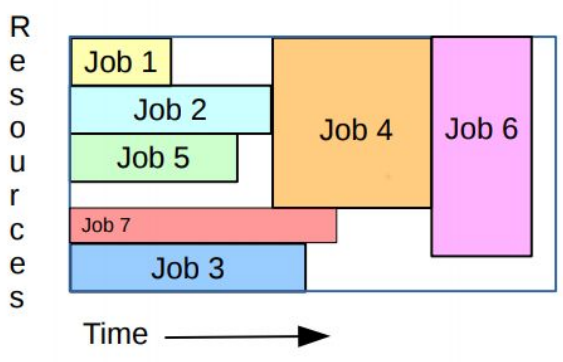
Esto puede requerir un amplio conocimiento sobre el hardware y el software del sistema (hwloc, autoconf, etc.).



- Lanzar y gestionar jobs

Propósito de un planificador de jobs

- Más jobs que recursos disponibles
  - Priorizar jobs en base a políticas
  - Gestionar tiempo sobre los recursos
  - Imponer límites configurables
  - Coordinarse con el gestor de recursos



Notas de campo

- Notas aleatorias, observaciones y configuraciones recomendadas (opiniones propias)
  - Aunque si se envía un bug, probablemente se obtengan sugerencias repetidas o muy similares
  - No son requisitos estrictos, puede haber diferentes aproximaciones para atacar un problema
  - Slurm se puede configurar de muchas maneras distintas para encajar en diferentes entornos, y no todas estas recomendaciones tienen por qué ser válidas universalmente

Notas de campo

- No duden en preguntar a lo largo de la presentación
  - Aunque podemos diferir discusiones más extensas para más tarde en favor del tiempo

Rotación de logs

- Buena práctica y es una petición común
  - Aunque normalmente mal configurada
- Si se usa *logrotate*, asegurarse de enviar un signal adecuado a los daemons correctos
- slurmdbd necesita un SIGHUP cuando slurmdbd.log se rota, muchos sites tenían esto mal configurado y perdieron logs
- Todos los daemons hacen un restart con SIGHUP
  - Asegurarse de que los ficheros de config están en buen estado

Rotación de logs

- Desde 17.11 se soporta SIGUSR2 en {slurmctld, slurmdbd, slurmd} para sólo re-abrir los logs.
  - Evita reconfigs, es mejor opción.
- Otro error: al menos un cluster manager ha enviado *logrotate* scripts que ejecutan *scontrol reconfigure* cuando rotan logs locales
  - Causa O(N) reconfiguraciones en el cluster en rápida sucesión, y como consecuencia puede haber problemas temporales de rendimiento

Límites fijos para accounts

- Opción de QOS Flags=NoDecay
  - Permite establecer una asignación fija de recursos para un grupo.
  - Una “bank account”, por decirlo de otro modo.
  - GrpTRESMins, GrpWall, UsageRaw no decaerán, incluso usando Priority{DecayHalfLife,UsageResetPeriod}. Como una cuota fija.
  - Jobs que corran contra la QOS eventualmente alcanzarán el límite.
  - Cambiar RawUsage=0 para resetear, o cambiar los límites.
  - Gracias a Josh Samuelson (U. Nebraska Lincoln) por el patch.

Mantener una configuración ordenada

- Al definir tanto Nodos como Particiones, se dispone de la palabra reservada DEFAULT.
  - Usada para definir opciones por defecto a líneas sucesivas.
  - Por ello no se puede tener una partición llamada “default”.
- La siguiente línea DEFAULT sobreescribirá valores previos.
  - Pero sin resetearlos todos. E.j., si la primera define Features=foo,bar, y la segunda Weight=100, las líneas después de la segunda tendrán ambas opciones por defecto.
- Expresiones de rangos: rack[0-63,70-80]_blade[0-41]

Mantener una configuración ordenada

- Antes

```
NodeName=node0 SocketsPerBoard=2 CoresPerSocket=4 ThreadsPerCore=2 RealMemory=32768 GRES=gpu:k80:4 Weight=10
NodeName=node1 SocketsPerBoard=2 CoresPerSocket=6 ThreadsPerCore=2 RealMemory=32768 Weight=10
NodeName=node2 SocketsPerBoard=2 CoresPerSocket=6 ThreadsPerCore=2 RealMemory=32768 Weight=10
NodeName=node3 SocketsPerBoard=2 CoresPerSocket=6 ThreadsPerCore=2 RealMemory=32768 Weight=10
NodeName=node4 SocketsPerBoard=2 CoresPerSocket=6 ThreadsPerCore=2 RealMemory=32768 Weight=10
NodeName=node5 SocketsPerBoard=2 CoresPerSocket=6 ThreadsPerCore=2 RealMemory=32768 Weight=10
NodeName=node6 SocketsPerBoard=2 CoresPerSocket=8 ThreadsPerCore=2 RealMemory=65536 Weight=2
NodeName=node7 SocketsPerBoard=2 CoresPerSocket=8 ThreadsPerCore=2 RealMemory=65536 Weight=2
NodeName=node8 SocketsPerBoard=2 CoresPerSocket=8 ThreadsPerCore=2 RealMemory=65536 Weight=2
NodeName=node9 SocketsPerBoard=2 CoresPerSocket=8 ThreadsPerCore=2 RealMemory=65536 Weight=2
NodeName=node10 SocketsPerBoard=2 CoresPerSocket=8 ThreadsPerCore=2 RealMemory=65536 Weight=2
NodeName=node11 SocketsPerBoard=2 CoresPerSocket=8 ThreadsPerCore=2 RealMemory=65536 Weight=2
```

Mantener una configuración ordenada

- Después

```
NodeName=DEFAULT SocketsPerBoard=2 ThreadsPerCore=2 RealMemory=32768 Weight=10
NodeName=node0 CoresPerSocket=4 GRES=gpu:k80:4
NodeName=node[1-5] CoresPerSocket=6
NodeName=node[6-11] CoresPerSocket=8 RealMemory=65536 Weight=2
```

Asignando memoria

- Si se usa CR_{Core,CPU,Socket}_Memory, asegurarse de definir DefMemPerCPU ó DefMemPerNode.
  - Elegir una u otra. DefMemPerCPU normalmente preferida.
  - E.j. DefMemPerCPU casi igual que (Memoria / CPUs).
    - Mejor redondear un poquito abajo, al GB más cercano.
    - Ayuda a llenar mejor los nodos.
  - Si no se define, el primer job que no especifique --mem[-per-cpu] se le asignará el nodo entero.
    - Dejando 0 MB libres para otros jobs.

Configuración Nodos y Particiones

- DefMemPer[CPU|Node] es uno de los motivos por los que se desaconseja definir particiones con nodos heterogéneos, especialmente con diferente tamaño de memoria.
- Es preferible minimizar el número de particiones, quizá tantas como tipos de nodos con hardware distinto.
- Si se decide englobar nodos distintos en una misma partición, se puede definir un Weight más bajo a nodos con hardware más barato, de modo que tengan preferencia de asignación.

Configuración Nodos

- Los valores de las características de los nodos deben ser menor o igual que el hardware real disponible.
- Si un nodo registra más sockets/threads/cores/memory no pasa nada, aunque Slurm asignará recursos en base a los valores configurados.
- Se recomienda a los sites redondear hacia el GB bajo más próximo.

Configuración Nodos - Continuación

- Redondear hacia abajo RealMemory consigue:
  - Evitar marcar el nodo a DOWN si la BIOS reporta un tamaño de memoria un poco menor.
    - Puede ocurrir con cambios firmware, reemplazo de DIMM, o incluso kernel upgrade. Si el nodo reporta 4MB menos que el valor configurado, se mantendrá offline.
    - Al usar CR_*_Memory da un respiro al SO.
      - Si Slurm no puede asignar ese espacio, nunca será distribuido entre los jobs. Ayuda a evitar estados de OOM.

Configuración Nodos - Continuación

- Otra alternativa:
  - La opción MemSpecLimit en la definición de nodos se puede usar para apartar algo de memoria para slurmd/slurmstepd, y los jobs no podrán asignar esa porción.
    - La parte asignable a jobs será RealMemory - MemSpecLimit.
  - La memoria de los daemons slurmd/slurmstepd se limitará con cgroups a MemSpecLimit, cerciorarse por tanto de no definir un valor muy bajo.

Configuración Nodos - Continuación

- No se pueden añadir o quitar nodos “on the fly”.
- Slurm tiene estructuras de datos internas (bitmaps) relacionadas con la planificación y la comunicación que están diseñadas alrededor de un tamaño fijo de bits por razones de rendimiento, y no se puede cambiar sin hacer un restart.
- … tampoco se pueden “renombrar” particiones, esto cancelaría todos los jobs PD o R enviados contra la “antigua”.

Restringir memoria correctamente

1. Existen 2 mecanismos, se recomienda deshabilitar el primero y habilitar el segundo.
   - JobAcctGather (funciona por encuesta)
   - TaskPlugin (funciona por interrupción)

slurm.conf:

```
MemLimitEnforce=no
JobAcctGatherParams=NoOverMemoryKill
TaskPlugin=task/cgroup (aunque mejor 'task/cgroup,task/affinity'
apilados)
```

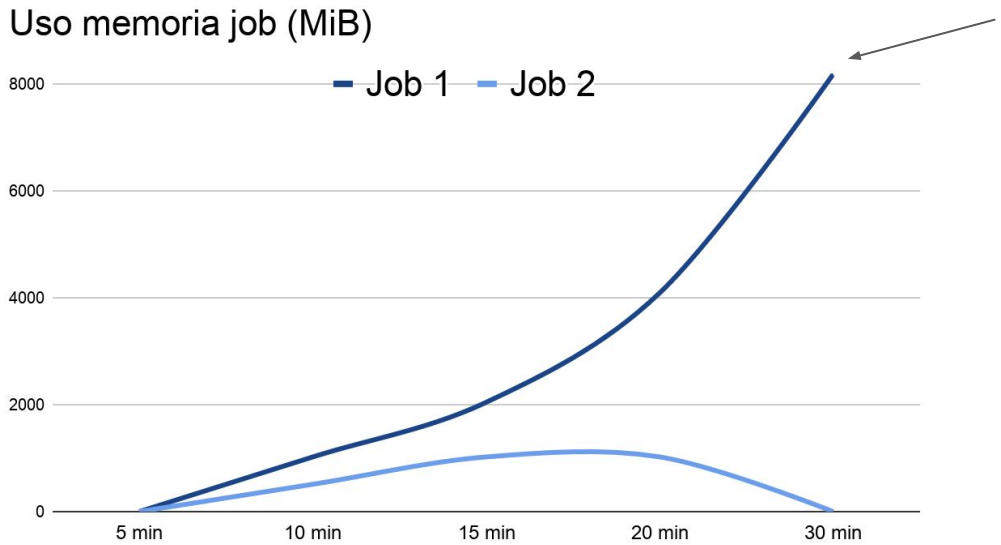
cgroup.conf:

```
ConstrainCores=yes
ConstrainRAMSpace=yes (y leer AllowedRAMSpace)
ConstrainSwapSpace=yes (y leer AllowedSwapSpace y MemorySwappiness)
TaskAffinity=no
```

Notas extra:

- Limpieza automática de jerarquías de cgroup desde 17.02.3 (no se necesita ReleaseAgent)
- MemorySwappiness desde 17.11.2.
- Detección mejorada de eventos oom-kill desde 17.11.3 (bug 3820) →

Restringir memoria correctamente



- Uso > límite → memory.failcnt++
- Antes de 17.11.3, esto era suficiente para marcar el job como OutOfMemory.
- Problema: el Kernel puede reclamar páginas y el proceso acabar con éxito, pero el job era marcado OutOfMemory.
- Solución1: cgroups-v2 incorpora contadores de eventos:
  - low, high, max, oom, oom_kill
  - Pero muchos sites no tienen kernels tan recientes…
- Solución2: cgroups-v1 + eventfd() sobre memory.oom_control para detectar eventos oom_kill, en vivo.

Partition Priority

- Cuánta gente usa la opción Priority en una partición en conjunto con PriorityWeightPartition para ajustar las prioridades de los jobs de los usuarios?

Partition Priority

- Sí, es una pregunta con trampa. Quizá ajustar esta opción no tenga las consecuencias esperadas.
- El planificador ordena los jobs en una única cola según el siguiente orden:
  1. Preemption
  2. Advanced reservations
  3. Partition PriorityTier
  4. Job Priority
  5. JobId

Job Priority =
PriorityWeightAge * AgeFactor +
PriorityWeightFairShare * FairShareFactor +
PriorityWeightJobSize * JobSizeFactor +
PriorityWeightPartition * PriorityJobFactor +
PriorityWeightQOS * QOSFactor +
PriorityWeightTRES * TRESFactor.

Partition Priority

- Por tanto el efecto que tiene la opción PriorityJobFactor en los jobs de esa partición es la contribución a uno de los términos de los diferentes factores que componen JobPriority (priority/multifactor plugin)
- Pero Job Priority sólo importa *cuando se compara con jobs en la misma Tier,* puesto que jobs de PriorityTier superiores siempre tendrán preferencia.

Partition Priority

- Desde la versión 16.05 se separa la opción Priority en una Partition en dos partes:
  - PriorityTier - sólo define la preemption tier, no afecta a JobPriority.
  - PriorityJobFactor - normalizado, se usa con PriorityWeightPartition para cambiar valores en Job Priority. No define ninguna Tier o capa.
- Por retrocompatibilidad, si se define Priority, el valor se aplica a ambas opciones PriorityTier y PriorityJobFactor.
  - Aunque PriorityJobFactor puede tener poco impacto.

Problemas afinando Backfill

- El más común: valor muy corto de bf_window.
- Debería ser al menos el máximo TimeLimit permitido por límites, en lo razonable. Máx. de ~2 semanas (preferencia).
  - Valores mayores implican mayor consumo de memoria, y reducción del rendimiento de backfill.
- Valores muy bajos dejarían en PD eternamente jobs grandes.
- Si se incrementa el valor de bf_window, es recomendable incrementar bf_resolution en proporción.

Problemas afinando Backfill

- Otras opciones recomendadas / a considerar:
  - bf_continue
  - bf_min_[prio|age]_reserve (optimizar utilización del sistema)
- Y por supuesto… forzar que todos los jobs sean enviados con un TimeLimit, sino backfill no funcionará.
  - Definir un DefaultTime acomodará a los usuarios y acabarán todos con el mismo TimeLimit. Evitar esto.

Sobre rendimiento, tiempo y similares

- Las llamadas a sistema time() y gettimeofday() han de ser rápidas para un buen rendimiento de slurmctld.
  - Éste las invoca con frecuencia.
- Algunas plataformas VM no manejan esto bien, ya que deshabilitan el vDSO de Linux que normalmente está en uso.
  - vDSO evita un cambio de contexto al Kernel (y vuelta) y hace de time() una referencia directa a memoria. slurmctld asume vDSO disponible.
  - Xen especialmente culpable de esto.

Sobre rendimiento, tiempo y similares

- Hardware:
  - Se prefieren pocos cores pero rápidos en el nodo de slurmctld.
  - El acceso a StateSaveLocation ha de ser rápido.
    - IOPS en este fs puede ser cuello de botella para el throughput de los jobs.
      - Almenos 1 directorio y 2 ficheros creados por job.
      - + las llamadas a unlink().
    - Usar JobArrays en vez de jobs individuales puede mejorar esto de manera significativa, ya que sólo un job script y un fichero de environment se guarda por array entero.

Sobre rendimiento, tiempo y similares

- slurmctld crea muchos threads.
  - 10 - 100’s de threads en uso en cualquier momento.
- Desafortunadamente, slurmctld no es muy concurrente.
  - Locks de structs internas limitan muchas veces el #threads concurrentes a… 1 thread.
  - Estamos trabajando en esto y hemos hecho progresos últimamente.
- Hardware:
  - Dual 16-core Xeons es algo exagerado para el nodo de slurmctld, 6-cores con mayor frecuencia (y más baratos!) ya rinden bien.

Job Submit Plugins

- Permiten añadir lógica de negocio arbitraria para filtrar/denegar/modificar jobs enviados/modificados.
- Ejecutados en el proceso slurmctld, y tienen acceso a las estructuras de datos internas.
  - Cuidado que también pueden corromper cualquier cosa si están mal diseñados.
- Un job_submit.lua plugin opcional usa un lenguaje de scripting interpretado y “seguro”, en vez de implementar en C.

Job Submit Plugins

- Lua (o un nuevo plugin C) permiten hacer casi de todo.
- Potencialmente incluso llamar a APIs externas.
- Encadenar workflows propios de cada site.
- Imponer lógica arbitraria
  - Por ejemplo forzar a los usuarios a especificar un valor de --time, para backfill, como se ha sugerido antes.

Job Submit Plugins

- Advertencias
  - Antes había mencionado que slurmctld es “highly threaded”, pero no “highly concurrent”, cierto?
  - Por razones de seguridad, el job_submit plugin opera sosteniendo write locks en 2 estructuras de datos internas principales.
  - Así que otros threads dependen de estos locks para ejecutarse de manera concurrente.
  - Por tanto, evitar hacer los scripts muy lentos, o pagar el precio en productividad del sistema y capacidad de respuesta.

Slurmdbd se cuelga

- La fuente más común de problemas con slurmdbd viene de peticiones sacct demasiado agresivas.
  - Causadas por error, o mal scripting.
- Dependiendo del sistema, ‘sacct -S 2010-01-01’ puede devolver millones de filas.
  - Puede vaciar la memoria disponible del nodo de slurmdbd, o simplemente tardar un tiempo no razonable en acabar.

Slurmdbd se cuelga

- La nueva opción MaxQueryTimeRange de slurmdbd.conf permite a los admins restringir el máximo rango de tiempo que los usuarios pueden pedir en una misma petición.
  - Peticiones que devuelvan > 3GB datos también fallarán.

```
test@box:~/t$ sacct -S 2007-01-01
JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
sacct: error: slurmdbd: Too wide of a date range in query
test@box:~/t$
```

High-Availability

- Mis preferencias:
  - Definir dos hosts, uno para slurmctld primario y otro para backup.
    - Unidos a un filesystem compartido y rápido para StateSaveLocation.
  - Co-locar slurmdbd con su propia instancia de MariaDB. No preocuparse de levantar un backup para slurmdbd.
  - Si se va corto de máquinas, hospedar el backup ctld en la misma máquina que slurmdbd. Pero mantenerlos separados por defecto.

High-Availability

- Problemas comunes:
  - StateSaveLocation necesita estar disponible (y rápido) a la vez tanto en primary como backup ctld.
  - Si el filesystem de StateSaveLocation falla, el cluster no funcionará. Elegir cuidadosamente, y evitar introducir complejidad innecesaria.
  - E.j. preferiría un sólo slurmctld con un disco local SSD para muchos entornos, en vez de un deployment más elaborado que involucre DRDB + GFS.

High-Availability

- Similarmente, para slurmdbd el dominio de fallo es la instalación de MySQL/MariaDB.
  - En mi experiencia, configurar HA para la DB es más probable que introduzcan más cortes que los fallos hw que intentan prevenir.
  - Aún así, para que quede claro, se deberían hacer backups.
  - slurmcltd puede sobrevivir y mantener el clúster corriendo mientras guarda temporalmente mensajes destinados para slurmdbd.
    - El tamaño del pool de mensajes está acotado por:
      - MAX(10000, 4xNodes + 2xMaxJobCount).

High-Availability

- Los triggers en slurmctld pueden ayudar a complementar la monitorización.
- Y sí, probablemente se debería monitorizar el clúster.
- Las opciones HealthCheck[Program|Interval] pueden ser de utilidad para detectar problemas en los nodos de cómputo, tomando las acciones convenientes como poner un nodo a DRAIN cuando se detecta un problema.

X11

- Desde 17.11 Slurm soporta x11 forwarding desde los nodos de cómputo y ya no es necesario configurar un SPANK plugin
  - Requisitos:
    - Paquete libssh2 instalado en el clúster
    - Paquete libssh2-devel en la máquina donde se compile Slurm
  - Configuración:
    - PrologFlags=x11 en slurm.conf
    - scontrol reconfigure
    - Configurar claves ssh RSA sin password para los usuarios