2º curso / 2º cuatr.

Grados en Ing. Informática

Arquitectura de Computadores

Seminario 0. Entorno de programación: atcgrid y gestor de carga de trabajo

Material elaborado por Mancia Anguita López y Julio Ortega Lopera Última modificación: 02/2022









AC A PIC

- Cluster de prácticas (atcgrid)
- Gestor de carga de trabajo (workload manager)
- > Ejemplo de script

AC NATC

- Cluster de prácticas (atcgrid)
 - > Componentes
 - > Placa madre
 - > Chips de procesamiento (procesadores)
 - > Acceso
- Gestor de carga de trabajo
- Ejemplo de script

Cluster de prácticas (atcgrid): componentes

AC S PTC

Nodos de cómputo (atcgrid[1-4]):

→ Tres servidores rack SuperMicro
SuperServer 6016T-T (atcgrid[1-3])
http://www.supermicro.com/products/system/10/6016/SYS-6016T-T.cfm





→ Un servidor rack SuperMicro SYS-6019U-TR4 1U (atcgrid4)
https://www.supermicro.com/products/system/1u/6019/SYS-6019U-TR4.cfm



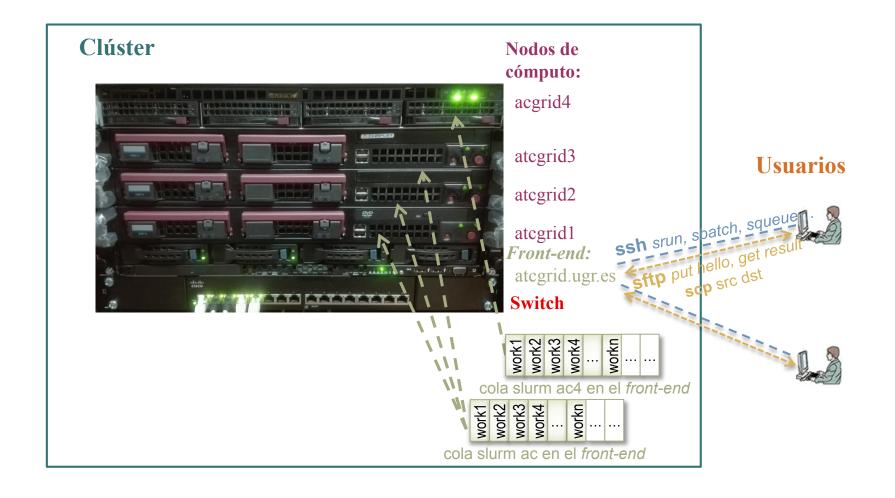




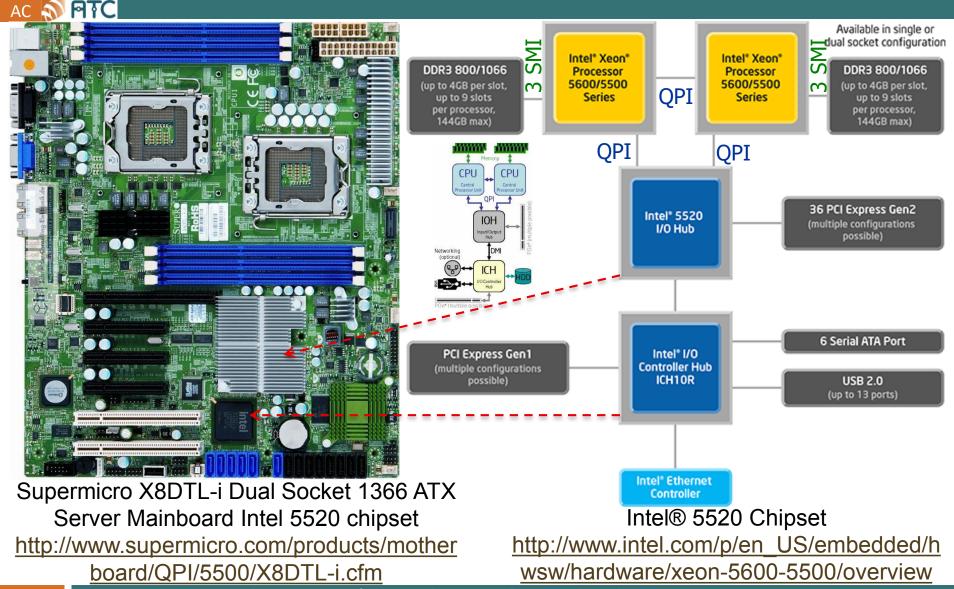
Nodo *front-end* (host, master): Asus RS300-E9-PS4

Cluster de prácticas (atcgrid): componentes

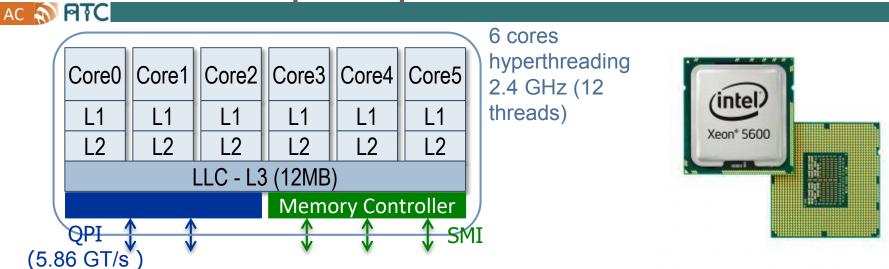


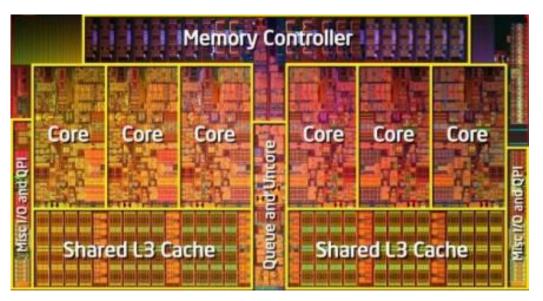


Cluster de prácticas>nodos atcgrid[1-3]: placa madre



Cluster de prácticas>nodos atcgrid[1-3]: chip de procesamiento

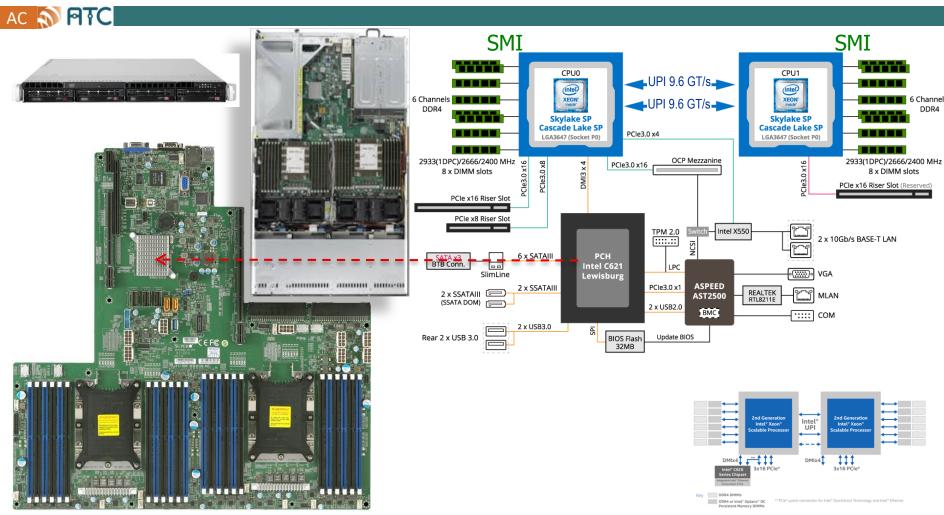




Intel Xeon E5645 (6 cores/12 threads, 12M L3 Cache compartida, 2.40 GHz cada core, 5.86 GT/s Intel® QPI)

http://ark.intel.com/products/48768
8?wapkw=(E5645)

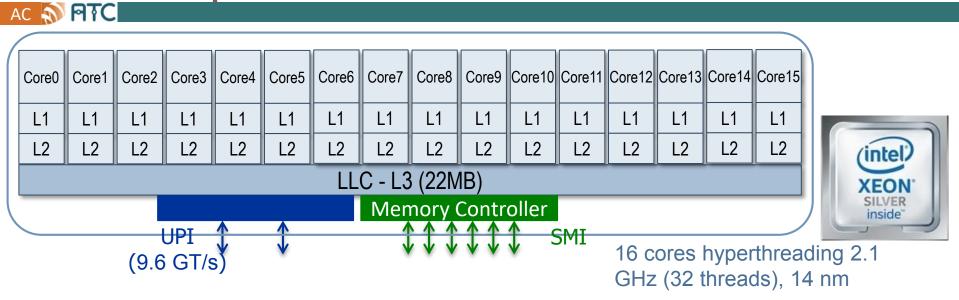
Cluster de prácticas> nodo atcgrid4: placa madre



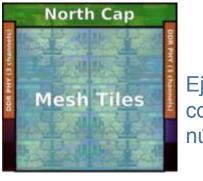
Supermicro SYS-6019U-TR4 1U https://www.supermicro.com/en/products/mothe rboard/X11DPU

Intel® C621 Chipset
Intel® C621 Chipset Product Specifications

Cluster de prácticas (atcgrid4): chip de procesamiento de la CPU



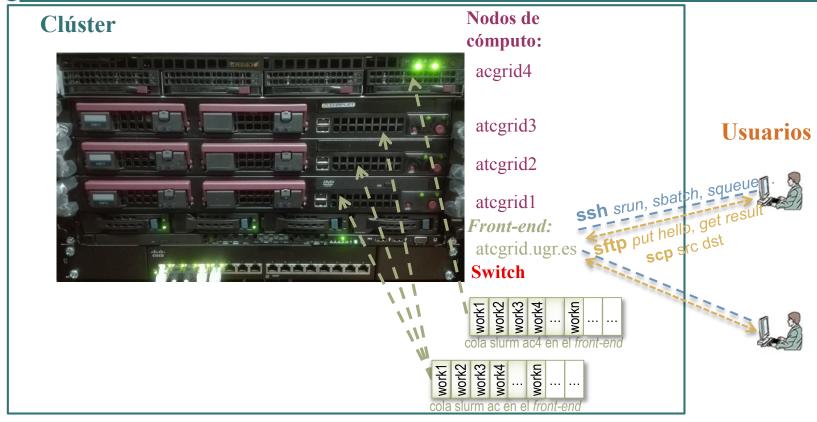
Intel® Xeon® Silver 4216 (16 cores/32 threads, 22MB L3 Cache compartida, 2.10 GHz máxima 3,2 GHz, cada core, 9.6 GT/s Intel® UPI)



Ejemplo con 18 núcleos

Cluster de prácticas (atcgrid): acceso





- Cada usuario tiene un home en el nodo front-end del clúster atcgrid. Se puede acceder al home:
 - > Para ejecutar comandos (srun, sbatch, squeue...), con un cliente ssh (secure shell):
 - Linux: \$ ssh -X username@atcgrid.ugr.es (pide password del usuario "username")
 - Para cargar y descargar ficheros (put hello, get slurm-9.out, ...), con un cliente sftp (secure file transfer protocol)
 - Linux: \$ sftp username@atcgrid.ugr.es (pide password del usuario "username")

AC N PTC

- Cluster de prácticas (atcgrid)
- > Gestor de carga de trabajo
- Ejemplo de script

Ejemplos con comandos slurm



> Se ejecutarán en el front-end con conexión ssh

Ejemplo	Explicación
<pre>srun -pac -Aac ./hello srun -p ac4 -A ac lscpu</pre>	srun envía a ejecutar un trabajo (en los ejemplos, el ejecutable hello, y Iscpu) a través de una cola slurm. Si aparece $-p$, se envía a nodos de la cola especificada con $-p$ (un trabajo solo puede usar un nodo de la cola ac).
<pre>sbatch -p ac script.sh sbatch -p acwrap "echo Hola" sbatch -p acwrap "./hello" sbatch -p acwrap "echo Hola; ./hello"</pre>	sbatch envía a ejecutar un script (en este caso script.sh, "echo Hola", "./hello" y "echo Hola; ./hello") a través de una cola slurm. La salida se devuelve en un fichero. La ejecución con srun es interactiva, con sbatch es en segundo plano. Se recomienda usar sbatch
squeue	Muestra todos los trabajos en ejecución y los que están encolados
scancel jobid	Elimina el trabajo con identificador "jobid"
sinfo	Lista información de las particiones (colas) y de los nodos
sinfo -p ac -o"%10D %10G %20b %f"	Lista los nodos (D), los recursos (G), y las características activas (b) y disponibles (f) en la partición especificada (-p) (%[[.]size]type[suffix])

Ejemplos con comandos slurm

AC SO PIC

Se ejecutarán en el front-end con conexión ssh

Ejemplo	Explicación
sbatch -pac -n1 -c12 script.sh srun -pac -n1 -c12 helloomp	Slurm está configurado para asignar recursos a los procesos (llamados tasks en slurm) a nivel de core físico. Esto significa que por defecto slurm asigna un core a un proceso, para asignar x se debe usar con sbatch/srun la opcióncpus-per-task=x (o su forma abreviada -cx). Para asegurar que solo se crea un proceso hay que incluir ntasks=1 (-n1) en sbatch/srun.
<pre>sbatch -pac -n1 -c12hint=nomultithread script.sh srun -pac -n1 -c12hint=nomultithread helloomp</pre>	En slurm, por defecto, cpu se refiere a cores lógicos (ej. en la opción -c), si no se quieren usar cores lógicos hay que añadir la opciónhint=nomultithread a sbatch/srun. sbatch tendrá en cuentahint=nomultithread si se usa srun dentro del script delante del ejecutable.
<pre>sbatch -pac -n1 -c12exclusivehint=nomultithread script.sh srun -pac -n1 -c12exclusivehint=nomultithread helloomp</pre>	Para que no se ejecute más de un proceso en un nodo de cómputo de atcgrid hay que usarexclusive con sbatch/srun (se recomienda no utilizarlo en los srun dentro de un script).

Tabla resumen: https://slurm.schedmd.com/pdfs/summary.pdf
Páginas de manual: https://slurm.schedmd.com/man_index.html

Particiones slurm (colas) en atcgrid



```
$ sinfo
PARTITION AVAIL
                              NODES
                                      STATE NODELIST
                  TIMELIMIT
                       1:00
                                       idle atcgrid[1-3]
ac*
              up
ac4
                       1:00
                                       idle atcgrid4
              up
                                       idle atcgrid[1-3]
                       2:00
aapt
              up
                       1:00
                                       idle atcgrid[1-3]
acap
              up
```

* significa que ac es la cola utilizada por defecto, es decir, cuando no se usa -p

AC N PTC

- Cluster de prácticas (atcgrid)
- Gestor de carga de trabajo
- > Ejemplo de script

Ejemplo hello OpenMP

AC N PTC

- Cada thread imprime su identificador
 - > El identificador se obtiene
 con la función OpenMP
 omp get thread num()

HelloOMP.c

```
/* Compilar con:
gcc -O2 -fopenmp -o HelloOMP HelloOMP.c
#include <stdio.h>
#include <omp.h>
int main(void) {
#pragma omp parallel
  printf("(%d:!!!Hello world!!!)",
            omp get thread num());
return(0);
```

Script para la ejecución del ejemplo HelloOMP en atcgrid



script_helloomp.sh

```
#!/bin/bash
#Órdenes para el Gestor de carga de trabajo (no intercalar instrucciones del script):
#1. Asigna al trabajo un nombre
#SBATCH -- job-name=helloOMP
#2. Asignar el trabajo a una cola (partición)
#SBATCH --partition=ac
#3. Asignar el trabajo a un account
#SBATCH --account=ac
#4. Para que el trabajo no comparta recursos #SBATCH -exclusive
#5. Para que se genere un único proceso del SO que pueda usar un máximo de 12 núcleos
#SBATCH --ntasks 1 --cpus-per-task 12
#Obtener información de las variables del entorno del Gestor de carga de trabajo:
echo "Id. usuario del trabajo: $SLURM JOB USER"
echo "Id. del trabajo: $SLURM_JOBID"
echo "Nombre del trabajo especificado por usuario: $SLURM JOB NAME"
echo "Directorio de trabajo (en el que se ejecuta el script): $$LURM $UBMIT DIR"
echo "Cola: $SLURM JOB PARTITION"
echo "Nodo que ejecuta este trabajo:$SLURM_SUBMIT_HOST"
echo "Nº de nodos asignados al trabajo: $SLURM JOB NUM NODES"
echo "Nodos asignados al trabajo: $SLURM JOB NODELIST"
echo "Nº CPUs disponibles para el trabajo en el nodo: $SLURM JOB CPUS PER NODE"
#Instrucciones del script para ejecutar código:
echo -e "\n 1. Ejecución helloOMP una vez sin cambiar nº de threads (valor por defecto):\n"
srun ./HelloOMP
echo -e "\n 2. Ejecución helloOMP varias veces con distinto nº de threads:\n"
for ((P=12;P>0;P=P/2))
  #export OMP NUM THREADS=$P
  echo -e "\n - Para $P threads:"
   srun --cpus-per-task=$P --hint nomultithread ./HelloOMP
done
       No olvidar poner srun delante del ejecutable
```

Instrucciones de script

estor de

s del carga

Se puede descomentar export OMP NUM THREADS=\$P y quitar --cpus-per-task=\$P

Utilidades

AC S PTC

- Formateo de código a insertar en los cuadernos:
 - https://pinetools.com/syntax-highlighter,
 - https://highlight.hohli.com/index.php,
 - https://tohtml.com/

Tiempos WSL1 vs WSL2 (máquina virtual con poca sobrecarga) vs IDE

AC NATC

WSL1 (daxpy_alfabeta_omp.c)

```
Nº threads: 1 / Tiempo:0.020914700
Nº threads: 2 / Tiempo:0.014129800
Nº threads: 4 / Tiempo:0.012987300
Nº threads: 1 / Tiempo:0.020808200
Nº threads: 2 / Tiempo:0.014388400
Nº threads: 4 / Tiempo:0.012656300
Nº threads: 1 / Tiempo:0.020901700
Nº threads: 2 / Tiempo:0.014201200
Nº threads: 4 / Tiempo:0.013142600
Nº threads: 1 / Tiempo:0.020793800
Nº threads: 2 / Tiempo:0.013782500
Nº threads: 4 / Tiempo:0.013168300
Nº threads: 1 / Tiempo:0.021101600
Nº threads: 2 / Tiempo:0.014150400
Nº threads: 4 / Tiempo:0.013186700
```

WSL2 (daxpy_alfabeta_omp.c)

Nº threads: 1 / Tiempo:0.028192300
Nº threads: 2 / Tiempo:0.012024500
Nº threads: 4 / Tiempo:0.014016100
Nº threads: 1 / Tiempo:0.027466000
Nº threads: 2 / Tiempo:0.013317100
Nº threads: 4 / Tiempo:0.013590700
Nº threads: 1 / Tiempo:0.036062800
Nº threads: 2 / Tiempo:0.017655300
Nº threads: 4 / Tiempo:0.015197700

Ejecución desde un IDE:

Nº threads: 1 / Tiempo:0.053106200 Nº threads: 2 / Tiempo:0.020059500 Nº threads: 4 / Tiempo:0.029696300 Nº threads: 1 / Tiempo:0.060597600 Nº threads: 2 / Tiempo:0.025821100 Nº threads: 4 / Tiempo:0.030059800