AC-P0.pdf

patrivc

Arquitectura de Computadores

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

AC SEMINARIO 0. Entorno de programación: atcgrid y gestor de carga de trabajo.

Clúster: es un grupo de ordenadores que se unen mediante una red de alta velocidad, de tal forma que el conjunto se ve como un único ordenador, mucho más potente que los ordenadores comunes. Una de sus principales ventajas es que no es necesario que los equipos que lo integren sean iguales a nivel hardware ni que dispongan del mismo sistema operativo.

Nodo de un clúster: son cada uno de los elementos del cluster. Es el nombre genérico que se dará a cualquier máquina que utilicemos para montar un cluster, como pueden ser ordenadores de sobremesa o servidores. Aún cuando podemos utilizar cualquier tipo de hardware para montar nuestro sistema, es siempre buena idea que haya cierto parecido entre las capacidades de todos los nodos ya que, en caso contrario, habrá siempre cierta tendencia o enviar el trabajo a realizar a aquel equipo que disponga de una mayor capacidad de procesamiento.

Front-end: Son los nodos que interactúan con el usuario / mundo, a diferencia de los nodos de computación que simplemente se alimentan de algoritmos y datos para procesar.

Nodos de cómputo que nos vamos a encontrar en estas prácticas: atcgrid4, atcgrid3, atcgrid2, atcgrid1.

Ejemplos de comandos:

srun -pac -Aac ./hello **srun -p -ac4 -A ac lscpu**

-srun envia a ejecutar un trabajo a través de una cola slurm
-p se envía a nodos de la cola especificada con -p (un trabajo solo puede usar un nodo de la cola ac)

sbatch -p ac script.sh / sbatch -p ac - -wrap "echo Hola" / sbatch -p ac - -wrap "./ hello"
sbatch -p ac - -wrap "echo Hola ; ./hello"

-sbatch envia a ejecutar un script 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 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 *-pac -o"%10D %10G %20b %f"*

Lista los nodos (D), los recursos (G) y las características activas (b) y disponibles (f) en la participación especificada (-p) (%[[.]size]type[suffix])

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

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

El identificador de cada thread se obtiene con **omp_get_thread_num()**

SBATCH - -job-name=HelloOMP

Asigna al trabajo un nombre

SBATCH - -partition=ac

Asignar el trabajo a una partición (cola)

SBATCH - -account=ac

Asignar el trabajo a un account

Por defecto, slurm asigna un core a un proceso, para asignar mas de uno se debe usar **sbatch/srun - -cpus-per-task**

En **slurm,** por defecto, **cpu** se refiere a cores lógicos. Si **no** se quiere usar cores lógicos hay que añadir la opción **- -hint=nomultithread** a **sbatch/srun**

Para asegurarnos de que solo se crea un procesos hay que incluir **--ntasks=1 (-n1)** en **sbatch/srun**

Para que no se ejecute más de un proceso en un nodo de atcgrid hay que usar **- -exclusive** con **sbatch/srun**

Los srun dentro de un script heredan las opciones fijadas en el sbacth que se usa para enviar el script a la columna slurm.

Las opciones de sbatch se pueden especificar también dentro del script (usando #SBATCH, ver ejemplos en el script del seminario)

Cores físicos: sockets * cores por socket
Cores lógicos: físicos * hebras

Parallel de omp se encarga de crear hilos para explotar el paralelismo que permite la CPU.

Ejemplo: ¿Qué orden srun usaría para que HelloOMP utilice los 12 cores físicos de un nodo de cómputo de atcgrid (se debe imprimir un único mensaje desde cada uno de ellos, en total, 12)?

srun -p ac -n1 --cpus-per-task=12 --hint=nomultithread HelloOMP

-n1 -> para crear un solo proceso
-p ac -> escoge la cola ac

--cpus-per-task=12 -> para que srun use 12 cores fisicos
--hint=nomultithread -> especificamos que no use cores logicos

sacctmgr Ver y modificar la información de la cuenta

sacct Muestra los datos contables

salloc Obtener una asignación de trabajo

sbcast Transfiere el archivo a los nodos de cálculo de un trabajo

scancel Trabajos de señal, matrices de trabajo y/o pasos de trabajo

scontrol Se utiliza para ver y modificar la configuración y el estado. También ve la interfaz gráfica de un usuario.

Utilizar siempre con **sbatch** las opciones –**n1** y **--cpus-per-task**, **--exclusive** y, para usar cores físicos y no lógicos, no olvide incluir **--hint=nomultithread**.

Utilizar siempre con **srun**, si lo usa fuera de un script, las opciones **–n1** y **--cpus-per-task** y, para usar cores físicos y no lógicos, no olvide incluir **--hint=nomultithread**.

Recordar que los **srun** dentro de un script heredan las opciones utilizadas en el **sbatch** que se usa para enviar el script a la cola **slurm**.

Se recomienda usar **sbatch** en lugar de **srun** para enviar trabajos a ejecutar a través **slurm** porque éste último deja bloqueada la ventana hasta que termina la ejecución, mientras que usando **sbatch** la ejecución se realiza en segundo plano.

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 acceder a atcgrid)

-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”)

Los **vectores locales** se almacenan en la pila.
Los **vectores dinámicos** se almacenan en el heap.

Los **vectores globales** se almacenan en memoria.

En atcgrid[1-3] -> tenemos 24 cores lógicos y 12 cores físicos

En atcgrid4 -> hay 64 cores lógicos y 32 cores físicos.

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) (32 cores lógicos y 16 cores físicos)

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

Intel Xeon 5600 (6 cores hyperthreading 2.4 GHz (12 threads)) (6 cores físicos, 12 cores lógicos)

Ejemplo: ¿en qué estructura de datos devuelve clock_gettime() la información de tiempo? ¿qué información devuelve exactamente la función clock_gettime()en la estructura de datos descrita en el apartado (b)? ¿qué representan los valores numéricos que devuelve?

- La función **clock_gettime()** tiene el siguiente prototipo:
  int clock_gettime(clockid_t clk_id, struct timespec *tp);
- El primer argumento de la función hace referencia a un FLAG para especificar cómo obtener el tiempo. En el código SumaVectoresC.c se utiliza **CLOCK_REALTIME**, que muestra el tiempo real del sistema con la mayor precisión posible.
- El segundo argumento es un puntero a un struct timespec en el que se almacenarán los datos. Para ello, cuando se vaya a llamar a la función clock_gettime(), hay que pasar el segundo argumento por referencia.
- La función devuelve un entero que puede ser 0 si se ha calculado todo perfectamente, o -1 en caso de error.
- La estructura de datos del segundo argumento es de la siguiente forma:

```
strut timespec{
        time_t tv_sec;          /* segundos */
        long tv_nsec;           /* nanosegundos */
}
```

La función obtiene el tiempo del reloj especificado en el FLAG pasado como argumento 1 de la función. En este caso **CLOCK_REALTIME**.

Devuelve el tiempo en segundos y en nanosegundos. La forma de obtener este tipo de datos es a través de la representación que realiza UNIX del tiempo, a través de un contador numérico que permite convertir posteriormente a un formato legible por los humanos. El inicio de este contador está en el 1 de Enero de 1970 a las 00:00 (como valor numérico tendría 0).

A la fecha que estoy realizando este ejercicio tendría valor 1582985195, que traducido a Fecha legible sería 29 de Febrero de 2020 a las 14:06:35. Estos contadores son los que se almacenan en el struct.

Ejemplo: ¿Cuál es el máximo valor que se puede almacenar en la variable N teniendo en cuenta su tipo? Razonar respuesta.

La variable **N** es de tipo ***Unsigned Int***, es decir, *Entero Sin Signo.* Comprende el rango de valores desde
el 0 hasta 2^(32)-1 (Ya que el dato es de 4Bytes, que son 32 bits), por tanto, puede tener como valor máximo ésta última cantidad, que en formato numérico es 4294967295.

Ejemplo: Escribir en el cuaderno de prácticas las diferencias que hay entre el código fuente C y el código fuente C+ + para la suma de vectores.

Descripción diferencia	En C	En C++
En c se declara al principio la variable para rellenar el vector, en c++ se puede dentro del for	int i;	for(int i=0...
En c uso punteros a los vectores	double *v1, *v2, *v3;	double v1[N],v2[N],v3[N];
Para imprimir por pantalla	printf	cout
Los vectores se utilizan dinámicamente sin necesidad de usar malloc y realloc en c++.	malloc y realloc	No necesita

Limpia los vectores	#ifdef VECTOR_DYNAMIC
	free(v1);
	free(v2);
	free(v3);
		#ifdef VECTOR_DYNAMIC
		delete [] v1;
		delete [] v2;
		delete [] v3;

Ejercicios:

1. **Se ha usado la siguiente orden para ejecutar script.sh en un nodo de cómputo atcgrid[1-3]: sbatch -p -ac - -account=ac -n1 - -cpus-per-task=1 --hint=nomultithread. Teniendo en cuenta esta orden y el proceder por defecto de la instalación de slurm de atcgrid, va a utilizar un máximo de núcleos lógicos de**

   a) 1.↵

   b) 2

   c) Ninguna respuesta es correcta

   d) 3

2. **¿Qué orden puedo utilizar para conectarme al front-end de atcgrid para enviar/recibir ficheros?**

   a) sftp usuario@atcgrid.ugr.es ↵

   b) ssh usuario@atcgrid.ugr.es

   c) telnet usuario@atcgrid.ugr.es

   d) sbatch -p ac script.sh

3. **¿Con qué comando SLURM se pueden ver todos los trabajos en ejecución y los que están encolados?**

   a) sjobs

   b) sinfo

   c) snumjobs

d) squeue ↵

4. **¿En qué zona de memoria se alojan los vectores dinámicos de un programa en c?**

   a) heap ↵

   b) data

   c) Ninguna otra respuesta es correcta

   d) stack

5. **En un script para SLURM, ¿qué significa la línea #SBATCH - -job-name=helloOMP?**

   a) El nombre asignado al trabajo ↵

   b) El usuario que manda el trabajo

   c) El directorio donde está el ejecutable

   d) La cola a la que se manda el trabajo

6. **¿Cuál de las siguientes órdenes usaría para que se cree un único proceso con el código paralelo HelloOMP que use 12 hebras, de forma que cada hebra sea distinto de un nodo de cómputo atcgrid[1-3]?**

   a) srun -p ac - -account=ac -n1 HelloOMP

   b) srun -p ac - -account=ac -n1 - -cpus-per-task=8 HelloOMP

   c) srun -p ac - -account=ac -n1 - -cpus-per-task=24 HelloOMP

   d) srun -p ac - -account=ac -n1 - -cpus-per-task=12 --hint=nomultithread HelloOMP↵

7. **¿Cuántos zócalos (sockets) tiene el cluster atcgrid en total sin contar el host o front-end?**

   a) 2

   b) 4

   c) 8 ↵

   d) 12

8. **¿Cuántos cores lógicos tiene cada microprocesador Intel Xeon Silver 4216?**

   a) 64

   b) 12

   c) 32 ↵

   d) 16

9. **¿Qué opción de sbatch evita que haya otros trabajos ejecutándose simultáneamente en el mismo nodo al que se envía su trabajo?**

   a) - -exclusive ↵

   b) Ninguna respuesta es correcta

   c) - -hint=nomultithread

   d) -p ac -n1

10. **¿Cuántos cores lógicos tiene cada nodo de cómputo atcgrid[1-3]?**

    a) 6

    b) 12

    c) 3

    d) 24 ↵

#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): $SLURM_SUBMIT_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 "CPUs por nodo: $SLURM_JOB_CPUS_PER_NODE"
```