

## Arquitectura de Computadores (AC)

### Cuaderno de prácticas.

### Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos):

Grupo de prácticas:

Fecha de entrega:

Fecha evaluación en clase:

Ejercicios basados en los ejemplos del seminario práctico

## importante: compilar con -fopenmp

1. En el primer ejemplo de ejecución en atcgrid usando TORQUE se ejecuta el ejemplo HelloOMP.c usando la siguiente orden: `echo 'hello/HelloOMP' | qsub -q ac`. El resultado de la ejecución de este código en atcgrid se puede ver en el seminario. Conteste a las siguientes preguntas:

- a. ¿Para qué se usa en qsub la opción -q?

**RESPUESTA:** qsub hace que se cree un trabajo que ejecutará el script dado (en este caso HelloOMP) en un cliente. La opción -q permite definir el destino al que se envía dicho trabajo (la cola de trabajos a la que enviamos la petición, ac, en este caso).

- b. ¿Cómo sabe el usuario que ha terminado la ejecución en atcgrid?

**RESPUESTA:** Cuando se completa la ejecución el servidor responde con un número seguido de “.atcgrid”. Además, se generan dos archivos cuyo sufijo va precedido de e y o respectivamente con el resultado (salida) de la ejecución del programa y la salida de error del mismo.

- c. ¿Cómo puede saber el usuario si ha habido algún error en la ejecución?

**RESPUESTA:** Consultando el contenido del archivo con la salida de error generado durante la ejecución del programa.

- d. ¿Cómo ve el usuario el resultado de la ejecución?

**RESPUESTA:** Llevando el archivo de salida (cuyo sufijo comienza con una “o”) al ordenador local y consultándolo con cualquier editor de texto o la orden cat.

- e. ¿Por qué en el resultado de la ejecución aparecen 24 saludos “¡¡¡Hello World!!!”?

**RESPUESTA:** Porque se ha asignado un nodo entero de la placa y se están creando 24 threads.

2. En el segundo ejemplo de ejecución en atcgrid usando TORQUE el script `script_helloomp.sh` usando la siguiente orden: `qsub script_helloomp.sh`. El script ejecuta varias veces el ejecutable del código HelloOMP.c. El resultado de la ejecución de este código en atcgrid se puede ver en el seminario. Conteste a las siguientes preguntas:

- a. ¿Por qué no acompaña a al orden qsub la opción -q en este caso?

**RESPUESTA:** porque en el propio script se utiliza un comando de PBS script (`#PBS -q ac`) que asigna a los trabajos la cola ac.

- b. ¿Cuántas veces ejecuta el script el ejecutable HelloOMP en atcgrid? ¿Por qué lo ejecuta ese número de veces?

**RESPUESTA:** Se ejecuta 4 veces, porque está programado para ejecutarse con 12 Threads en un principio e ir ejecutándose con la mitad tantas veces como pueda mientras el número de threads sea mayor que 0. Por eso se ejecuta con 12, 6, 3 y 1 thread.

- C. ¿Cuántos saludos “¡¡¡Hello World!!!” se imprimen en cada ejecución? (indique el número exacto) ¿Por qué se imprime ese número?

**RESPUESTA:** 12, 6, 3 y 1, que se corresponden con el número de threads creados. Se imprime ese número porque el programa escribe un hello world precedido del número de thread una vez por cada thread.

- Realizar las siguientes modificaciones en el script “`!!!Hello World!!!`”:
  - Eliminar la variable de entorno `$PBS_O_WORKDIR` en el punto en el que aparece.
  - Añadir lo necesario para que, cuando se ejecute el script, se imprima la variable de entorno `$PBS_O_WORKDIR`.

Ejecutar el script con estas modificaciones. ¿Qué resultados de ejecución se obtienen en este caso? Incorporar en el cuaderno de trabajo volcados de pantalla que muestren estos resultados.

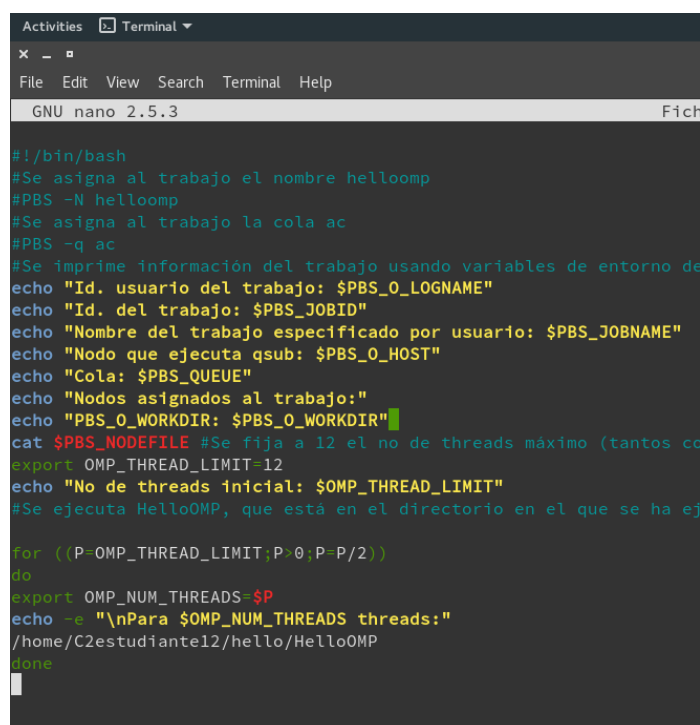
**RESPUESTA:** Al eliminar la variable de entorno se obtiene el resultado de la página siguiente. En un principio puede parecer que no hay ningún error pero al abrir el archivo con la salida de error encontramos:

```
[C2estudiante12@atcgrid hello]$ ls  
helloOMP    helloomp.e41748  helloomp.o41748  script  
[C2estudiante12@atcgrid hello]$ cat helloomp.o41748  
Id. usuario del trabajo: C2estudiante12  
Id. del trabajo: 41748.atcgrid  
Nombre del trabajo especificado por usuario: helloomp  
Nodo que ejecuta qsub: atcgrid  
Cola: ac  
Nodos asignados al trabajo:  
PBS_O_WORKDIR: /home/C2estudiante12/hello  
atcgrid1  
atcgrid1  
atcgrid1  
atcgrid1  
atcgrid1  
atcgrid1  
atcgrid1  
atcgrid1  
atcgrid1  
atcgrid1  
atcgrid1  
atcgrid1  
atcgrid1  
atcgrid1  
atcgrid1  
atcgrid1  
atcgrid1  
atcgrid1  
atcgrid1  
atcgrid1  
No de threads inicial: 12  
  
Para 12 threads:  
  
Para 6 threads:  
  
Para 3 threads:  
  
Para 1 threads:  
[C2estudiante12@atcgrid hello]$
```

un principio puede parecer que no hay ningún error pero al abrir el archivo con la salida de error encontramos:

```
/var/lib/torque/mom_priv/jobs/41752.atcgrid.SC:      line  
23: /hello/HelloOMP: No such file or directory  
  
/var/lib/torque/mom_priv/jobs/41752.atcgrid.SC:      line  
23: /hello/HelloOMP: No such file or directory  
  
/var/lib/torque/mom_priv/jobs/41752.atcgrid.SC:      line  
23: /hello/HelloOMP: No such file or directory  
  
/var/lib/torque/mom_priv/jobs/41752.atcgrid.SC:      line  
23: /hello/HelloOMP: No such file or directory
```

Es decir, no está encontrando al programa a ejecutar, porque no tiene el path completo. Al imprimir la variable \$PBS\_O\_WORKDIR podemos ver que contiene, precisamente, la ruta /home/C2estudiante12/hello, que es donde se encuentra el script (C2estudiante12 es mi directorio asignado).



```

GNU nano 2.5.3

#!/bin/bash
#Se asigna al trabajo el nombre helloomp
#PBS -N helloomp
#Se asigna al trabajo la cola ac
#PBS -q ac
#Se imprime información del trabajo usando variables de entorno de
echo "Id. usuario del trabajo: $PBS_O_LOGNAME"
echo "Id. del trabajo: $PBS_JOBID"
echo "Nombre del trabajo especificado por usuario: $PBS_JOBNAME"
echo "Nodo que ejecuta qsub: $PBS_O_HOST"
echo "Cola: $PBS_QUEUE"
echo "Nodos asignados al trabajo:"
echo "PBS_O_WORKDIR: $PBS_O_WORKDIR"
cat $PBS_NODEFILE #Se fija a 12 el no de threads maximo (tantos co
export OMP_THREAD_LIMIT=12
echo "No de threads inicial: $OMP_THREAD_LIMIT"
#Se ejecuta HelloOMP, que está en el directorio en el que se ha ej

for ((P=OMP_THREAD_LIMIT; P>0; P=P/2))
do
export OMP_NUM_THREADS=$P
echo -e "\nPara $OMP_NUM_THREADS threads:"
/home/C2estudiante12/hello/HelloOMP
done

```

## Resto de ejercicios

4. Incorporar en el fichero .zip que se entregará al profesor el fichero /proc/cpuinfo de alguno de los nodos de atcgrid (atcgrid1, atcgrid2, atcgrid3), y del PC del aula de prácticas o de su PC. Indique qué ha hecho para obtener el contenido de /proc/cpuinfo en atcgrid.

**RESPUESTA:** He ejecutado: `echo "cat /proc/cpuinfo > atcgrid_cpuinfo" | qsub -q ac` y luego he traído a mi computador el archivo con la salida

para obtener la información en atcgrid he ejecutado desde el terminal que operamos con ssh: `'cat /proc/cpuinfo > atcgrid_info'` y luego lo he traído a mi ordenador desde el terminal abierto con sftp usando `get atcgrid_info`. Error → esto da la información del nodo front-end

Teniendo en cuenta el contenido de cpuinfo conteste a las siguientes preguntas (justifique las respuestas):

a. ¿Cuántos cores físicos y cuántos cores lógicos tiene el PC del aula de prácticas o su PC?

**RESPUESTA:** para saber los cores físicos nos fijamos en el apartado 'core\_id' de cada núcleo y vemos que, en mi caso, se repiten 0,1,2,3 dos veces. Es decir, que mi computador tiene 4 cores físicos. Para saber los lógicos, nos fijamos en el valor del campo 'processor', que en mi caso no se repite y va del 0 al 7, por lo que puedo afirmar que mi computador tiene 7 cores lógicos.

b. ¿Cuántos cores físicos y cuántos cores lógicos tiene un nodo de atcgrid?

**RESPUESTA:** De la misma manera pero con el archivo de atcgrid podemos deducir que un nodo de atcgrid tiene 24 cores lógicos y 6 físicos.

5. En el Listado 1 se puede ver un código fuente C que calcula la suma de dos vectores y en el Listado 2 una versión con C++:

```
v3 = v1 + v2; v3(i) = v1(i) + v2(i), i=0,...N-1
```

Los códigos utilizan directivas del compilador para fijar el tipo de variable de los vectores (v1, v2 y v3). En los comentarios que hay al principio de los códigos se indica cómo hay que compilarlos. Los vectores pueden ser:

- Variables locales: descomentando en el código `#define VECTOR_LOCAL` y comentando `#define VECTOR_GLOBAL` y `#define VECTOR_DYNAMIC`
- Variables globales: descomentando `#define VECTOR_GLOBAL` y comentando `#define VECTOR_LOCAL` y `#define VECTOR_DYNAMIC`
- Variables dinámicas: descomentando `#define VECTOR_DYNAMIC` y comentando `#define VECTOR_LOCAL` y `#define VECTOR_GLOBAL`. Si se usan los códigos tal y como están en Listado 1 y Listado 2, sin hacer ningún cambio, los vectores (v1, v2 y v3) serán variables dinámicas.

Por tanto, se debe definir sólo una de las siguientes constantes: `VECTOR_LOCAL`, `VECTOR_GLOBAL` o `VECTOR_DYNAMIC`.

- a. En los dos códigos (Listado 1 y Listado 2) se utiliza la función `clock_gettime()` para obtener el tiempo de ejecución del trozo de código que calcula la suma de vectores. En el código se imprime la variable `ncgt`, ¿qué contiene esta variable? ¿qué información devuelve exactamente la función `clock_gettime()`? ¿en qué estructura de datos devuelve `clock_gettime()` la información (indicar el tipo de estructura de datos y describir la estructura de datos)?

**RESPUESTA:** `clock_gettime` devuelve un `struct timespec`, que representa un intervalo dado en segundos y nanosegundos desde un momento establecido por convención (el 1 de enero de 1970), por ello, la línea:

```
ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+(double) ((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));
```

está calculando el tiempo exacto que ha pasado entre las dos llamadas a `clock_gettime`, que han sido almacenadas en `cgt1` y `cgt2`. Ojo, tanto el valor en segundos como el valor en nanosegundos representan el mismo tiempo; es decir, si uno marcara horas y el otro minutos, no sería que el segundo marcara los minutos hasta la siguiente hora, sino que ambos marcarían el tiempo que ha pasado desde aquel momento hasta ahora, uno en horas y otro en minutos. Eso pasa con los segundos y nanosegundos, por eso no puede darse el caso de que los nanosegundos posteriores sean menores que los previos aunque haya pasado exactamente un segundo entre uno y otro, los nanosegundos no se reinician al aumentar los segundos.

- b. 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.

**RESPUESTA:**

Descripción diferencia	En C	En C++
BIBLIOTECAS	Stdlib, stdio	Iostream, cstdlib(es la misma)
Memoria dinámica	malloc	new
I/O	Printf	Cout/cin
Liberar memoria	free	delete
Indice bucles	Int i al principio de main	for(int i = 0 ; i < ... ; i++)

6. Generar el ejecutable del código fuente C del Listado 1 para vectores locales (para ello antes de compilar debe descomentar la definición de `VECTOR_LOCAL` y comentar las definiciones de `VECTOR_GLOBAL` y `VECTOR_DYNAMIC`). Ejecutar el código ejecutable resultante en atcgrid usando el la cola TORQUE. Incorporar volcados de pantalla que demuestren la ejecución correcta en atcgrid.

**RESPUESTA:**

```

pinguino@1N0: ~/hello
C2estudiante12@atcgrid:~$ ls
atcgrid_cpuinfo  hello  SumaVectores
C2estudiante12@atcgrid:~$ echo './SumaVectores 10000' | qsub -q ac
atcgrid:43301.atcgrid
C2estudiante12@atcgrid:~$ ls
atcgrid_cpuinfo  hello  STDIN.e43301  STDIN.o43301  SumaVectores
C2estudiante12@atcgrid:~$ cat S
STDIN.e43301  STDIN.o43301  SumaVectores
[0](1
C2estudiante12@atcgrid:~$ cat STDIN.o43301
Tiempo(seg.):0.000052115 / Tamaño Vectores:10000 / V1[0]+V2[0]=V3
[0](1000.000000+1000.000000=2000.000000) / / V1[9999]+V2[9999]=V3[9999](1999.900
000+0.100000=2000.000000) /
[1]
C2estudiante12@atcgrid:~$

```

7. Ejecutar en atcgrid el código generado en el apartado anterior usando el script del Listado 3. Generar el ejecutable usando la opción de optimización `-O2` tal y como se indica en el comentario que hay al principio del programa. Ejecutar el código también en su PC local para los mismos tamaños. ¿Se obtiene error para alguno de los tamaños? En caso afirmativo, ¿a qué se debe este error?

**RESPUESTA:**

Se obtienen segmentation faults (core dumped) para el tamaño de 262144. Estos se deben a que los vectores locales (por su tamaño) superan el tamaño máximo de la pila del programa y a la hora de reservar la memoria estáticamente necesaria para dichos valores se accede a posiciones fuera de la memoria asignada la pila y se producen segmentation faults.

8. Generar los ejecutables del código fuente C para vectores globales y para dinámicos. Genere el ejecutable usando `-O2`. Ejecutar los dos códigos en atcgrid usando un script como el del Listado 3 (hay que poner en el script el nombre de los ficheros ejecutables generados en este ejercicio) para el mismo rango de tamaños utilizado en el ejercicio anterior. Ejecutar también los códigos en su PC local. ¿Se obtiene error usando vectores globales o dinámicos? ¿A qué cree que es debido?

**RESPUESTA:**

dinamico:

atc:

9. Rellenar una tabla como la Tabla 1 para atcgrid y otra para el PC local con los tiempos de ejecución obtenidos en los ejercicios anteriores para el trozo de código que realiza la suma de vectores. En la columna “Bytes de un vector” hay que poner el total de bytes reservado para un vector. Ayudándose de una hoja de cálculo represente en una misma gráfica los tiempos de ejecución obtenidos en atcgrid para vectores locales, globales y dinámicos (eje y) en función del tamaño en bytes de un vector (eje x). Utilice escala logarítmica en el eje de ordenadas (eje y) en todas las gráficas. ¿Hay diferencias en los tiempos de ejecución con vectores locales, globales y dinámicos?

### RESPUESTA:

**Tabla 1** .Tiempos de ejecución de la suma de vectores para vectores locales, globales y dinámicos

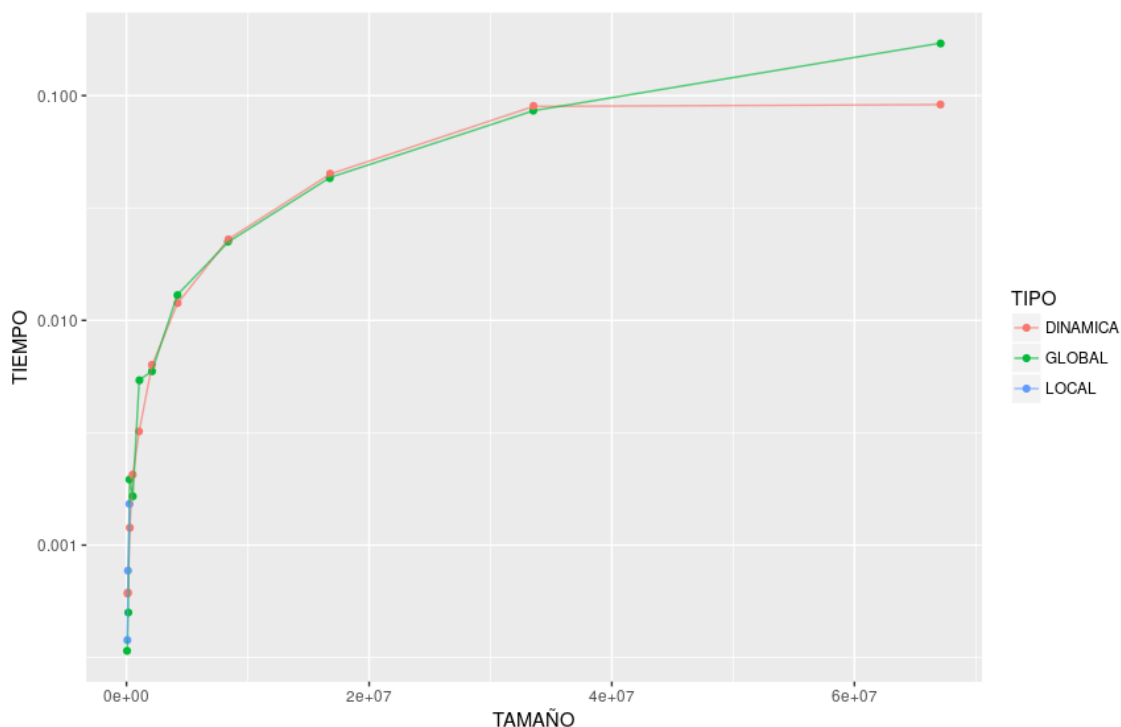
Atc:

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000377658		
131072	1048576	0.000765141		
262144	2097152	0.001534730		
524288	4194304			
1048576	8388608			
2097152	16777216			
4194304	33554432			
8388608	67108864			
16777216	134217728			
33554432	268435456			
67108864	536870912			

nota: me faltaba completar este cuadro pero el servidor de atcgrid se ha caído y no puedo completarlo. La gráfica está realizada con los tiempos tomados en mi ordenador personal.

Mi pc

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000377658	0.000337390	0.000607448
131072	1048576	0.000765141	0.000497967	0.000619128
262144	2097152	0.001534730	0.001944658	0.001190672
524288	4194304		0.001659244	0.002073463
1048576	8388608		0.005437451	0.003212821
2097152	16777216		0.005932408	0.006301071
4194304	33554432		0.013000507	0.011867425
8388608	67108864		0.022335344	0.022955541
16777216	134217728		0.043246462	0.044764227
33554432	268435456		0.085736456	0.089530898
67108864	536870912		0.171324702	0.091177003



**10.** Modificar el código fuente C para que el límite de los vectores cuando se declaran como variables globales sea igual al máximo número que se puede almacenar en la variable N ( $MAX=2^{32}-1$ ). Generar el ejecutable usando variables globales. ¿Qué ocurre? ¿A qué es debido? Razone además por qué el máximo número que se puede almacenar en N es  $2^{32}-1$ .

### RESPUESTA:

Aparece este error:

```
sumaVectores.c:(.text.startup+0x69): relocation truncated to fit: R_X86_64_32S against symbol `v2'
defined in COMMON section in /tmp/ccgFEJiw.o
```

para los vectores v2 y v3.

El máximo número que se puede almacenar en N es  $2^{32}-1$  porque N es una variable entera (de tipo int) que se codifica con 4 Bytes ( $8 \times 4 = 32$  bits)