


<p>2º curso / 2º cuatr.</p> <p>Grado Ing. Inform.</p> <p>Doble Grado Ing. Inform. y Mat.</p>	<h2>Arquitectura de Computadores (AC)</h2> <h3>Cuaderno de prácticas.</h3> <h3>Bloque Práctico 0. Entorno de programación</h3> <p>Estudiante (nombre y apellidos): Carlos de la Torre</p> <p>Grupo de prácticas: A2</p> <p>Fecha de entrega: 11/03 23:59</p> <p>Fecha evaluación en clase: 19/03</p>	
--	--	---

1. En el primer ejemplo de ejecución en atcgrid usando TORQUE se ejecuta el ejemplo HelloOMP.c de la página 12 del seminario usando la siguiente orden: `echo 'hello/HelloOMP' | qsub -q ac`. El resultado de la ejecución de este código en atcgrid se puede ver en la página 19 del seminario. Conteste a las siguientes preguntas:

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

**RESPUESTA:**

se usa para asignar a la cola de ejecución de AC el trabajo helloOMP

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

**RESPUESTA:**

Se puede mirar la salida del comando `qstat` o bien se puede comprobar si se han generado los ficheros de salida de la ejecución del comando.

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

**RESPUESTA:**

Lo primero que hay que mirar es si tenemos los dos ficheros correspondiente a la ejecución del comando (\*.o y \*.e) y después comprobar si el fichero con extensión \*.e tiene un tamaño 0 para comprobar la falta de errores

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

**RESPUESTA:**

Editando el fichero "nombre\_del\_ejecutable.o\_numero\_ID"

- e. ¿Por qué en el resultado de la ejecución aparecen 24 saludos ";;;Hello World!!!"?

**RESPUESTA:**

Por que el nodo donde se ha ejecutado el comando de helloOMP tiene dos procesadores de 6 cores cada uno, los cuales usan tecnología HyperThreading

2. En el segundo ejemplo de ejecución en atcgrid usando TORQUE el script `script_helloomp.sh` de la página 22 del seminario 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 la página 26 del seminario. Conteste a las siguientes preguntas:

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

**RESPUESTA:**

Por que en la cabecera del script linea (5) tenemos una asignación de bandera -q ac

- b. ¿Cuántas veces ejecuta el script el ejecutable HelloOMP en atcgrid?

**RESPUESTA:**

Lo ejecuta 4 veces.

- c. ¿Cuántos saludos “¡¡¡Hello World!!!” se imprimen en cada ejecución? ¿Por qué se imprime ese número?

**RESPUESTA:**

La primera vez son 12 iteraciones, la segunda vez son 6 iteraciones por que se divide por 2, la tercera vez son 3 iteraciones por que volvemos a dividir por dos y por ultimo como la división es entera una única iteración por que dividimos por 2.

3. 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:**

Una vez que se ha quitado la variable global se sustituye por la carpeta en la que se encuentra alojado el script de esa manera solucionamos el problema de dirección del script. Como lo que se pide es el resultado de la ejecución del programa lo que se hace es añadir el fichero txt con la respuesta del nodo atcgrid1.ugr.es.

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), del PC del aula de prácticas y de su PC (si tiene Linux instalado). Indique qué ha hecho para obtener el contenido de /proc/cpuinfo en atcgrid.

**RESPUESTA:**

Lo mas rápido para conseguir la información es usar el esqueleto del script que ya tenemos y cambiar los comandos de iteración por la petición de la información del fichero cpuinfo, así de esta manera lo único que tenemos que poner dentro del script es “cat /proc/cpuinfo”, se adjunta el fichero donde se ha obtenido la información tanto del nodo como del equipo local el de el aula se ha omitido por que se usa solo el equipo personal para realizar las practicas.

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++:

$$v3 = v1 + v2; \quad v3(i) = v1(i) + v2(i), \quad i=0,...N-1$$

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`.

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 devuelve la función `clock_gettime()`?

**RESPUESTA:**

La variable `ncgt` es el abreviatura de N ClockGetTime y lo que contiene esta variables es el tiempo que tarda el algoritmo en calcular la suma de los vectores.

La función `clock_gettime()` devuelve el valor del reloj del sistema con una precisión de nanosegundos.

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:**

Si se refiere a las diferencias que puede existir en la sintaxis del lenguaje, están claras la primer es que en la reserva de memoria se utiliza `malloc` en C y en C++ no esta, segundo al imprimir los resultados se usa `printf` en C y `cout` en C++ y por ultimo pues a la hora de liberar memoria se usa `free` en C y `delete` en C++.

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:**

```
[A2estudiante7@atcgrid 00_Practica0_Hello]$ echo $(pwd)/SumaVectoresC_vlocales 123456 | qsub -q ac
26672.atcgrid
[A2estudiante7@atcgrid 00_Practica0_Hello]$ qstat
Job id          Name             User              Time Use S Queue
-----
26672.atcgrid   STDIN            A2estudiante7     00:00:00 C ac
[A2estudiante7@atcgrid 00_Practica0_Hello]$ cat STDIN.o26672
Tiempo(seg.):0.001538494
Tamaño Vectores:123456
v1[0]+v2[0]=v3[0] (12345.600000+12345.600000=24691.200000)
v1[123455]+v2[123455]=v3[123455] (24691.100000+0.100000=24691.200000)
[A2estudiante7@atcgrid 00_Practica0_Hello]$
```

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:**

Si por supuesto que se obtiene errores y es debido a la reserva de la memoria para almacenar los vectores que queremos sumar.

8. Generar los ejecutables del código fuente C para vectores globales y para dinámicos. Generar 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:**

No, Por que no se reserva la memoria antes de usarla osea se hace una reserva del espacio virtual pero no se reserva la memoria física de esa manera podemos reservar cuanta memoria se quiera sin pasarnos de la memoria física.

9. Rellenar una tabla 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). Realice otra gráfica con los tiempos obtenidos en el PC local.

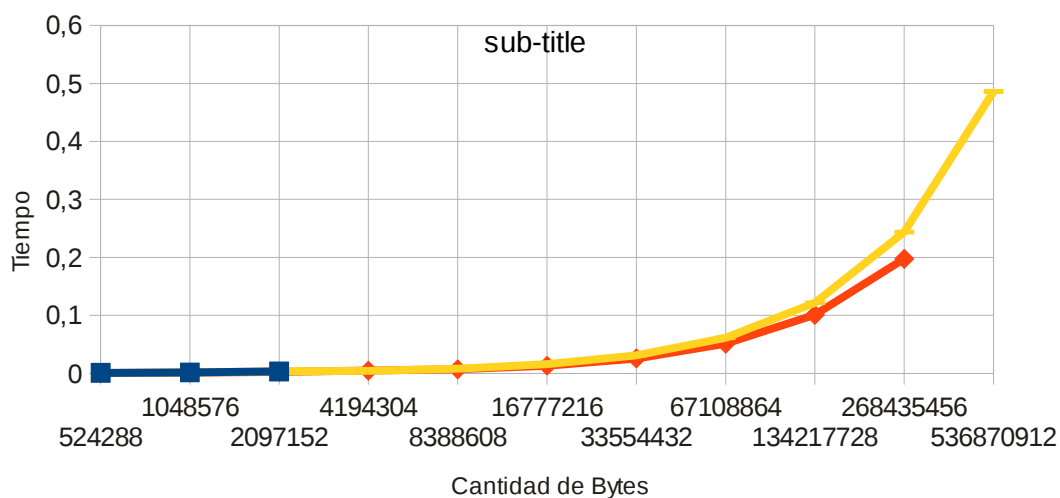
**RESPUESTA:****ATCGRID**

Nº de Componentes	Bytes de Un vector	Tiempo para Vect. locales	Tiempo para Vect. globales	Tiempo para Vect. dinámicos
65536	524288	0,00083258	0,000830786	0,000872552
131072	1048576	0,001647259	0,0010571	0,001714417
262144	2097152	0,003363893	0,002560233	0,003483482
524288	4194304		0,004748451	0,004294527
1048576	8388608		0,006994903	0,00815199
2097152	16777216		0,013098524	0,015767988
4194304	33554432		0,025679007	0,030924471
8388608	67108864		0,051170564	0,061279265
16777216	134217728		0,100797722	0,12169675
33554432	268435456		0,19741886	0,243695821
67108864	536870912			0,486265262

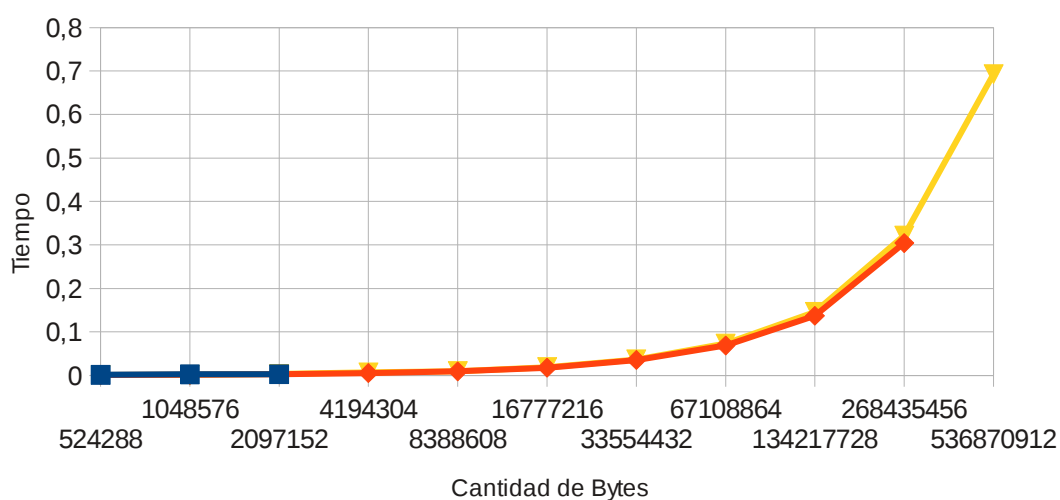
**Ordenador Local**

Nº de Componentes	Bytes de Un vector	Tiempo para Vect. locales	Tiempo para Vect. globales	Tiempo para Vect. dinámicos
65536	524288	0,001246427	0,000862223	0,000653982
131072	1048576	0,002536862	0,001359068	0,00137847
262144	2097152	0,002775158	0,002583738	0,003184129
524288	4194304		0,005316804	0,006877731
1048576	8388608		0,009359646	0,009613991
2097152	16777216		0,017640039	0,018784701
4194304	33554432		0,035429404	0,036739063
8388608	67108864		0,068971878	0,073002241
16777216	134217728		0,137056954	0,14593202
33554432	268435456		0,304472157	0,321600202
67108864	536870912			0,692901786

## Eficiencia Experimental de la suma de Vectores en C



## Eficiencia Experimental de la suma de Vectores en C



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? Razone además por qué el máximo número que se puede almacenar en N es  $2^{32}-1$ .

**RESPUESTA:**

No se puede compilar, no consigo darme cuenta por que este numero es el mayor que puede tener N.