2º curso / 2º cuatr.

Grado Ing. Inform.

Doble Grado Ing. Inform. y Mat.

# Arquitectura de Computadores (AC)

## Cuaderno de prácticas.
## Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos): Francisco Carrillo Pérez
Grupo de prácticas: A2
Fecha de entrega: 10/03/2016
Fecha evaluación en clase: 11/03/2016

---

### Ejercicios basados en los ejemplos del seminario práctico

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 en el seminario. Conteste a las siguientes preguntas:

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

      **RESPUESTA**: Se usa para indicar a que cola se manda el trabajo.

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

      **RESPUESTA**: Se sabe que ha terminado ya que genera los archivos .o y .e .

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

      **RESPUESTA**: Si el archivo .e tiene un tamaño distinto a 0 podemos afirmar que ha habido un error en la ejecución.

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

      **RESPUESTA**: Mirando el archivo .o .

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

      **RESPUESTA**: Porque tenemos 12 cores físicos con Hyperthreading por lo tanto tenemos 12 hebras.

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 la cabecera le indicamos a la cola que queremos mandarlo.

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

      **RESPUESTA**: Lo ejecuta 4 veces ya que en el bucle for se empieza con un valor de 12 y se va reduciendo a la mitad en cada iteración.

   c. ¿Cuántos saludos "¡¡¡Hello World!!!" se imprimen en cada ejecución? (indique el número exacto) ¿Por qué se imprime ese número?
      **RESPUESTA**: Se ejecuta tantas veces como hebras lanzemos. Se imprime ese número de veces ya que 12 es el valor de procesadores que tenemos entonces se pueden realizar de forma paralela.

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**:

Debemos añadir que el directorio es P0 porque si no se obtiene el siguiente error:

```
[A2estudiante4@atcgrid P0]$ cat helloomp.e28615
/var/lib/torque/mom_priv/jobs/28615.atcgrid.SC: line 23: /HelloOMP: No such
file or directory
/var/lib/torque/mom_priv/jobs/28615.atcgrid.SC: line 23: /HelloOMP: No such
file or directory
/var/lib/torque/mom_priv/jobs/28615.atcgrid.SC: line 23: /HelloOMP: No such
file or directory
/var/lib/torque/mom_priv/jobs/28615.atcgrid.SC: line 23: /HelloOMP: No such
file or directory
```

En el caso de que ya añadamos donde se encuentra el ejecutable, el resultado sería el siguiente:

```
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
No de threads inicial: 12
Directorio de trabajo: /home/A2estudiante4/P0

Para 12 threads:
(11:!!!Hello world!!!)(8:!!!Hello world!!!)(0:!!!Hello world!!!)(10:!!!Hello
 world!!!)(3:!!!Hello world!!!)(2:!!!Hello world!!!)(1:!!!Hello world!!!)(7:
!!!Hello world!!!)(6:!!!Hello world!!!)(9:!!!Hello world!!!)(5:!!!Hello worl
d!!!)(4:!!!Hello world!!!)
Para 6 threads:
(0:!!!Hello world!!!)(4:!!!Hello world!!!)(2:!!!Hello world!!!)(1:!!!Hello w
orld!!!)(5:!!!Hello world!!!)(3:!!!Hello world!!!)
Para 3 threads:
(0:!!!Hello world!!!)(2:!!!Hello world!!!)(1:!!!Hello world!!!)
Para 1 threads:
(0:!!!Hello world!!!)[A2estudiante4@atcgrid P0]$
```

## 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**: Con el comando echo 'cat /proc/cpuinfo' | qsub -q ac obtenemos el contenido del /proc/cpuinfo en el atcgrid.

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? Tiene 2 físicos y 4 lógicos.

**RESPUESTA**:

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

**RESPUESTA**: Cada nodo tiene 12 cores físicos y 2 lógicos.

**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**:

La variable ncgt contiene la diferencia del tiempo final (cgt2) menos el tiempo inicial (cgt1) más una correción de error en nanosegundos.

La función clock_gettime() devuelve el tiempo de reloj que hay en ese instante. La devuelve en una estructura timespec y guarda el valor en una variable de ese tipo que se le pasa como segundo argumento.

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.

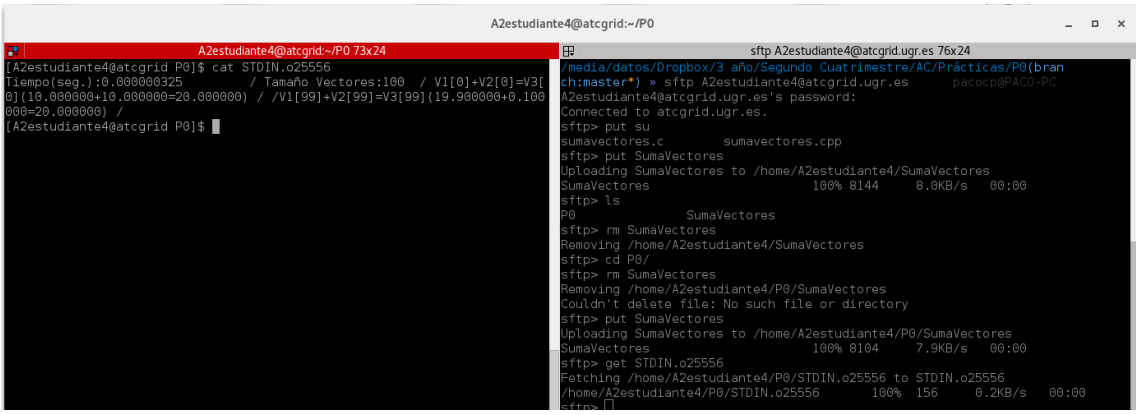
**RESPUESTA**:

| Descripción diferencia | En C | En C++ |
|---|---|---|
| Para el Vector Dynamic,en C se utilizan punteros y malloc | double *v1, *v2, *v3;<br>v1 = (double*) | v1 = new double [N];<br>//si no hay espacio suficiente |

| | | |
|---|---|---|
| para la reserva de espacio de memoria mientras que en C++ se hace con new | malloc(N*sizeof(double));// malloc necesita el tamaño en bytes<br>v2 = (double*) malloc(N*sizeof(double)); //si no hay espacio suficiente malloc devuelve NULL<br>v3 = (double*) malloc(N*sizeof(double)); | new genera una excepción<br>v2 = new double [N];<br>v3 = new double [N]; |
| | | |
| | | |
| | | |
| | | |

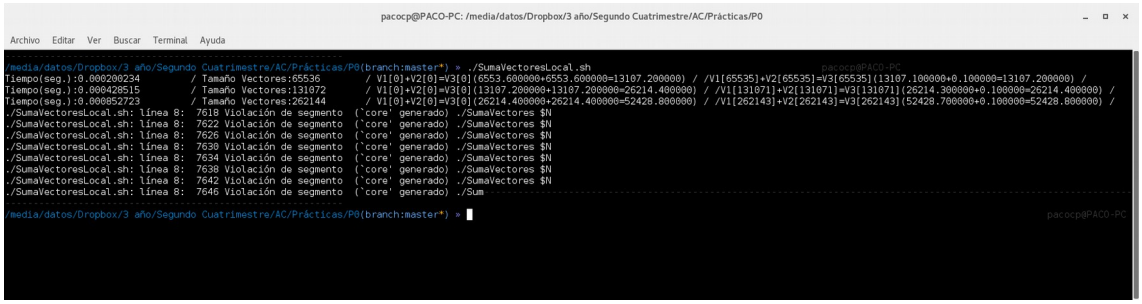
**6.** Generar el ejecutable del código fuente C del Listado 1 para vectores locales (para ello antes de compilar debe descomentar la definición de `VECTOR_LOCAL` y comentar las definiciones de `VECTOR_GLOBAL` y `VECTOR_DYNAMIC`). Ejecutar el código ejecutable resultante en atcgrid usando el la cola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

**RESPUESTA**: Utilizando un tamaño de 100:




**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 produce un error a partir de la 4 iteración.




El error se debe al que a ser un vector local que no reserva memoria dinámica, se produce un error al intentar utilizar ese tamaño en el vector.

**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**: En los vectores globales se obtiene un error y es que se repite un tamaño. Esto es debido a que hemos puesto ese tamaño como el máximo, entonces se queda en 2^25.

**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?

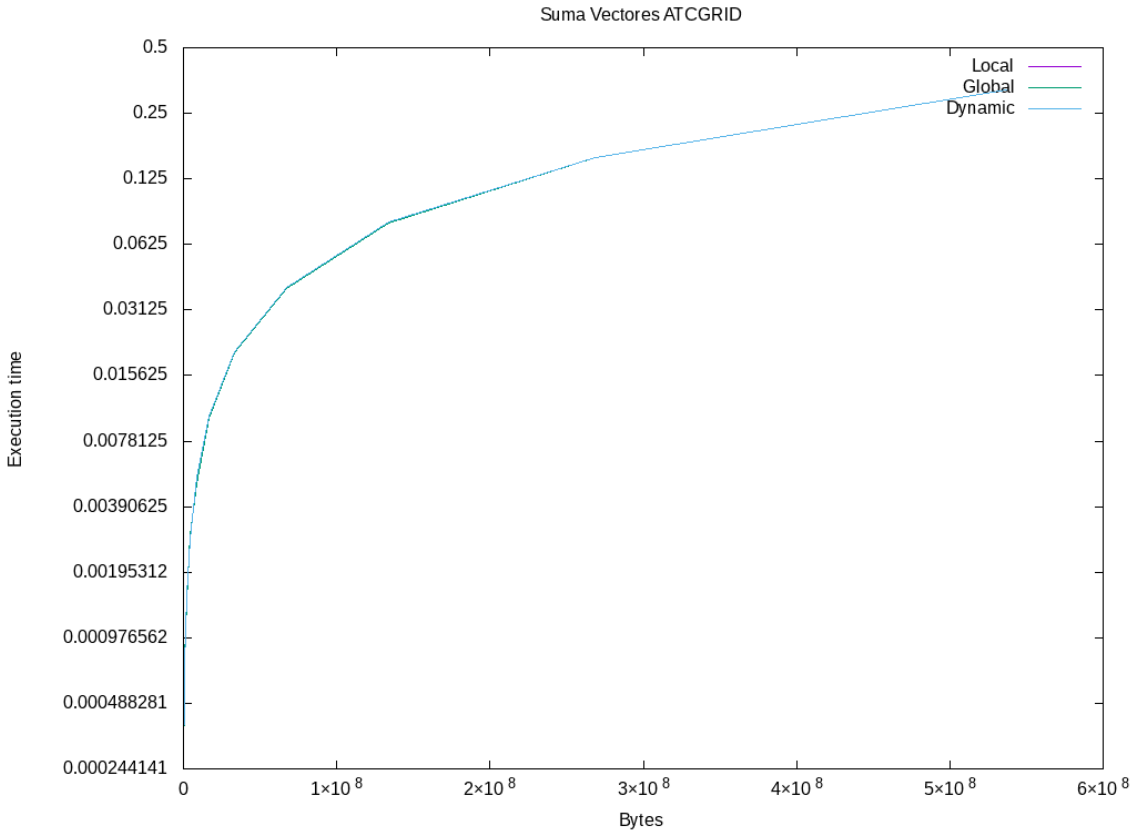
**RESPUESTA**:

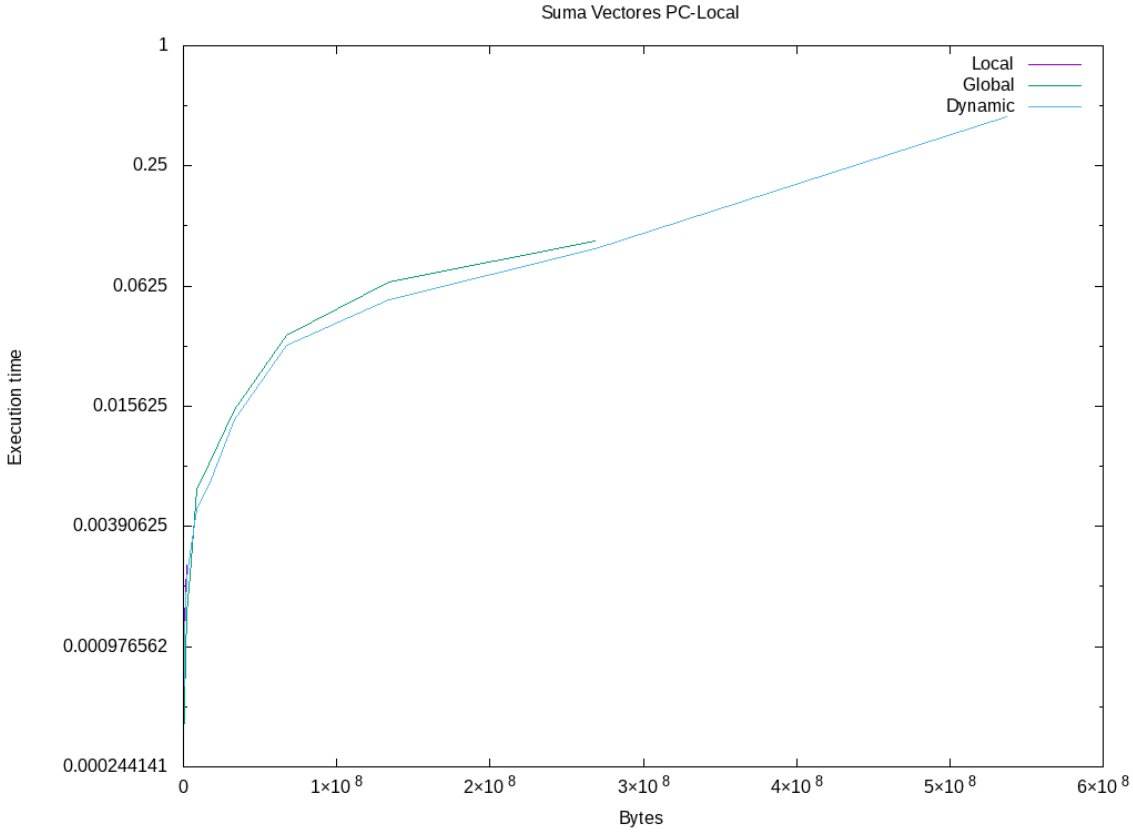
**Tabla 1 .**Tiempos de ejecución de la suma de vectores para vectores locales, globales y dinámicos en ATCGRID

| Nº de Componentes | Bytes de un vector | Tiempo para vect. locales | Tiempo para vect. globales | Tiempo para vect. dinámicos |
|---|---|---|---|---|
| 65536 | 524288 | 0.000384163 | 0.000387690 | 0.000390614 |
| 131072 | 1048576 | 0.000759910 | 0.000780774 | 0.000728937 |
| 262144 | 2097152 | 0.001524225 | 0.001444023 | 0.001540365 |
| 524288 | 4194304 | | 0.002910819 | 0.002768890 |
| 1048576 | 8388608 | | 0.004994734 | 0.005317961 |
| 2097152 | 16777216 | | 0.009959203 | 0.010215546 |
| 4194304 | 33554432 | | 0.019856898 | 0.020054671 |
| 8388608 | 67108864 | | 0.039366770 | 0.039844601 |
| 16777216 | 134217728 | | 0.078793286 | 0.078837624 |
| 33554432 | 268435456 | | 0.156794847 | 0.157102214 |
| 67108864 | 536870912 | | | 0.319227630 |

**Tabla 2 .**Tiempos de ejecución de la suma de vectores para vectores locales, globales y dinámicos en PC Local

| Nº de Componentes | Bytes de un vector | Tiempo para vect. locales | Tiempo para vect. globales | Tiempo para vect. dinámicos |
|---|---|---|---|---|
| 65536 | 524288 | 0.000674045 | 0.000401056 | 0.000621170 |
| 131072 | 1048576 | 0.001300607 | 0.000524065 | 0.001113689 |
| 262144 | 2097152 | 0.002491668 | 0.001041495 | 0.002158730 |
| 524288 | 4194304 | | 0.002236895 | 0.002851950 |
| 1048576 | 8388608 | | 0.006015504 | 0.004779667 |
| 2097152 | 16777216 | | 0.008044922 | 0.006288919 |
| 4194304 | 33554432 | | 0.014961748 | 0.013361869 |
| 8388608 | 67108864 | | 0.035344148 | 0.031605166 |
| 16777216 | 134217728 | | 0.065905154 | 0.053644494 |
| 33554432 | 268435456 | | 0.105181196 | 0.095664025 |
| 67108864 | 536870912 | | | 0.443468808 |


Suma Vectores ATCGRID
Local
Global
Dynamic
Execution time
0.5
0.25
0.125
0.0625
0.03125
0.015625
0.0078125
0.00390625
0.00195312
0.000976562
0.000488281
0.000244141
0
1×10 8
2×10 8
3×10 8
4×10 8
5×10 8
6×10 8
Bytes


Suma Vectores PC-Local
Local
Global
Dynamic
Execution time
1
0.25
0.0625
0.015625
0.00390625
0.000976562
0.000244141
0
1×10 8
2×10 8
3×10 8
4×10 8
5×10 8
6×10 8
Bytes

**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**: Nos da un error de que no puede reubicar la memoria para los vectores. El error es el siguiente: