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): Juan Carlos Ruiz García
Grupo de prácticas: C2
Fecha de entrega: 05/03/2017
Fecha evaluación en clase:

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: La opción -q especifíca a que 'queue/cola' queremos mandar el trabajo a ejecutar.

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

   RESPUESTA: Cuando ejecutamos algo en atcgrid, si usamos el comando 'qstat' nos muestra el estado de la ejecución del trabajo enviado. Si nos fijamos en la columna marcada con 'S' y aparece una 'C' en dicha columna, sabremos que la ejecución del trabajo se encuentra en estado 'Complete'.

```
[C2estudiante17@atcgrid ~]$ echo './HelloOMP' | qsub -q ac
42892.atcgrid
[C2estudiante17@atcgrid ~]$ qstat
Job ID                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
42892.atcgrid              STDIN            C2estudiante17  00:00:00 C ac
```

```
-  the job state:
      C -     Job is completed after having run/
      E -  Job is exiting after having run.
```

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

   RESPUESTA: Cuando termina la ejecución de un trabajo de la cola AC, se generan dos ficheros, uno con la extensión .e y otro con la extensión .o. Si el fichero con la extensión .e está vacío quiere decir que no ha ocurrido ningun error durante la ejecución del programa, en caso contrario el error se encontrará dentro de dicho fichero.

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

   RESPUESTA: En el interior del fichero con la extensión .o.

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

   RESPUESTA: Porque se nos han asignado los 24 nucleos lógicos de uno de los nodos atcgrid.

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: Dentro del script hay una linea 'PBS -q ac' que especifica la cola.

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

RESPUESTA: Lo ejecuta un total de cuatro veces. Es un bucle que va desde 12 a 1 dividiendo entre 2 (12, 6, 3, 1)

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

RESPUESTA: En la primera ejecución imprime 12, en la segunda 6, en la tercera 3 y en la ultima 1. Esto es debido a que lanza la ejecución del programa con 12, 6, 3 y 1 hebras correspondientemente.

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:

```
Para 12 threads:

/home/C2estudiante17

Para 6 threads:

/home/C2estudiante17

Para 3 threads:

/home/C2estudiante17

Para 1 threads:

/home/C2estudiante17
```

```
#Se ejecuta HelloOMP, que está en el directorio en el que se ha ejecutado qsub
for ((P=OMP_THREAD_LIMIT;P>0;P=P/2))
do
export OMP_NUM_THREADS=$P
echo -e "\nPara $OMP_NUM_THREADS threads:"
echo -e "\n$PBS_O_WORKDIR"
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: Para obtener el contenido de /proc/cpuinfo en atcgrid he utilizado el comando "echo 'cat /proc/cpuinfo' | qsub -q ac".

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? Mi equipo cuenta con 1 chip de procesamiento, con 2 cores físicos y 2 cores logicos cada uno de ellos, lo que forma un total de 4 nucleos lógicos.

RESPUESTA:

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

RESPUESTA: Un nodo de atcgrid cuenta con 2 chips de procesamientos, cada uno de ellos con 6 cores físicos, con 2 cores lógicos cada uno de estos 6, lo que suma un total de 24 nucleos logicos.

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 es de tipo double y contiene el tiempo de ejecución que se tarda en sumar los vectores.

Lo que devuelve realmente es un int que valdrá 0 en caso de devolver éxito y -1 en caso de error, pero el segundo argumento que se le pasa (es el contenedor) es un struct del tipo:

```
struct timespec {
      time_t   tv_sec;        /* seconds */
      long     tv_nsec;       /* nanoseconds */
};
```

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++
Reserva espacio de vectores	Se utiliza malloc	Se utiliza new
Salida estandar	Se utiliza printf	Se utiliza cout
Liberar reserva de espacio	Se utiliza free	Se utiliza delete

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:

```
[C2estudiante17@atcgrid ~]$ echo './SumaVectores 100' | qsub -q ac
43949.atcgrid
[C2estudiante17@atcgrid ~]$ ls
HelloOMP                        script_helloomp.sh  STDIN.o43949
script_helloomp_modificado.sh  STDIN.e43949        SumaVectores
[C2estudiante17@atcgrid ~]$ cat STDIN.o43949
Tiempo(seg.):0.000000328         Tamaño Vectores:100     V1[0]+V2[0]=V3[0](10.00000
0+10.000000=20.000000) V1[99]+V2[99]=V3[99](19.900000+0.100000=20.000000)
```

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:

Ejecutamos el .sh y obtenemos error debido a que desbordamos la pila.

```
[C2estudiante17@atcgrid ~]$ qsub SumaVectores.sh
43953.atcgrid
[C2estudiante17@atcgrid ~]$ qstat
Job ID                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
43953.atcgrid              ...resC_vlocales C2estudiante17  00:00:00 E ac

[C2estudiante17@atcgrid ~]$ ls
HelloOMP                        SumaVectores                    SumaVectores.sh
script_helloomp_modificado.sh  SumaVectoresC_vlocales.e43953
script_helloomp.sh              SumaVectoresC_vlocales.o43953
```

Si nos fijamos mas detalladamente en el fichero .o generado, veremos que a partir del valor 262144 ya no se muestran las siguientes salidas. Esto indica que hemos desbordado la pila.

```
Tiempo(seg.):0.000385803   Tamaño Vectores:65536   V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) V1[65535
Tiempo(seg.):0.000763983   Tamaño Vectores:131072  V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.400000) V1[131
Tiempo(seg.):0.001523867   Tamaño Vectores:262144  V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.800000) V1[262
```

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:

PC LOCAL

Variables Globales:

```
juanka1995@juanka-laptop ~/practicas/AC/Seminarios/practica 0/SumaVectores/C/Global $ ./SumaVectoresGlobalLocal.sh
Tiempo(seg.):0.000483694        Tamaño Vectores:65536   V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) V1[65535]+V2[65535]=V3[65535](13107.100000+0.100000=1310
7.200000)
Tiempo(seg.):0.000664775        Tamaño Vectores:131072  V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.400000) V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000
=26214.400000)
Tiempo(seg.):0.001340829        Tamaño Vectores:262144  V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.800000) V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000
=52428.800000)
Tiempo(seg.):0.002535176        Tamaño Vectores:524288  V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000) V1[524287]+V2[524287]=V3[524287](104857.500000+0.1000
00=104857.600000)
Tiempo(seg.):0.004520274        Tamaño Vectores:1048576         V1[0]+V2[0]=V3[0](104857.600000+104857.600000=209715.200000) V1[1048575]+V2[1048575]=V3[1048575](209715.
100000+0.100000=209715.200000)
Tiempo(seg.):0.009199184        Tamaño Vectores:2097152         V1[0]+V2[0]=V3[0](209715.200000+209715.200000=419430.400000) V1[2097151]+V2[2097151]=V3[2097151](419430.
300000+0.100000=419430.400000)
Tiempo(seg.):0.017693867        Tamaño Vectores:4194304         V1[0]+V2[0]=V3[0](419430.400000+419430.400000=838860.800000) V1[4194303]+V2[4194303]=V3[4194303](838860.
700000+0.100000=838860.800000)
Tiempo(seg.):0.032831887        Tamaño Vectores:8388608         V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1677721.600000) V1[8388607]+V2[8388607]=V3[8388607](167772
1.500000+0.100000=1677721.600000)
Tiempo(seg.):0.064946460        Tamaño Vectores:16777216        V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.200000) V1[16777215]+V2[16777215]=V3[16777215](3
355443.100000+0.100000=3355443.200000)
Tiempo(seg.):0.136066873        Tamaño Vectores:33554432        V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) V1[33554431]+V2[33554431]=V3[33554431](6
710886.300000+0.100000=6710886.400000)
Tiempo(seg.):0.126284688        Tamaño Vectores:33554432        V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) V1[33554431]+V2[33554431]=V3[33554431](6
710886.300000+0.100000=6710886.400000)
juanka1995@juanka-laptop ~/practicas/AC/Seminarios/practica 0/SumaVectores/C/Global $
```

Variables Dinamicas:

```
juanka1995@juanka-laptop ~/practicas/AC/Seminarios/practica 0/SumaVectores/C/Dynamic $ ./SumaVectoresDynamicLocal.sh
Tiempo(seg.):0.001113603        Tamaño Vectores:65536   V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) V1[65535]+V2[65535]=V3[65535](13107.100000+0.100000=1310
7.200000)
Tiempo(seg.):0.001209653        Tamaño Vectores:131072  V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.400000) V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000
=26214.400000)
Tiempo(seg.):0.001937153        Tamaño Vectores:262144  V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.800000) V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000
=52428.800000)
Tiempo(seg.):0.002763249        Tamaño Vectores:524288  V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000) V1[524287]+V2[524287]=V3[524287](104857.500000+0.1000
00=104857.600000)
Tiempo(seg.):0.005759123        Tamaño Vectores:1048576         V1[0]+V2[0]=V3[0](104857.600000+104857.600000=209715.200000) V1[1048575]+V2[1048575]=V3[1048575](209715.
100000+0.100000=209715.200000)
Tiempo(seg.):0.008060876        Tamaño Vectores:2097152         V1[0]+V2[0]=V3[0](209715.200000+209715.200000=419430.400000) V1[2097151]+V2[2097151]=V3[2097151](419430.
300000+0.100000=419430.400000)
Tiempo(seg.):0.016005531        Tamaño Vectores:4194304         V1[0]+V2[0]=V3[0](419430.400000+419430.400000=838860.800000) V1[4194303]+V2[4194303]=V3[4194303](838860.
700000+0.100000=838860.800000)
Tiempo(seg.):0.031021111        Tamaño Vectores:8388608         V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1677721.600000) V1[8388607]+V2[8388607]=V3[8388607](167772
1.500000+0.100000=1677721.600000)
Tiempo(seg.):0.060135289        Tamaño Vectores:16777216        V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.200000) V1[16777215]+V2[16777215]=V3[16777215](3
355443.100000+0.100000=3355443.200000)
Tiempo(seg.):0.121791766        Tamaño Vectores:33554432        V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) V1[33554431]+V2[33554431]=V3[33554431](6
710886.300000+0.100000=6710886.400000)
Tiempo(seg.):0.236192868        Tamaño Vectores:67108864        V1[0]+V2[0]=V3[0](6710886.400000+6710886.400000=13421772.800000) V1[67108863]+V2[67108863]=V3[67108863](
13421772.700000+0.100000=13421772.800000)
```

ATCGRID

Variables Globales:

```
Tiempo(seg.):0.000286567        Tamaño Vectores:65536   V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) V1[65535]+V2[65535]=V3[65535](13107.100000
+0.100000=13107.200000)
Tiempo(seg.):0.000804386        Tamaño Vectores:131072  V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.400000) V1[131071]+V2[131071]=V3[131071](26214.300000
+0.100000=26214.400000)
Tiempo(seg.):0.001547647        Tamaño Vectores:262144  V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.800000) V1[262143]+V2[262143]=V3[262143](52428.700000
+0.100000=52428.800000)
Tiempo(seg.):0.002778879        Tamaño Vectores:524288  V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000) V1[524287]+V2[524287]=V3[524287](104857.500000
+0.100000=104857.600000)
Tiempo(seg.):0.005065095        Tamaño Vectores:1048576         V1[0]+V2[0]=V3[0](104857.600000+104857.600000=209715.200000) V1[1048575]+V2[1048575]=V3[1048575]
(209715.100000+0.100000=209715.200000)
Tiempo(seg.):0.010061922        Tamaño Vectores:2097152         V1[0]+V2[0]=V3[0](209715.200000+209715.200000=419430.400000) V1[2097151]+V2[2097151]=V3[2097151]
(419430.300000+0.100000=419430.400000)
Tiempo(seg.):0.019954980        Tamaño Vectores:4194304         V1[0]+V2[0]=V3[0](419430.400000+419430.400000=838860.800000) V1[4194303]+V2[4194303]=V3[4194303]
(838860.700000+0.100000=838860.800000)
Tiempo(seg.):0.039712216        Tamaño Vectores:8388608         V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1677721.600000) V1[8388607]+V2[8388607]=V3[8388607]
(1677721.500000+0.100000=1677721.600000)
Tiempo(seg.):0.078091319        Tamaño Vectores:16777216        V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.200000) V1[16777215]+V2[16777215]=V3[16777215]
(3355443.100000+0.100000=3355443.200000)
Tiempo(seg.):0.157049801        Tamaño Vectores:33554432        V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) V1[33554431]+V2[33554431]=V3[33554431]
(6710886.300000+0.100000=6710886.400000)
Tiempo(seg.):0.156727475        Tamaño Vectores:33554432        V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) V1[33554431]+V2[33554431]=V3[33554431]
(6710886.300000+0.100000=6710886.400000)
```

Variables Dinámicas:

```
Tiempo(seg.):0.000381345        Tamaño Vectores:65536   V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) V1[65535]+V2[65535]=V3[65535](13107.100000
+0.100000=13107.200000)
Tiempo(seg.):0.000770262        Tamaño Vectores:131072  V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.400000) V1[131071]+V2[131071]=V3[131071](26214.300000
+0.100000=26214.400000)
Tiempo(seg.):0.001524485        Tamaño Vectores:262144  V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.800000) V1[262143]+V2[262143]=V3[262143](52428.700000
+0.100000=52428.800000)
Tiempo(seg.):0.003118821        Tamaño Vectores:524288  V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000) V1[524287]+V2[524287]=V3[524287](104857.500000
+0.100000=104857.600000)
Tiempo(seg.):0.005124161        Tamaño Vectores:1048576         V1[0]+V2[0]=V3[0](104857.600000+104857.600000=209715.200000) V1[1048575]+V2[1048575]=V3[1048575]
(209715.100000+0.100000=209715.200000)
Tiempo(seg.):0.009932724        Tamaño Vectores:2097152         V1[0]+V2[0]=V3[0](209715.200000+209715.200000=419430.400000) V1[2097151]+V2[2097151]=V3[2097151]
(419430.300000+0.100000=419430.400000)
Tiempo(seg.):0.019673438        Tamaño Vectores:4194304         V1[0]+V2[0]=V3[0](419430.400000+419430.400000=838860.800000) V1[4194303]+V2[4194303]=V3[4194303]
(838860.700000+0.100000=838860.800000)
Tiempo(seg.):0.039542189        Tamaño Vectores:8388608         V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1677721.600000) V1[8388607]+V2[8388607]=V3[8388607]
(1677721.500000+0.100000=1677721.600000)
Tiempo(seg.):0.078714328        Tamaño Vectores:16777216        V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.200000) V1[16777215]+V2[16777215]=V3[16777215]
(3355443.100000+0.100000=3355443.200000)
Tiempo(seg.):0.157716293        Tamaño Vectores:33554432        V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) V1[33554431]+V2[33554431]=V3[33554431]
(6710886.300000+0.100000=6710886.400000)
Tiempo(seg.):0.316359115        Tamaño Vectores:67108864        V1[0]+V2[0]=V3[0](6710886.400000+6710886.400000=13421772.800000) V1[67108863]+V2[67108863]=V3
[67108863](13421772.700000+0.100000=13421772.800000)
```

Como podemos ver ahora no ocurre ningun tipo de error. Esto es debido a que ahora las variables no se almacenan en la pila, si no en el heap o en la zona de datos del programa.

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:

ATCGRID

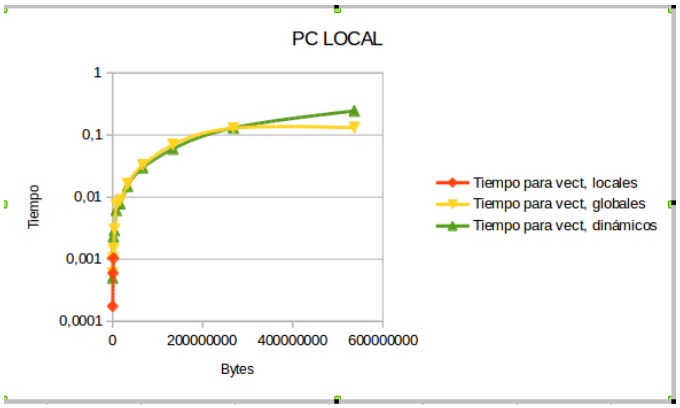
Tabla 1 . Tiempos de ejecución de la suma de vectores para vectores locales, globales y dinámicos

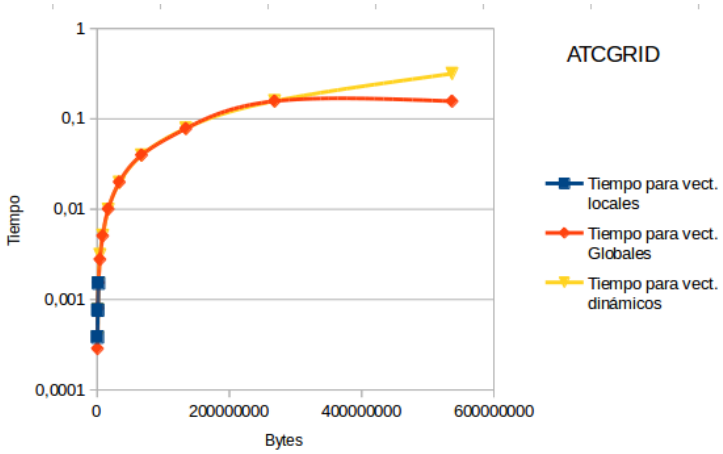
Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000385803	0.000286567	0.000381345
131072	1048576	0.000763983	0.000804386	0.000770262
262144	2097152	0.001523867	0.001547647	0.001524485
524288	4194304	-	0.002778879	0.003118821
1048576	8388608	-	0.005065095	0.005124161
2097152	16777216	-	0.010061922	0.009932724
4194304	33554432	-	0.019954980	0.019673438
8388608	67108864	-	0.039712216	0.039542189
16777216	134217728	-	0.078091319	0.078714328
33554432	268435456	-	0.157049801	0.157716293
67108864	536870912	-	0.156727475	0.316359115

PC LOCAL

Tabla 2 . Tiempos de ejecución de la suma de vectores para vectores locales, globales y dinámicos

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000173914	0.000597002	0.000499794
131072	1048576	0.000591345	0.001030828	0.001241893
262144	2097152	0.001030832	0.001440173	0.002310970
524288	4194304	-	0.003025072	0.002905682
1048576	8388608	-	0.007505884	0.006161948
2097152	16777216	-	0.008830261	0.007859638
4194304	33554432	-	0.016285385	0.015012666
8388608	67108864	-	0.032872364	0.029822471
16777216	134217728	-	0.069600036	0.059615740
33554432	268435456	-	0.128501006	0.130120098
67108864	536870912	-	0.131029924	0.244502163





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:

Lo que sucede es que estamos haciendo una reserva de memoria superior a 2GB consecutivos ya que reservar un vector de doubles con 2^32 – 1 celdas equivale a unos 32 GB aproximadamente. En el siguiente https://software.intel.com/en-us/articles/avoiding-relocation-errors-when-building-applications-with-large-global-or-static-data-on-intel64/ viene la explicación detallada de lo comentado anteriormente.

El máximo numero que se puede almacenar en N es 2^32 -1 debido a que N es un integer el cual ocupa 32bits. Esto quiere decir que el maximo numero que se puede representar con 32 bits es el 2^32 – 1.

```
juanka1995@juanka-laptop ~/practicas/AC/Practicas/practica_0/SumaVectores/C $ gcc -O2 SumaVectoresModificado.c -o Global/SumaVectoresGlobalModificado -lrt
/tmp/cc3I4QDU.o: In function `main':
SumaVectoresModificado.c:(.text.startup+0x79): relocation truncated to fit: R_X86_64_32S against symbol `v2' defined in COMMON section in /tmp/cc3I4QDU.o
SumaVectoresModificado.c:(.text.startup+0xc0): relocation truncated to fit: R_X86_64_32S against symbol `v2' defined in COMMON section in /tmp/cc3I4QDU.o
SumaVectoresModificado.c:(.text.startup+0xc8): relocation truncated to fit: R_X86_64_32S against symbol `v3' defined in COMMON section in /tmp/cc3I4QDU.o
SumaVectoresModificado.c:(.text.startup+0xfc): relocation truncated to fit: R_X86_64_32S against symbol `v3' defined in COMMON section in /tmp/cc3I4QDU.o
SumaVectoresModificado.c:(.text.startup+0x115): relocation truncated to fit: R_X86_64_32S against symbol `v2' defined in COMMON section in /tmp/cc3I4QDU.o
SumaVectoresModificado.c:(.text.startup+0x12b): relocation truncated to fit: R_X86_64_PC32 against symbol `v3' defined in COMMON section in /tmp/cc3I4QDU.o
SumaVectoresModificado.c:(.text.startup+0x135): relocation truncated to fit: R_X86_64_PC32 against symbol `v2' defined in COMMON section in /tmp/cc3I4QDU.o
collect2: error: ld returned 1 exit status
```