

2° curso / 2° 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): Daniel Bolaños Martínez

Grupo de prácticas: A1

Fecha de entrega: 09/03/2017

Fecha evaluación en clase: 10/03/2017

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 ver en el seminario. Conteste a las siguientes preguntas:

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

RESPUESTA:

Se usa para indicar la cola destino donde se ejecutará nuestro trabajo.

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

RESPUESTA:

Al usar la orden `qstat` en el apartado S (State) de la cola, vendrá determinado una C de (Completed). Además si termina la ejecución se habrán generado los archivos `".o"` y `".e"` correspondientes.

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

RESPUESTA:

Consultando el archivo con extensión `".e"`, el cual contiene los errores de ejecución de nuestro trabajo y viendo si no está vacío.

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

RESPUESTA:

Mostrando el contenido del archivo con extensión `".o"` que se crea después de la ejecución del trabajo y contiene el resultado de la misma.

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

RESPUESTA:

Porque el cluster `atcgrid` cuenta con dos procesadores de 6 cores/12 threads cada uno y por cada thread se muestra un mensaje.

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 la elección de la cola ya viene definida en el script con la orden `#PBS -q ac`.

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

RESPUESTA:

Lo ejecuta 4 veces, para 12, 6, 3 y 1 threads respectivamente en cada caso.

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

RESPUESTA:

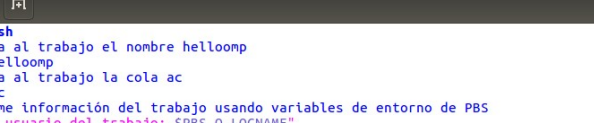
Para 12, 6, 3 y 1 thread/s, 12, 6, 3 y 1 veces respectivamente.

Cada thread, imprime su propio saludo, por lo que hay uno por cada thread existente.

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.



```
script_helloomp.sh (~/.Escritorio/2anio/AC/hello/) - gedit
Abrir ▾  Guardar

#!/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 PBS
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 "Directorio de trabajo:" $PBS_O_WORKDIR
cat $PBS_NODEFILE #Se fija a 12 el no de threads máximo (tantos como cores en un nodo)
export OMP_THREAD_LIMIT=12
echo "No de threads inicial: $OMP_THREAD_LIMIT"
#Se ejecuta HelloOMP, que está en el directorio en el que se ha ejecutado qsub
for ((P=OMP_THREAD_LIMIT;P>0;P=P/2))
do
export OMP_NUM_THREADS=$P
echo -e "\nPara $OMP_NUM_THREADS threads:"
./HelloOMP
done
```

```
danielbolanos@HP-Notebook: ~/Escritorio/2anio/AC/hello
5
Id. usuario del trabajo: Eiestudiante20
Id. del trabajo: 41735.atcgrid
Nombre del trabajo especificado por usuario: helloomp
Nodo que ejecuta qsub: atcgrid
Cola: ac
Nodos asignados al trabajo:
Directorio de trabajo: /home/Eiestudiante20/hello
atcgrid1
atcgrid1
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:
```

RESPUESTA:

Aparece el directorio de trabajo desde el que se ejecuta el script, pero no hay resultados de la ejecución del propio programa ya que al borrar la variable `$PBS_O_WORKDIR`, no es capaz de encontrar el ejecutable `HelloOMP`.

Añadimos la variable de entorno con la orden:

echo "Directorio trabajo:" \$PBS_0 WORKDIR en el script.

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:

Podemos obtener esa información ejecutando la siguiente orden en el terminal ssh:

```
$ 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?

RESPUESTA:

Mi PC tiene 4 processor en su cpuinfo, lo que equivale a 4 cores lógicos.

Para calcular los cores físicos, nos fijamos en el número de valores diferentes tanto del apartado core_id, que determina los cores físicos, como el de physical_id que determina los sockets.

En este caso, hay 2 cores físicos y 1 socket, lo que hacen un total de 2 cores físicos.

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

RESPUESTA:

El nodo de atcgrid tiene 24 processor en su cpu info, lo que equivale a 24 cores lógicos.

De la misma forma, tenemos en este caso, 6 cores físicos (6 valores distintos de core_id) y 2 sockets (2 valores distintos de physical_id), lo que hacen un total de 24 cores físicos por nodo atcgrid.

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 de tiempo entre el momento de inicio (almacenado en `cgt1`) y finalización (almacenado en `cgt2`) del bucle que realiza la suma de vectores, de esta forma, sabremos el tiempo que se tarda en realizar esta tarea.

La función `clock_gettime()` devuelve un valor en un instante de tiempo y almacena en una estructura los segundos y nanosegundos obtenidos en ese momento preciso.

La estructura que utiliza la función denominada `timespec` viene definida en la biblioteca `time.h`:

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

Información obtenida de: https://linux.die.net/man/3/clock_gettime

- 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++
Imprimir por pantalla	<code>printf()</code>	<code>cout <<</code>
Liberar espacio	<code>free()</code>	<code>delete</code>
Espacio de nombres	no	si
Reserva de espacio	<code>malloc()</code>	<code>new</code>
Nombre bibliotecas	<code>stdlib.h</code>	<code>cstdlib</code>

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:

Subimos a través del terminal `sftp` el ejecutable `SumaVectoresC` al front-end con la orden:

```
$ put SumaVectoresC
```

Una vez allí, ejecutamos el programa con el siguiente comando en el terminal `ssh`:

```
$ echo "hello/SumaVectoresC 5" | qsub -q ac (5 o la longitud que deseemos)
```

Obtenemos los archivos “.o” y “.e”, si la ejecución ha sido satisfactoria, mostramos el archivo “.o” para ver los resultados de la ejecución.

```
Eiestudiante20@atcgrid:~/hello
43919.atcgrid          STDIN          Eiestudiante20  00:00:00
C ac
[Eiestudiante20@atcgrid hello]$ cat STDIN.o*
Faltan no componentes del vector
[Eiestudiante20@atcgrid hello]$ rm STDIN*
[Eiestudiante20@atcgrid hello]$ echo "hello/SumaVectoresC 5" | qsub -
q ac
43920.atcgrid
[Eiestudiante20@atcgrid hello]$ qstat
Job ID          Name          User          Time Use S
Queue
-----
43920.atcgrid          STDIN          Eiestudiante20  00:00:00
C ac
[Eiestudiante20@atcgrid hello]$ cat STDIN.o*
Tiempo(seg.):0.000000165          / Tamaño Vectores:5          / V1[0]+V2[0]
=V3[0](0.500000+0.500000=1.000000) / / V1[4]+V2[4]=V3[4](0.900000+0.1
00000=1.000000) /
[Eiestudiante20@atcgrid hello]$
```

Captura Terminal ssh.

```
danibolano@Aspire-E5-575G: ~
sftp> ll
hello
sftp> lcd hello
sftp> ll
Capturas          helloomp.o41725          STDIN.o41724
HelloOMP          helloomp.o41735          STDIN.o43676
HelloOMP.c        script_helloomp.sh        SumaVectoresC
sftp> ls
hello
sftp> cd hello
sftp> put Suma*
Uploading SumaVectoresC to /home/Eiestudiante20/hello/SumaVectoresC
SumaVectoresC          100% 8888          8.7KB/s          00:00
sftp> ls
HelloOMP          SumaVectoresC          helloomp.e41735
helloomp.o41735          script_helloomp.sh
sftp>
```

Captura Terminal sftp.

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:

Ejecutando el script desde la terminal ssh con la orden:

```
$ qsub script_vectores.sh
```

Podemos ver que en el fichero generado con el nombre SumaVectoresC_vlocales.o43927, obtenemos la ejecución del programa para distintas longitudes de vectores.

Si accedemos ahora al archivo SumaVectoresC_vlocales.e43927, vemos que se producen varios errores por segmentation fault o violación de segmento, debido a que desbordamos el tamaño de la pila con la longitud de los vectores.

Tal y como está definido el script, se va calculando los resultados de la ejecución de la suma de vectores multiplicando la longitud por 2, teniendo en cuenta que la última ejecución realizada con éxito es 262144, la siguiente sería 524288 por lo que resulta evidente que se haya producido desbordamiento a partir de ese número para las siguientes ejecuciones.

Si lo hacemos desde mi PC local, obtenemos que la cifra aproximada a partir de la que desborda el programa es la longitud 350000.

```
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
Tiempo(seg.):0.000387388          / 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.):0.000758770          / 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.):0.001444227          / 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) /
```

```
[Eiestudiante20@atcgrid hello]$ cat SumaVectoresC_vlocales.e43927
/var/lib/torque/mom_priv/jobs/43927.atcgrid.SC: line 20: 22431 Segmen
tation fault (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
/var/lib/torque/mom_priv/jobs/43927.atcgrid.SC: line 20: 22434 Segmen
tation fault (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
/var/lib/torque/mom_priv/jobs/43927.atcgrid.SC: line 20: 22441 Segmen
tation fault (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
/var/lib/torque/mom_priv/jobs/43927.atcgrid.SC: line 20: 22447 Segmen
tation fault (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
/var/lib/torque/mom_priv/jobs/43927.atcgrid.SC: line 20: 22454 Segmen
tation fault (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
/var/lib/torque/mom_priv/jobs/43927.atcgrid.SC: line 20: 22460 Segmen
tation fault (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
/var/lib/torque/mom_priv/jobs/43927.atcgrid.SC: line 20: 22466 Segmen
tation fault (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
/var/lib/torque/mom_priv/jobs/43927.atcgrid.SC: line 20: 22473 Segmen
tation fault (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
```



```

Violación de segmento ('core' generado)
danibolanos@Aspire-E5-575G:~/Escritorio$ ./SumaVectoresC 350000
Violación de segmento ('core' generado)
danibolanos@Aspire-E5-575G:~/Escritorio$ ./SumaVectoresC 330000
Tiempo(seg.):0.001180261 / Tamaño Vectores:330000 / V1[0]
+V2[0]=V3[0](33000.000000+33000.000000=66000.000000) / / V1[329999]+V2[
329999]=V3[329999](65999.900000+0.100000=66000.000000) /
danibolanos@Aspire-E5-575G:~/Escritorio$

```

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:

Usando vectores globales y dinámicos no se producen errores ni segmentation fault ni en el nodo de atcgrid ni en el PC local; la razón por la que ocurre esto, es evidente, al contrario que las variables locales, las cuales se almacenan en pila, las variables globales y dinámicas, se almacenan en la zona de datos del programa o en el heap.

```

Tiempo(seg.):0.020670902 / Tamaño Vectores:4194304 / V1[
0]+V2[0]=V3[0](419430.400000+419430.400000=838860.800000) / / V1[4194
303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tiempo(seg.):0.040632605 / Tamaño Vectores:8388608 / V1[
0]+V2[0]=V3[0](838860.800000+838860.800000=1677721.600000) / / V1[838
8607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000)
/
Tiempo(seg.):0.081905559 / Tamaño Vectores:16777216 / V1[
0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.200000) / / V1[1
6777215]+V2[16777215]=V3[16777215](3355443.100000+0.100000=3355443.20
0000) /
Tiempo(seg.):0.164116842 / Tamaño Vectores:33554432 / V1[
0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) / / V1[3
3554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.40
0000) /
Tiempo(seg.):0.159483822 / Tamaño Vectores:33554432 / V1[
0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) / / V1[3
3554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.40
0000) /
[Eiestudiante20@atcgrid hello]$

```

Ejecución Vectores Globales.

```

0]+V2[0]=V3[0](419430.400000+419430.400000=838860.800000) / / V1[4194
303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tiempo(seg.):0.039782049 / Tamaño Vectores:8388608 / V1[
0]+V2[0]=V3[0](838860.800000+838860.800000=1677721.600000) / / V1[838
8607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000)
/
Tiempo(seg.):0.079480929 / Tamaño Vectores:16777216 / V1[
0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.200000) / / V1[1
6777215]+V2[16777215]=V3[16777215](3355443.100000+0.100000=3355443.20
0000) /
Tiempo(seg.):0.157911866 / Tamaño Vectores:33554432 / V1[
0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) / / V1[3
3554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.40
0000) /
Tiempo(seg.):0.315517922 / 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) /
[Eiestudiante20@atcgrid hello]$ cat SumaVectoresC vdinamicos.e*

```

Ejecución Vectores Dinámicos.

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:

La diferencia entre los tiempos de ejecución con vectores de los tres tipos es muy pequeña, pocos nanosegundos para igual número de componentes, tardando más los vectores dinámicos y menos los locales.

Respecto a la comparación entre ordenador local y atcgrid, mi PC, tiene unos valores más bajos de tiempo de ejecución que el nodo atcgrid.

Nota: Eliminamos en ambos casos el último valor de los vectores globales, porque al tener un máximo de componentes y sobrepasarlo, en realidad lo que hace es adjudicarle el tiempo del anterior caso válido.

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

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000387388	0.000383930	0.000400243
131072	1048576	0.000758770	0.000787542	0.000746246
262144	2097152	0.001444227	0.001532891	0.001541253
524288	4194304	-	0.003169870	0.002945698
1048576	8388608	-	0.005358598	0.005308132
2097152	16777216	-	0.010534070	0.010073553
4194304	33554432	-	0.020670902	0.020021283
8388608	67108864	-	0.040632605	0.039782049
16777216	134217728	-	0.081905559	0.079480929
33554432	268435456	-	0.164116842	0.157911866
67108864	536870912	-	-	0.315517922

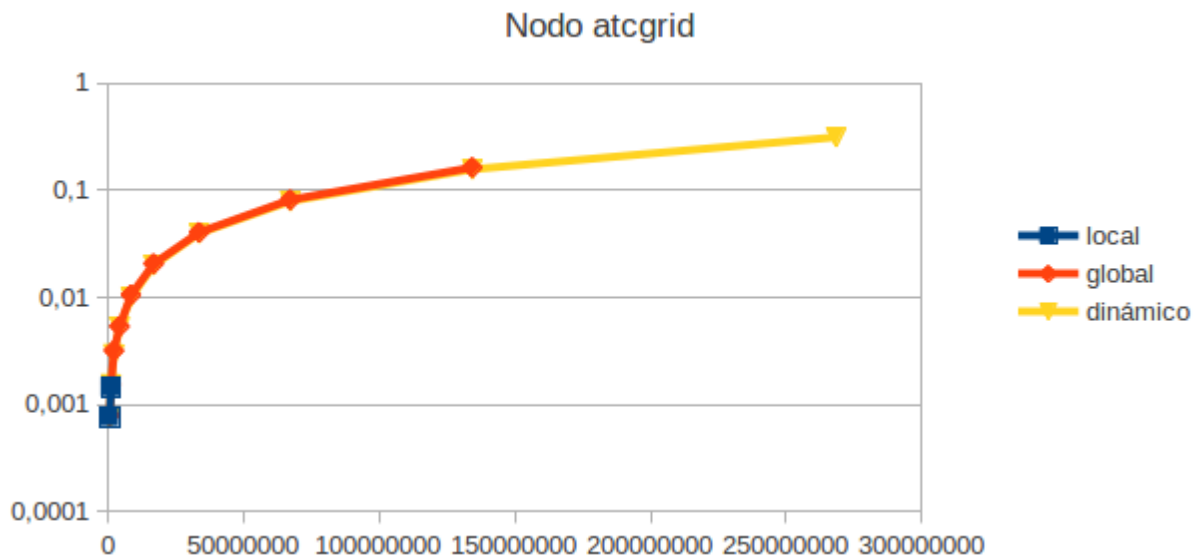
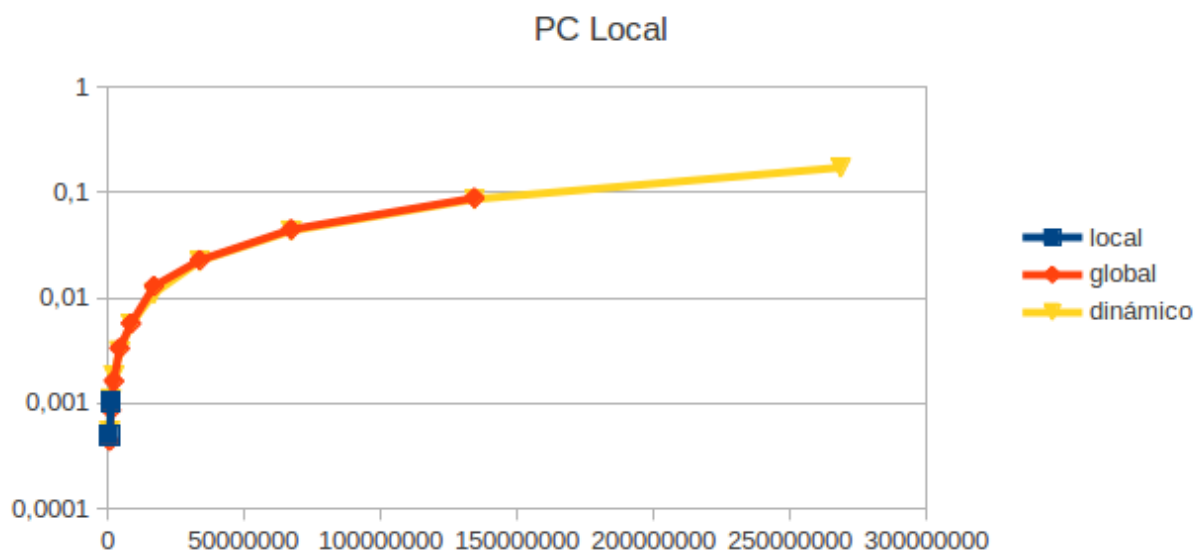


Tabla 2 .Tiempos de ejecución de la suma de vectores para vectores locales, globales y dinámicos

PC Local

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000520326	0.001220482	0.000675365
131072	1048576	0.000493036	0.000440318	0.000538833
262144	2097152	0.001027356	0.000861360	0.001102133
524288	4194304	-	0.001623057	0.001834961
1048576	8388608	-	0.003316856	0.003139780
2097152	16777216	-	0.005705151	0.005650716
4194304	33554432	-	0.012892473	0.011016799
8388608	67108864	-	0.022803497	0.022392792
16777216	134217728	-	0.044571392	0.043320979
33554432	268435456	-	0.088671505	0.086310123
67108864	536870912	-	-	0.172492033



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:

Para cambiar el límite, solo se tiene que cambiar el código `#define MAX 33554432 // = 2^{25}` , por `#define MAX 4294967295 // = $2^{32}-1$` .

Al compilar el nuevo código para variables globales, aparecen unos mensajes de error que dicen lo siguiente:

```
SumaVectoresC_v2.c:(.text.startup+0x79): reubicación truncada para
ajustar: R_X86_64_32S contra el símbolo `v2' definido en la sección
COMMON en /tmp/ccwwtF5L.o
SumaVectoresC_v2.c:(.text.startup+0xc0): reubicación truncada para
ajustar: R_X86_64_32S contra el símbolo `v2' definido en la sección
COMMON en /tmp/ccwwtF5L.o
SumaVectoresC_v2.c:(.text.startup+0xc8): reubicación truncada para
ajustar: R_X86_64_32S contra el símbolo `v3' definido en la sección
COMMON en /tmp/ccwwtF5L.o
SumaVectoresC_v2.c:(.text.startup+0xfc): reubicación truncada para
ajustar: R_X86_64_32S contra el símbolo `v3' definido en la sección
COMMON en /tmp/ccwwtF5L.o
SumaVectoresC_v2.c:(.text.startup+0x115): reubicación truncada para
ajustar: R_X86_64_32S contra el símbolo `v2' definido en la sección
COMMON en /tmp/ccwwtF5L.o
SumaVectoresC_v2.c:(.text.startup+0x12b): reubicación truncada para
ajustar: R_X86_64_PC32 contra el símbolo `v3' definido en la sección
COMMON en /tmp/ccwwtF5L.o
SumaVectoresC_v2.c:(.text.startup+0x135): reubicación truncada para
ajustar: R_X86_64_PC32 contra el símbolo `v2' definido en la sección
COMMON en /tmp/ccwwtF5L.o
```

Esto significa que nos estamos pasando del máximo tamaño a representar y es identificado como desbordamiento por el compilador. El máximo número de N es $2^{32}-1$ porque es el número máximo que se puede representar con un vector de enteros en 32 bits.