2º curso / 2º cuatr.

Grado Ing. Inform.

Doble Grado Ing. Inform. y Mat.

Arquitectura de Computadores (AC)

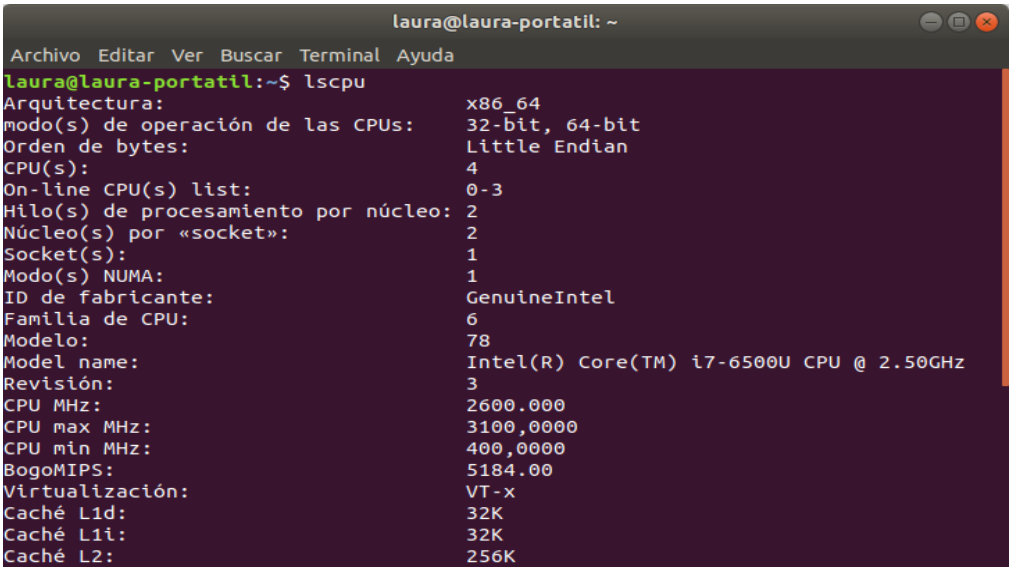
Cuaderno de prácticas.
Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos): Laura Sánchez Parra
Grupo de prácticas: Miércoles
Fecha de entrega: 7 Marzo
Fecha evaluación en clase: 14 Marzo

---

1. Incorpore volcados de pantalla que muestren lo que devuelve `lscpu` en atcgrid y en su PC.

CAPTURAS:



Conteste a las siguientes preguntas:

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

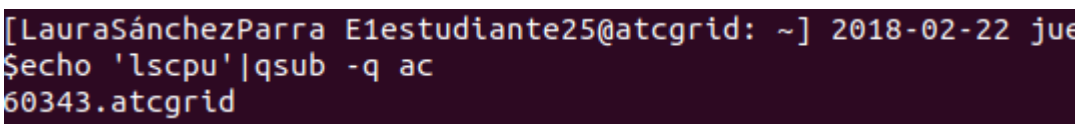
RESPUESTA:

Mi PC tiene 4 cores lógicos, con 2 hilos de procesamiento por core físico, por tanto, cuenta con 2 cores físicos.

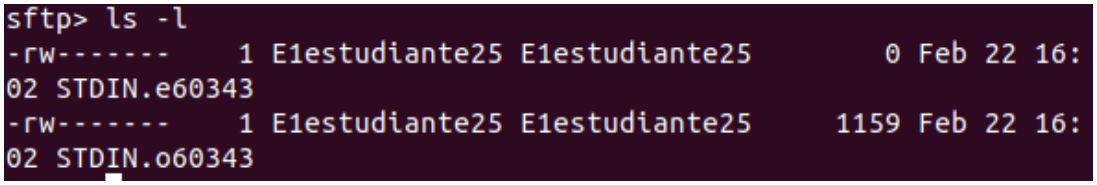
b.¿Cuántos cores físicos y cuántos cores lógicos tiene un nodo de atcgrid?

RESPUESTA:

Vamos a ejecutar el comando lscpu en el nodo de atcgrid asignado.



Comprobamos que el archivo de error está vacío para ver que no se han producido errores.



Finalmente, mostramos la información del archivo de salida.

```
[LauraSánchezParra E1estudiante25@atcgrid: ~] 2018-02-22 jueves
$cat STDIN.o60343
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                24
On-line CPU(s) list:   0-23
Thread(s) per core:    2
Core(s) per socket:    6
```

El nodo de atcgrid tiene 12 cores físicos y 24 lógicos.

2. 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 una variable de tipo double , contiene el tiempo de ejecución.

```
The functions clock_gettime() and clock_settime() retrieve and set  the
time of the specified clock clk_id.
```

Según el manual de Linux, clock_gettime() devuelve el tiempo de reloj.

```
RETURN VALUE
       clock_gettime(), clock_settime(), and clock_getres() return 0 for  suc-
       cess, or -1 for failure (in which case errno is set appropriately).
```

El valor que devuelve no tiene nada que ver con la información que nos interesa, simplemente devuelve 0 si se ha realizado con éxito, y -1 si ha ocurrido algún fallo.

La información interesante, es decir, el tiempo del reloj actual, se devuelve mediante una estructura de datos dada como parámetro:

```
The  res  and  tp  arguments  are  timespec structures, as specified in
<time.h>:

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

La estructura es timespec, que desglosa el tiempo en segundos y nanosegundos.

Es importante resaltar que el punto de referencia de clock_gettime() es el cero de EPOCH es decir el 1de enero de 1970.

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++
Librerías	#include <stdlib.h> #include<stdio.h> #include<time.h>	#include <cstdlib> #include<iostream> #include <time.h>
Inicialización de vectores con memoria dinámica	v1 = (double*) malloc(N*sizeof(double)	v1 = new double [N];
Liberación de memoria dinámicas	free(v1);	delete [] v1;
Imprimir por pantalla	printf	cout

3. 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`). Incorporar volcados de pantalla que demuestren la ejecución correcta en atcgrid o en su PC.

RESPUESTA:

En mi PC:

```
laura@laura-portatil:~$ ./SumaVectoresC 50
Tiempo(seg.):0.000000739        / Tamaño Vectores:50    /V1[0]+V2[0]=V3[0](5.000
000+5.000000=10.000000) / /V1[49]+V2[49]=V3[49](9.900000+0.100000=10.000000) /
```

Vamos a ejecutarlo en el atcgrid:

```
sftp> put SumaVectoresC
Uploading SumaVectoresC to /home/E1estudiante25/SumaVectoresC
SumaVectoresC
                                              100% 8776   494.9KB/
s   00:00
sftp>
```

Llevamos nuestro ejecutable del local al front-end y comprobamos si hay errores:

```
[LauraSánchezParra E1estudiante25@atcgrid: ~] 2018-02-22 jueves
$ls
SumaVectoresC
[LauraSánchezParra E1estudiante25@atcgrid: ~] 2018-02-22 jueves
$echo './SumaVectoresC 50'|qsub -q ac
60373.atcgrid
[LauraSánchezParra E1estudiante25@atcgrid: ~] 2018-02-22 jueves
$ls -l
total 16
-rw------- 1 E1estudiante25 E1estudiante25    0 feb 22 17:40 STDIN.
e60373
-rw------- 1 E1estudiante25 E1estudiante25  153 feb 22 17:40 STDIN.
o60373
-rwxr-xr-x 1 E1estudiante25 E1estudiante25 8776 feb 22 17:36 SumaVe
ctoresC
```

Como no hay errores, enviamos el archivo de salida al pc local y lo visualizamos:

```
sftp> get STDIN.o60373
Fetching /home/E1estudiante25/STDIN.o60373 to STDIN.o60373
/home/E1estudiante25/STDIN.o6 100%  153    30.3KB/s   00:00
sftp>
```

```
laura@laura-portatil:~$ cat STDIN.o60373
Tiempo(seg.):0.000000325        / Tamaño Vectores:50    /V1[0]+V2[0]=V3[0](5.000
000+5.000000=10.000000) / /V1[49]+V2[49]=V3[49](9.900000+0.100000=10.000000) /
```

4. 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 para los mismos tamaños. ¿Se obtiene error para alguno de los tamaños? En caso afirmativo, ¿a qué se debe este error? (Incorporar volcados de pantalla)

RESPUESTA:

```
[LauraSánchezParra E1estudiante25@atcgrid: ~] 2018-03-01 jueves
$ls -l
total 16
-rwxrwxr-x 1 E1estudiante25 E1estudiante25 8536 mar  1 13:02 SumaVectoresC_LOCAL
-rwxrwxr-x 1 E1estudiante25 E1estudiante25  759 mar  1 13:05 SumaVectores.sh
[LauraSánchezParra E1estudiante25@atcgrid: ~] 2018-03-01 jueves
$echo './SumaVectores.sh' | qsub -q ac
62817.atcgrid
[LauraSánchezParra E1estudiante25@atcgrid: ~] 2018-03-01 jueves
$ls
STDIN.e62817  STDIN.o62817  SumaVectoresC_LOCAL  SumaVectores.sh
[LauraSánchezParra E1estudiante25@atcgrid: ~] 2018-03-01 jueves
$ls -l
total 24
-rw------- 1 E1estudiante25 E1estudiante25  880 mar  1 13:06 STDIN.e62817
-rw------- 1 E1estudiante25 E1estudiante25 1025 mar  1 13:06 STDIN.o62817
-rwxrwxr-x 1 E1estudiante25 E1estudiante25 8536 mar  1 13:02 SumaVectoresC_LOCAL
-rwxrwxr-x 1 E1estudiante25 E1estudiante25  759 mar  1 13:05 SumaVectores.sh
[LauraSánchezParra E1estudiante25@atcgrid: ~] 2018-03-01 jueves
$cat STDIN.e62817
./SumaVectores.sh: line 21: 14154 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC_LOCAL $N
./SumaVectores.sh: line 21: 14157 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC_LOCAL $N
./SumaVectores.sh: line 21: 14161 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC_LOCAL $N
./SumaVectores.sh: line 21: 14168 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC_LOCAL $N
./SumaVectores.sh: line 21: 14172 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC_LOCAL $N
./SumaVectores.sh: line 21: 14175 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC_LOCAL $N
./SumaVectores.sh: line 21: 14178 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC_LOCAL $N
./SumaVectores.sh: line 21: 14181 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC_LOCAL $N
[LauraSánchezParra E1estudiante25@atcgrid: ~] 2018-03-01 jueves
$
```

Como podemos comprobar, en el atcgrid, que a partir de cierto número lo suficientemente grande, la ejecución produce 'core'.

Esto se debe a que las variables locales se almacenan en pila, y como sabemos, la pila tiene un espacio limitado. En el momento en que se rellena la pila, los siguientes datos se desbordan y se produce overfload, intentamos acceder a posiciones de memoria que ya no están asignadas a la pila y se produce una excepción de violación de segmento.

5. 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. ¿Se obtiene error usando vectores globales o dinámicos? ¿A qué cree que es debido? (Incorporar volcados de pantalla)

RESPUESTA:

GLOBAL:

```
[LauraSánchezParra E1estudiante25@atcgrid: ~] 2018-03-01 jueves
$ls -l
total 20
-rw------- 1 E1estudiante25 E1estudiante25    0 mar  1 13:23 STDIN.e62831
-rw------- 1 E1estudiante25 E1estudiante25 2617 mar  1 13:23 STDIN.o62831
-rwxr-xr-x 1 E1estudiante25 E1estudiante25 8624 mar  1 13:21 SumaVectoresC_GLOBAL
-rwxrwxr-x 1 E1estudiante25 E1estudiante25  760 mar  1 13:22 SumaVectores.sh
[LauraSánchezParra E1estudiante25@atcgrid: ~] 2018-03-01 jueves
```

Como podemos comprobar, el archivo de error está vacío para el caso de variables globales.

Sin embargo, si mostramos el archivo con los datos obtenidos de la ejecución veremos lo siguiente:

datos_global.txt

```
1  Tiempo(seg.):0.098492475   / Tamaño Vectores:33554432
2  Tiempo(seg.):0.000881945   / Tamaño Vectores:65536  /
3  Tiempo(seg.):0.000490704   / Tamaño Vectores:131072 /
4  Tiempo(seg.):0.000947576   / Tamaño Vectores:262144 /
5  Tiempo(seg.):0.001973089   / Tamaño Vectores:524288 /
6  Tiempo(seg.):0.003982256   / Tamaño Vectores:1048576
7  Tiempo(seg.):0.007687044   / Tamaño Vectores:2097152
8  Tiempo(seg.):0.013104351   / Tamaño Vectores:4194304
9  Tiempo(seg.):0.024995069   / Tamaño Vectores:8388608
10 Tiempo(seg.):0.049101994   / Tamaño Vectores:16777216
11 Tiempo(seg.):0.098942580   / Tamaño Vectores:33554432
12 Tiempo(seg.):0.098763196   / Tamaño Vectores:33554432
13
```

No es difícil darse cuenta que el tiempo para el último tamaño de vectores, 33554432 elementos es menor que el tiempo para vectores con la mitad de elementos.

Nos damos cuenta entonces que se produce un error. ¿A qué se debe?

La reserva de memoria para las variables globales se hace en tiempo de compilación, y por defecto se le asigna una porción de memoria de 2GB. Comprobando la tabla de tamaños y tiempos, veremos que para el último número de elementos, el tamaño en GB es superior a 2, por lo tanto, estamos incurriendo en un error.

DYNAMIC:

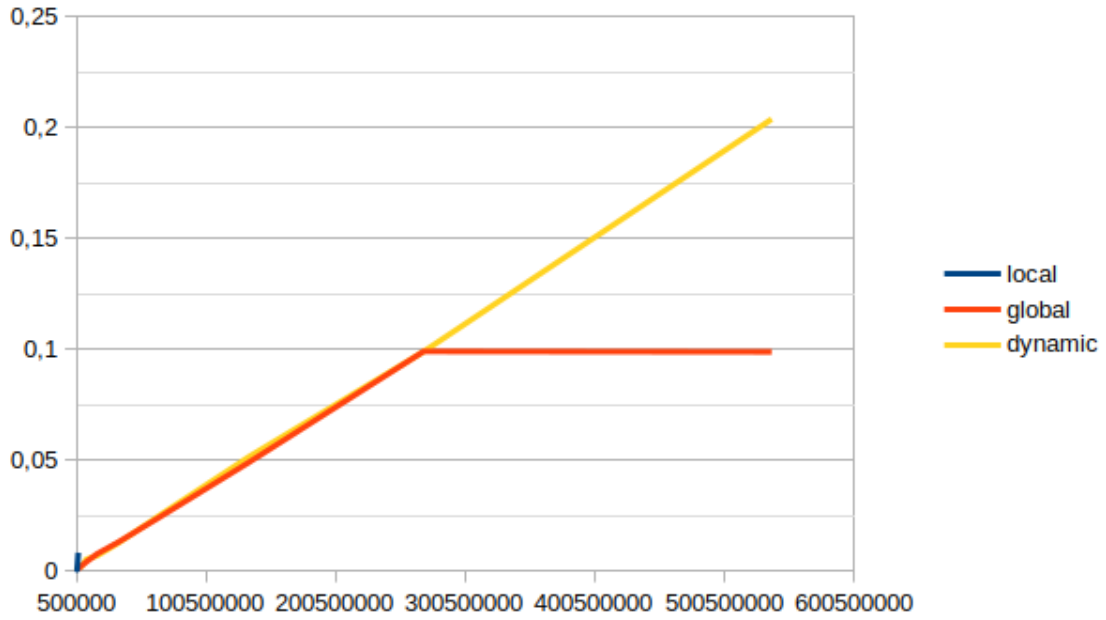
```
[LauraSánchezParra E1estudiante25@atcgrid: ~] 2018-03-01 jueves
$ls
SumaVectoresC_DYNAMIC  SumaVectores.sh
[LauraSánchezParra E1estudiante25@atcgrid: ~] 2018-03-01 jueves
$echo './SumaVectores.sh' | qsub -q ac
62832.atcgrid
[LauraSánchezParra E1estudiante25@atcgrid: ~] 2018-03-01 jueves
$ls -l
total 20
-rwxr-xr-x 1 E1estudiante25 E1estudiante25 12720 mar  1 13:27 SumaVectoresC_DYNAMIC
-rwxrwxr-x 1 E1estudiante25 E1estudiante25   761 mar  1 13:26 SumaVectores.sh
[LauraSánchezParra E1estudiante25@atcgrid: ~] 2018-03-01 jueves
$ls -l
total 24
-rw------- 1 E1estudiante25 E1estudiante25     0 mar  1 13:27 STDIN.e62832
-rw------- 1 E1estudiante25 E1estudiante25  2620 mar  1 13:27 STDIN.o62832
-rwxr-xr-x 1 E1estudiante25 E1estudiante25 12720 mar  1 13:27 SumaVectoresC_DYNAMIC
-rwxrwxr-x 1 E1estudiante25 E1estudiante25   761 mar  1 13:26 SumaVectores.sh
[LauraSánchezParra E1estudiante25@atcgrid: ~] 2018-03-01 jueves
```

6. Rellenar una tabla como la Tabla 1 para atcgrid y otra para su PC 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 y en su PC para vectores locales, globales y dinámicos (eje y) en función del tamaño en bytes de un vector (los valores de la segunda columna de la tabla, que están en escala logarítmica, deben estar en el eje x). Utilice escala logarítmica en el eje de ordenadas (eje y). ¿Hay diferencias en los tiempos de ejecución?

RESPUESTA:

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0,000890503	0,000881945	0,000214570
131072	1048576	0,000418824	0,000490704	0,000396451
262144	2097152	0,008109700	0,000947576	0,001413467
524288	4194304	Core	0,001973089	0,001944419
1048576	8388608	Core	0,003982256	0,005139431
2097152	16777216	Core	0,007687044	0,006811128
4194304	33554432	Core	0,013104351	0,012742842
8388608	67108864	Core	0,024995069	0,025847916
16777216	134217728	Core	0,049101994	0,051588245
33554432	268435456	Core	0,098942580	0,098900759
67108864	536870912	Core	0,098763196	0,203791232

Gráficamente:

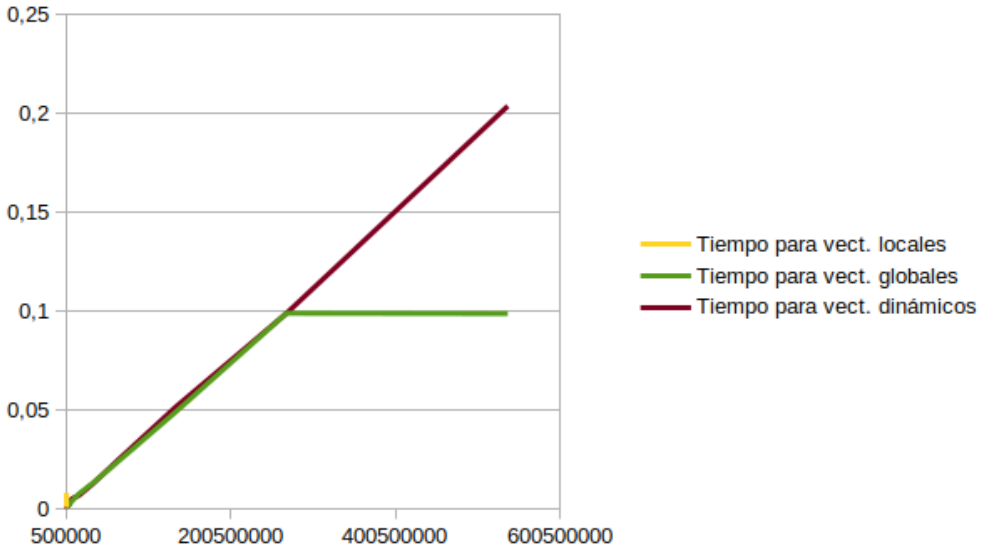


Como vemos, se ha producido un error en global, mientras que dynamic continua su ejecución y local deja de ejecutarse casi para los primeros valores.

Por otro lado, los datos obtenidos de la ejecución en el atcgrid son:

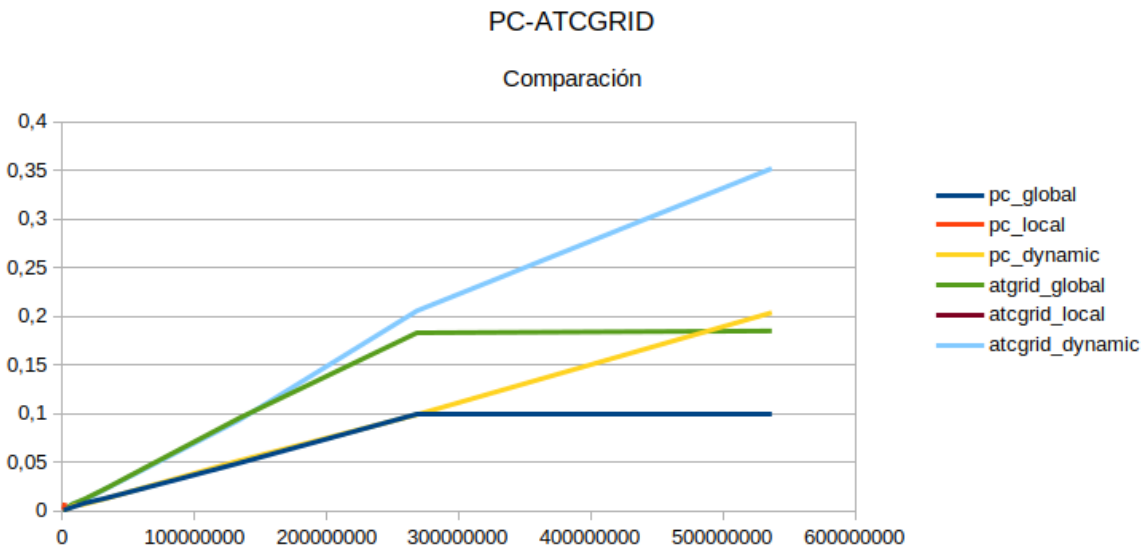
Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0,000421366	0,000476338	0,000420419
131072	1048576	0,000851256	0,000847464	0,000865304
262144	2097152	0,001533483	0,001290105	0,001588696
524288	4194304	core	0,003666085	0,003255534
1048576	8388608	core	0,007082477	0,007071442
2097152	16777216	core	0,011920947	0,010946437
4194304	33554432	core	0,023306494	0,022987640
8388608	67108864	core	0,047423196	0,046262578
16777216	134217728	core	0,095433444	0,093452759
33554432	268435456	core	0,183021304	0,205636492
67108864	536870912	core	0,184894549	0,352030667

Gráficamente:



La conclusión es la misma que para el caso de mi ordenador personal.

A simple vista parece que las gráficas son bastante similares, si pintamos una gráfica comparativa, como la siguiente, veremos que la ejecución en nuestro ordenador personal con variables locales, no dista mucho, si bien es cierto que en el nuestro ordenador, parece tiende a dispararse. Sin embargo, tanto para las variables globales como para las dinámicas, para vectores con menor número de componentes, los tiempos de ejecución son bastante próximos en el atcgrid y en nuestro pc, mientras que a mayor número de componentes, las ejecuciones en el pc son más rápidas.



7. 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:

```
laura@laura-portatil:~$ gcc -O2 SumaVectoresC_GLOBAL_MAX.c -o SumaVectoresC_GLOB
ALMAX
/tmp/ccYYZysG.o: En la función `main':
SumaVectoresC_GLOBAL_MAX.c:(.text.startup+0x5e): reubicación truncada para ajust
ar: R_X86_64_PC32 contra el símbolo `v2' definido en la sección COMMON en /tmp/c
cYYZysG.o
SumaVectoresC_GLOBAL_MAX.c:(.text.startup+0xb1): reubicación truncada para ajust
ar: R_X86_64_PC32 contra el símbolo `v3' definido en la sección COMMON en /tmp/c
cYYZysG.o
SumaVectoresC_GLOBAL_MAX.c:(.text.startup+0x130): reubicación truncada para ajus
tar: R_X86_64_PC32 contra el símbolo `v3' definido en la sección COMMON en /tmp/
ccYYZysG.o
SumaVectoresC_GLOBAL_MAX.c:(.text.startup+0x13b): reubicación truncada para ajus
tar: R_X86_64_PC32 contra el símbolo `v2' definido en la sección COMMON en /tmp/
ccYYZysG.o
SumaVectoresC_GLOBAL_MAX.c:(.text.startup+0x1a0): reubicación truncada para ajus
tar: R_X86_64_PC32 contra el símbolo `v2' definido en la sección COMMON en /tmp/
ccYYZysG.o
SumaVectoresC_GLOBAL_MAX.c:(.text.startup+0x1a7): reubicación truncada para ajus
tar: R_X86_64_PC32 contra el símbolo `v3' definido en la sección COMMON en /tmp/
ccYYZysG.o
collect2: error: ld returned 1 exit status
laura@laura-portatil:~$
```

Se produce un error. Este error se debe al espacio que se reserva en tiempo de compilación para almacenar variables globales. Hay vectores tan grandes que superan la capacidad del espacio reservado.

El mayor número que se puede almacenar es $2^{31} - 1$ pues es el mayor número que ocupa menos de 2GB.

Listado 1 . Código C que suma dos vectores

```
/* SumaVectoresC.c
 Suma de dos vectores: v3 = v1 + v2

 Para compilar usar (-lrt: real time library):
        gcc  -O2 SumaVectores.c -o SumaVectores –lrt
 gcc  -O2 –S SumaVectores.c –lrt   //para generar el código ensamblador

 Para ejecutar use: SumaVectoresC longitud
*/

#include <stdlib.h> // biblioteca con funciones atoi(), malloc() y free()
#include <stdio.h> // biblioteca donde se encuentra la función printf()
#include <time.h>       // biblioteca donde se encuentra la función clock_gettime()

//#define PRINTF_ALL      // comentar para quitar el printf ...
                          // que imprime todos los componentes
//Sólo puede estar definida una de las tres constantes VECTOR_ (sólo uno de los ...
//tres defines siguientes puede estar descomentado):
//#define VECTOR_LOCAL    // descomentar para que los vectores sean variables ...
                          // locales (si se supera el tamaño de la pila se ...
                          // generará el error "Violación de Segmento")
//#define VECTOR_GLOBAL// descomentar para que los vectores sean variables ...
                          // globales (su longitud no estará limitada por el ...
                          // tamaño de la pila del programa)
#define VECTOR_DYNAMIC    // descomentar para que los vectores sean variables ...
                          // dinámicas (memoria reutilizable durante la ejecución)
#ifdef VECTOR_GLOBAL
#define MAX 33554432         //=2^25
double v1[MAX], v2[MAX], v3[MAX];
#endif

int main(int argc, char** argv){

   int i;
   struct timespec cgt1,cgt2; double ncgt; //para tiempo de ejecución

   //Leer argumento de entrada (nº de componentes del vector)
   if (argc<2){
      printf("Faltan nº componentes del vector\n");
      exit(-1);
   }

   unsigned int N = atoi(argv[1]); // Máximo N =2^32-1=4294967295 (sizeof(unsigned
int) = 4 B)
   #ifdef VECTOR_LOCAL
   double v1[N], v2[N], v3[N];   // Tamaño variable local en tiempo de ejecución ...
                                 // disponible en C a partir de actualización C99
   #endif
   #ifdef VECTOR_GLOBAL
   if (N>MAX) N=MAX;
   #endif
   #ifdef VECTOR_DYNAMIC
   double *v1, *v2, *v3;
   v1 = (double*) malloc(N*sizeof(double));// malloc necesita el tamaño en bytes
   v2 = (double*) malloc(N*sizeof(double)); //si no hay espacio suficiente malloc
```

```
devuelve NULL
   v3 = (double*) malloc(N*sizeof(double));
      if ( (v1==NULL) || (v2==NULL) || (v3==NULL) ){
      printf("Error en la reserva de espacio para los vectores\n");
      exit(-2);
   }
   #endif

   //Inicializar vectores
   for(i=0; i<N; i++){
      v1[i] = N*0.1+i*0.1; v2[i] = N*0.1-i*0.1; //los valores dependen de N
   }

   clock_gettime(CLOCK_REALTIME,&cgt1);
   //Calcular suma de vectores
   for(i=0; i<N; i++)
      v3[i] = v1[i] + v2[i];

   clock_gettime(CLOCK_REALTIME,&cgt2);
   ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+
          (double) ((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));

   //Imprimir resultado de la suma y el tiempo de ejecución
   #ifdef PRINTF_ALL
   printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%u\n",ncgt,N);
   for(i=0; i<N; i++)
      printf("/ V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f) /\n",
                i,i,i,v1[i],v2[i],v3[i]);

   #else
      printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%u\t/ V1[0]+V2[0]=V3[0](%8.6f+
%8.6f=%8.6f) / /
                V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f) /\n",
                ncgt,N,v1[0],v2[0],v3[0],N-1,N-1,N-1,v1[N-1],v2[N-1],v3[N-1]);
   #endif

   #ifdef VECTOR_DYNAMIC
   free(v1); // libera el espacio reservado para v1
   free(v2); // libera el espacio reservado para v2
   free(v3); // libera el espacio reservado para v3
   #endif
   return 0;
}
```

Listado 2 . Código C++ que suma dos vectores

```
/* SumaVectoresCpp.cpp
  Suma de dos vectores: v3 = v1 + v2

  Para compilar usar (-lrt: real time library):
                  g++  -O2 SumaVectoresCpp.cpp -o SumaVectoresCpp -lrt

  Para ejecutar use: SumaVectoresCpp longitud
```

```
*/

#include <cstdlib>   // biblioteca con atoi()
#include <iostream> // biblioteca donde se encuentra la función cout
using namespace std;
#include <time.h>       // biblioteca donde se encuentra la función clock_gettime()

//#define COUT_ALL // comentar para quitar el cout ...
                   // que imprime todos los componentes
//Sólo puede estar definida una de las tres constantes VECTOR_ (sólo uno de los ...
//tres defines siguientes puede estar descomentado):
//#define VECTOR_LOCAL    // descomentar para que los vectores sean variables ...
                          // locales (si se supera el tamaño de la pila se ...
                          // generará el error "Violación de Segmento")
//#define VECTOR_GLOBAL// descomentar para que los vectores sean variables ...
                          // globales (su longitud no estará limitada por el ...
                          // tamaño de la pila del programa)
#define VECTOR_DYNAMIC    // descomentar para que los vectores sean variables ...
                          // dinámicas (memoria reutilizable durante la ejecución)
#ifdef VECTOR_GLOBAL
#define MAX 33554432       //=2^25
double v1[MAX], v2[MAX], v3[MAX];
#endif

int main(int argc, char** argv){

   struct timespec cgt1,cgt2;   //para tiempo de ejecución

   //Leer argumento de entrada (nº de componentes del vector)
   if (argc<2){
      cout << "Faltan nº componentes del vector\n" << endl ;
      exit(-1);
   }

   unsigned int N = atoi(argv[1]);
   #ifdef VECTOR_LOCAL
   double v1[N], v2[N], v3[N];
   #endif
   #ifdef VECTOR_GLOBAL
   if (N>MAX) N=MAX;
   #endif
   #ifdef VECTOR_DYNAMIC
   double *v1, *v2, *v3;
   v1 = new double [N];    //si no hay espacio suficiente new genera una excepción
   v2 = new double [N];
   v3 = new double [N];
   #endif

   //Inicializar vectores
   for(int i=0; i<N; i++){
      v1[i] = N*0.1+i*0.1; v2[i] = N*0.1-i*0.1; //los valores dependen de N
   }
    clock_gettime(CLOCK_REALTIME,&cgt1);
   //Calcular suma de vectores
   for(int i=0; i<N; i++)
      v3[i] = v1[i] + v2[i];
    clock_gettime(CLOCK_REALTIME,&cgt2);
```

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

   //Imprimir resultado de la suma y el tiempo de ejecución
   #ifdef COUT_ALL
   cout << "Tiempo(seg.):" << ncgt << "\t/ Tamaño Vectores:" << N << endl;
   for(int i=0; i<N; i++)
         cout << "/ V1[" << i << "]+V2[" << i << "]=V3" << i << "](" << v1[i] << "+"
<< v2[i] << "="
         << v3[i] << ") /\t" << endl;
   cout <<"\n"<< endl;
   #else
         cout << "Tiempo(seg.):" << ncgt << "\t/ Tamaño Vectores:" << N << "\t/
V1[0]+V2[0]=V3[0]("
         << v1[0] << "+" << v2[0] << "=" << v3[0] << ") / / V1[" << N-1 << "]+V2["
<< N-1 << "]=V3["
         << N-1 << "](" << v1[N-1] << "+" << v2[N-1] << "=" << v3[N-1] << ")/\n" <<
endl;
   #endif

   #ifdef VECTOR_DYNAMIC
   delete [] v1; // libera el espacio reservado para v1
   delete [] v2; // libera el espacio reservado para v2
   delete [] v3; // libera el espacio reservado para v3
   #endif
   return 0;
}
```

Listado 3 . Script para la suma de vectores (SumaVectores.sh). Se supone en el script que el fichero a ejecutar se llama SumaVectorC y que se encuentra en el directorio en el que se ha ejecutado qsub.

```
#!/bin/bash
#Se asigna al trabajo el nombre SumaVectoresC_vlocales
#PBS -N SumaVectoresC_vlocales
#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 "Directorio en el que se ha ejecutado qsub: $PBS_O_WORKDIR"
echo "Cola: $PBS_QUEUE"
echo "Nodos asignados al trabajo:"
cat $PBS_NODEFILE
#Se ejecuta SumaVectorC, que está en el directorio en el que se ha ejecutado qsub,
#para N potencia de 2 desde 2^16 a 2^26
for ((N=65536;N<67108865;N=N*2))
do
    $PBS_O_WORKDIR/SumaVectoresC $N
done
```