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 José González García
Grupo de prácticas: A2
Fecha de entrega:
Fecha evaluación en clase:

---

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

CAPTURAS:

MI PC:

```
[FranciscoJoseGonzalezGarcia fran@fran-Aspire-V3-572G:~/Descargas] 2018-02-26 lunes
$lscpu
Arquitectura:          x86_64
modo(s) de operación de las CPUs:32-bit, 64-bit
Orden de bytes:        Little Endian
CPU(s):                4
On-line CPU(s) list:   0-3
Hilo(s) de procesamiento por núcleo:2
Núcleo(s) por «socket»:2
Socket(s):             1
Modo(s) NUMA:          1
ID de fabricante:      GenuineIntel
Familia de CPU:        6
Modelo:                61
Model name:            Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz
Revisión:              4
CPU MHz:               2394.613
CPU max MHz:           3000,0000
CPU min MHz:           500,0000
BogoMIPS:              4789.22
Virtualización:        VT-x
Caché L1d:             32K
Caché L1i:             32K
Caché L2:              256K
Caché L3:              4096K
NUMA node0 CPU(s):     0-3
```

CLUSTER:

```
[FranciscoJoseGonzalezGarcia A2estudiante6@atcgrid:~] 2018-02-26 lunes
$cat STDIN.o61635
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
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 44
Model name:            Intel(R) Xeon(R) CPU           E5645  @ 2.40GHz
Stepping:              2
CPU MHz:               1602.611
CPU max MHz:           2401.0000
CPU min MHz:           1600.0000
BogoMIPS:              4800.14
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              12288K
NUMA node0 CPU(s):     0-5,12-17
NUMA node1 CPU(s):     6-11,18-23
```

FRONT-END:

```
[FranciscoJoseGonzalezGarcia A2estudiante6@atcgrid:~] 2018-02-26 lunes
$lscpu
Arquitectura:                          x86_64
modo(s) de operación de las CPUs:      32-bit, 64-bit
Orden de los bytes:                    Little Endian
CPU(s):                                1
Lista de la(s) CPU(s) en línea:        0
Hilo(s) de procesamiento por núcleo:   1
Núcleo(s) por «socket»:                1
«Socket(s)»                            1
Modo(s) NUMA:                          1
ID de fabricante:                      AuthenticAMD
Familia de CPU:                        15
Modelo:                                47
Nombre del modelo:                     AMD Athlon(tm) 64 Processor 3000+
Revisión:                              2
CPU MHz:                               1802.252
BogoMIPS:                              3604.50
Caché L1d:                             64K
Caché L1i:                             64K
Caché L2:                              512K
CPU(s) del nodo NUMA 0:                0
```

Conteste a las siguientes preguntas:

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

RESPUESTA:

MAQUINA	CORES FISICOS	CORES LOGICOS
atcgrid (front-end)	1	1
Mi PC	2	4

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

RESPUESTA:

Cores físicos: **12** Cores lógicos: **24**

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 contiene la diferencia de tiempo entre cgt1 y cgt2 para calcular el tiempo empleado en realizar la suma. La función clock_gettime() devuelve el tiempo transcurrido desde una fecha determinada denominada *EPOCH*. La función clock_gettime() hace uso del struct timespec para devolver el tiempo calculado. La estructura es la siguente:

```
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++
salida de datos	printf	cout
liberar vector dinámico	free(*vector)*	delete [] *vector*
reserva de memoria vector dinámico	(type) malloc(size)	new type [size]
bibliotecas usadas	<stdlib.h> <stdio.h>	<cstdlib> <iostream>

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:

```
[FranciscoJoseGonzalezGarcia fran@fran-Aspire-V3-572G:~/Universi
dad/AC/Practicas/Practica0] 2018-02-27 martes
$./SumaVectoresC 10000
Tiempo(seg.):0.000042613        / Tamaño Vectores:10000
/ V1[0]+V2[0]=V3[0](1000.000000+1000.000000=2000.000000) / /
 V1[9999]+V2[9999]=V3[9999](1999.900000+0.100000=2000.000000)/
```

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:

Se obtiene error a partir del tamaño N=524288 debido a que la pila no tiene capacidad suficiente para almacenar todos los elementos del vector y por eso se produce una violación de segmento al intentar acceder a posiciones de memoria fuera de la pila.

MI PC:

```
[FranciscoJoseGonzalezGarcia fran@fran-Aspire-V3-572G:~/Universidad/AC/Practicas/Practica0] 2018-02-27 martes
$./SumaVectores.sh
Tiempo(seg.):0.000380899        / Tamaño Vectores:65536        / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) / /     V1[65535]+V2[65535]=V3[65535](13107.1000
00+0.100000=13107.200000)/
Tiempo(seg.):0.000587577        / 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.):0.001386568        / 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)/
./SumaVectores.sh: línea 7: 14921 Violación de segmento  (`core' generado) ./SumaVectoresC $N
./SumaVectores.sh: línea 7: 14923 Violación de segmento  (`core' generado) ./SumaVectoresC $N
./SumaVectores.sh: línea 7: 14925 Violación de segmento  (`core' generado) ./SumaVectoresC $N
./SumaVectores.sh: línea 7: 14927 Violación de segmento  (`core' generado) ./SumaVectoresC $N
./SumaVectores.sh: línea 7: 14929 Violación de segmento  (`core' generado) ./SumaVectoresC $N
./SumaVectores.sh: línea 7: 14931 Violación de segmento  (`core' generado) ./SumaVectoresC $N
./SumaVectores.sh: línea 7: 14933 Violación de segmento  (`core' generado) ./SumaVectoresC $N
./SumaVectores.sh: línea 7: 14935 Violación de segmento  (`core' generado) ./SumaVectoresC $N
```

ATC-GRID:

```
[FranciscoJoseGonzalezGarcia A2estudiante6@atcgrid:~] 2018-02-27 martes
$cat SumaVectoresC_vlocales.o62280 SumaVectoresC_vlocales.e62280
Id. usuario del trabajo: A2estudiante6
Id. del trabajo: 62280.atcgrid
Nombre del trabajo especificado por usuario: SumaVectoresC_vlocales
Nodo que ejecuta qsub: atcgrid
Directorio en el que se ha ejecutado qsub: /home/A2estudiante6
Cola: ac
Tiempo(seg.):0.000421793        / Tamaño Vectores:65536        / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) / /     V1[65535]+V2[65535]=V3[65535](13107.1000
00+0.100000=13107.200000)/
Tiempo(seg.):0.000845389        / 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.):0.001689314        / 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)/
/var/lib/torque/mom_priv/jobs/62280.atcgrid.SC: line 20: 16373 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
/var/lib/torque/mom_priv/jobs/62280.atcgrid.SC: line 20: 16376 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
/var/lib/torque/mom_priv/jobs/62280.atcgrid.SC: line 20: 16379 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
/var/lib/torque/mom_priv/jobs/62280.atcgrid.SC: line 20: 16385 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
/var/lib/torque/mom_priv/jobs/62280.atcgrid.SC: line 20: 16389 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
/var/lib/torque/mom_priv/jobs/62280.atcgrid.SC: line 20: 16393 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
/var/lib/torque/mom_priv/jobs/62280.atcgrid.SC: line 20: 16398 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
/var/lib/torque/mom_priv/jobs/62280.atcgrid.SC: line 20: 16401 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
```

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:

No se obtiene errores en ninguno de los dos casos. El motivo es que en el caso de los vectores globales se almacenan en el segmento de datos del programa y en el caso de los vectores dinámicos se almacenan en el heap y en ambos casos se dispone de memoria suficiente para almacenar todos los datos.

MI PC:

Vectores dinámicos:

```
[FranciscoJoseGonzalezGarcia fran@fran-Aspire-V3-572G:~/Universidad/AC/Practicas/Practica0] 2018-02-27 martes
$./SumaVectores-dinamico.sh
Tiempo(seg.):0.000245345         / Tamaño Vectores:65536        / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) / /     V1[65535]+V2[65535]=V3[65535](13107.1000
00+0.100000=13107.200000)/
Tiempo(seg.):0.000528205         / 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.):0.001203445         / 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.):0.002268567         / Tamaño Vectores:524288       / V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000) / /     V1[524287]+V2[524287]=V3[524287](1048
57.500000+0.100000=104857.600000)/
Tiempo(seg.):0.004605333         / 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.):0.008388436         / 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.):0.017902925         / 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.):0.035696775         / 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.):0.070582132         / Tamaño Vectores:16777216     / V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.200000) / /     V1[16777215]+V2[16777215]=V3[167
77215](3355443.100000+0.100000=3355443.200000)/
Tiempo(seg.):0.138739080         / Tamaño Vectores:33554432     / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) / /     V1[33554431]+V2[33554431]=V3[335
54431](6710886.300000+0.100000=6710886.400000)/
Tiempo(seg.):0.257365040         / Tamaño Vectores:67108864     / V1[0]+V2[0]=V3[0](6710886.400000+6710886.400000=13421772.800000) / /     V1[67108863]+V2[67108863]=V3[67
108863](13421772.700000+0.100000=13421772.800000)/
```

Vectores globales:

```
[FranciscoJoseGonzalezGarcia fran@fran-Aspire-V3-572G:~/Universidad/AC/Practicas/Practica0] 2018-02-27 martes
$./SumaVectores-global.sh
Tiempo(seg.):0.000296779         / Tamaño Vectores:65536        / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) / /     V1[65535]+V2[65535]=V3[65535](13107.1000
00+0.100000=13107.200000)/
Tiempo(seg.):0.000541941         / 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.):0.001639669         / 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.):0.002174893         / Tamaño Vectores:524288       / V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000) / /     V1[524287]+V2[524287]=V3[524287](1048
57.500000+0.100000=104857.600000)/
Tiempo(seg.):0.004275559         / 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.):0.008619841         / 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.):0.017778150         / 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.):0.034849515         / 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.):0.071159161         / Tamaño Vectores:16777216     / V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.200000) / /     V1[16777215]+V2[16777215]=V3[167
77215](3355443.100000+0.100000=3355443.200000)/
Tiempo(seg.):0.140170573         / Tamaño Vectores:33554432     / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) / /     V1[33554431]+V2[33554431]=V3[335
54431](6710886.300000+0.100000=6710886.400000)/
Tiempo(seg.):0.142758991         / Tamaño Vectores:33554432     / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) / /     V1[33554431]+V2[33554431]=V3[335
54431](6710886.300000+0.100000=6710886.400000)/
```

ATC-GRID:

Vectores dinamicos:

```
[FranciscoJoseGonzalezGarcia A2estudiante6@atcgrid:~] 2018-02-27 martes
$cat SumaVectores-dinamico.sh.062293
Tiempo(seg.):0.000434535         / Tamaño Vectores:65536        / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) / /     V1[65535]+V2[65535]=V3[65535](13107.1000
00+0.100000=13107.200000)/
Tiempo(seg.):0.000854008         / 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.):0.001711913         / 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.):0.002526708         / Tamaño Vectores:524288       / V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000) / /     V1[524287]+V2[524287]=V3[524287](1048
57.500000+0.100000=104857.600000)/
Tiempo(seg.):0.006738439         / 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.):0.012218785         / 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.):0.023670683         / 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.):0.045700581         / 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.):0.091276321         / Tamaño Vectores:16777216     / V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.200000) / /     V1[16777215]+V2[16777215]=V3[167
77215](3355443.100000+0.100000=3355443.200000)/
Tiempo(seg.):0.178355067         / Tamaño Vectores:33554432     / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) / /     V1[33554431]+V2[33554431]=V3[335
54431](6710886.300000+0.100000=6710886.400000)/
Tiempo(seg.):0.175810522         / Tamaño Vectores:33554432     / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) / /     V1[33554431]+V2[33554431]=V3[335
54431](6710886.300000+0.100000=6710886.400000)/
```

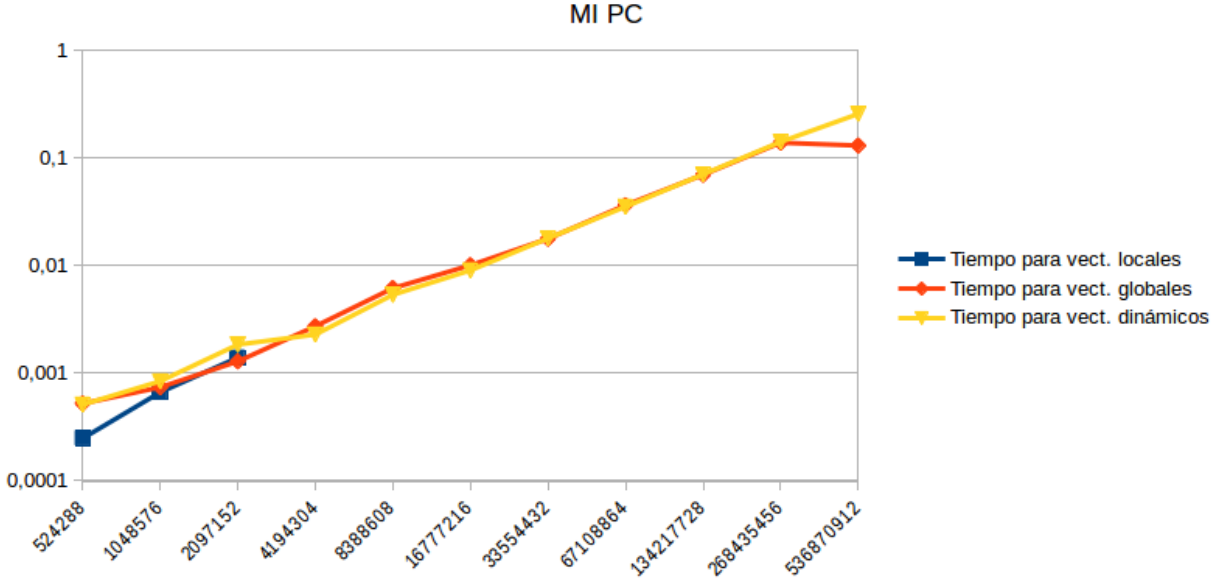
Vectores globales:

```
[FranciscoJoseGonzalezGarcia A2estudiante6@atcgrid:~] 2018-02-27 martes
$cat SumaVectores-global.sh.o62291
Tiempo(seg.):0.000274695        / Tamaño Vectores:65536         / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) / /     V1[65535]+V2[65535]=V3[65535](13107.1000
00+0.100000=13107.200000)/
Tiempo(seg.):0.000850424        / 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.):0.001690509        / 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.):0.002738065        / Tamaño Vectores:524288        / V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000) / /     V1[524287]+V2[524287]=V3[524287](1048
57.500000+0.100000=104857.600000)/
Tiempo(seg.):0.005894954        / 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.):0.012130150        / 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.):0.023776143        / 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.):0.047093195        / 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.):0.095600069        / Tamaño Vectores:16777216      / V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.200000) / /     V1[16777215]+V2[16777215]=V3[167
77215](3355443.100000+0.100000=3355443.200000)/
Tiempo(seg.):0.186037674        / Tamaño Vectores:33554432      / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) / /     V1[33554431]+V2[33554431]=V3[335
54431](6710886.300000+0.100000=6710886.400000)/
Tiempo(seg.):0.367994711        / Tamaño Vectores:67108864      / V1[0]+V2[0]=V3[0](6710886.400000+6710886.400000=13421772.800000) / /     V1[67108863]+V2[67108863]=V3[67
108863](13421772.700000+0.100000=13421772.800000)/
```

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?

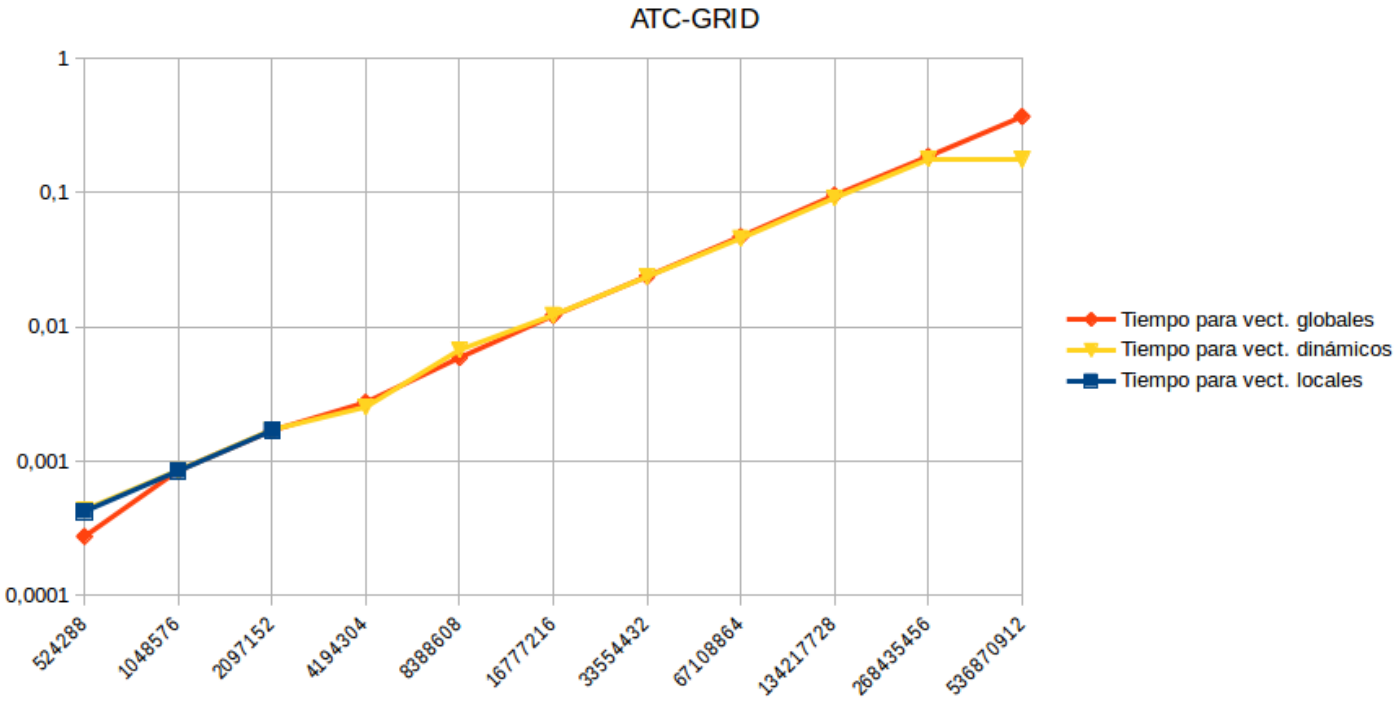
MI PC

N.º de componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.00024832	0,0005236	0,000511325
131072	1048576	0,000665659	0,000740975	0,00084374
262144	2097152	0,001399151	0,001289809	0,001846015
524288	4194304		0,002738092	0,002291875
1048576	8388608		0,006194141	0,005370506
2097152	16777216		0,010113557	0,009068859
4194304	33554432		0,017843555	0,017991357
8388608	67108864		0,036572584	0,035394835
16777216	134217728		0,070138195	0,070693131
33554432	268435456		0,139303096	0,141846831
67108864	536870912		0,130915103	0,257839466



ATC-GRID

N.º de componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0,000421793	0,000274695	0,000434535
131072	1048576	0,000845389	0,000850424	0,000854008
262144	2097152	0,001689314	0,001690509	0,001711913
524288	4194304		0,002738065	0,002526708
1048576	8388608		0,005894954	0,006738439
2097152	16777216		0,01213015	0,012218785
4194304	33554432		0,023776143	0,023670683
8388608	67108864		0,047093195	0,045700581
16777216	134217728		0,095600069	0,091276321
33554432	268435456		0,186037674	0,178355067
67108864	536870912		0,367994711	0,175810522



Como podemos comprobar tanto en las gráficas como en las tablas, las diferencias de tiempo entre ambas máquinas no es muy destacable. Las gráficas tienen una tendencia lineal, con ligeras alteraciones en algunos puntos, propias a que las máquinas no dedican el 100% del tiempo a la ejecución del programa.

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:

El ejecutable no se puede generar porque el enlazador produce un error cuando intenta asignar un espacio en el segmento de datos y no lo encuentra porque el tamaño dado excede al disponible.

El máximo número que se puede almacenar en N es 2^{32}-1 porque es un unsigned int que tiene un tamaño de 4 bytes (32 bits), por lo tanto puede almacenar valores desde 0 hasta 2^{32}-1.

```
[FranciscoJoseGonzalezGarcia fran@fran-Aspire-V3-572G:~/Universidad/AC/Practicas/Practica0] 2018-02-27 martes
$gcc -O2 SumaVectoresC.c -o SumaVectoresC-global-mod -lrt
/tmp/cccFTEl1.o: En la función `main':
SumaVectoresC.c:(.text.startup+0x79): reubicación truncada para ajustar: R_X86_64_32S contra el símbolo `v2' definido en la sección COMMON en /tmp/cccFTEl1.o
SumaVectoresC.c:(.text.startup+0xc0): reubicación truncada para ajustar: R_X86_64_32S contra el símbolo `v2' definido en la sección COMMON en /tmp/cccFTEl1.o
SumaVectoresC.c:(.text.startup+0xc8): reubicación truncada para ajustar: R_X86_64_32S contra el símbolo `v3' definido en la sección COMMON en /tmp/cccFTEl1.o
SumaVectoresC.c:(.text.startup+0xfc): reubicación truncada para ajustar: R_X86_64_32S contra el símbolo `v3' definido en la sección COMMON en /tmp/cccFTEl1.o
SumaVectoresC.c:(.text.startup+0x115): reubicación truncada para ajustar: R_X86_64_32S contra el símbolo `v2' definido en la sección COMMON en /tmp/cccFTEl1.o
SumaVectoresC.c:(.text.startup+0x12b): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v3' definido en la sección COMMON en /tmp/cccFTEl1.o
SumaVectoresC.c:(.text.startup+0x135): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v2' definido en la sección COMMON en /tmp/cccFTEl1.o
collect2: error: ld returned 1 exit status
```

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
```