2º curso / 2º cuatr.

Grado Ingeniería Informática

Arquitectura de Computadores (AC)

Cuaderno de prácticas.
Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos):Daniel Monjas Miguélez
Grupo de prácticas y profesor de prácticas:Miércoles, Mancia Anguita
Fecha de entrega:2 de marzo del 2020
Fecha evaluación en clase:4 de febrero del 2020

Antes de comenzar a realizar el trabajo de este cuaderno consultar el fichero con los normas de prácticas que se encuentra en SWAD

Parte I. Ejercicios basados en los ejemplos del seminario práctico

Crear el directorio con nombre `bp0` en atcgrid y en el PC local.

NOTA: En las prácticas se usa slurm como gestor de colas. Consideraciones a tener en cuenta:

- Slurm está configurado para asignar recursos a los procesos (llamados *tasks* en slurm) a nivel de core físico. Esto significa que por defecto slurm asigna un core a un proceso, para asignar más de uno se debe usar con `sbatch/srun` la opción `--cpus-per-task`.
- En slurm, por defecto, `cpu` se refiere a cores lógicos (ej. en la opción `--cpus-per-task`), si no se quieren usar cores lógicos hay que añadir la opción `--hint=nomultithread` a `sbatch/srun`.
- Para asegurar que solo se crea un proceso hay que incluir `-n1` en `sbatch/srun`.
- Para que no se ejecute más de un proceso en un nodo de atcgrid hay que usar `--exclusive` con `sbatch/srun` (se recomienda no utilizarlo en los `srun` dentro de un script).
- Los `srun` dentro de un script heredan las opciones fijadas en el `sbatch` que se usa para enviar el script a la cola slurm.

1. Ejecutar `lscpu` en el PC y en un nodo de cómputo de atcgrid. (Crear directorio `ejer1`)

   (a) Mostrar con capturas de pantalla el resultado de estas ejecuciones.

   RESPUESTA: La primera captura es de mi PC y la segunda de un nodo de cómputo de atcgrid

```
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~] 2020-02-28 viernes
$lscpu
Architecture:        x86_64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):              8
On-line CPU(s) list: 0-7
Thread(s) per core:  2
Core(s) per socket:  4
Socket(s):           1
NUMA node(s):        1
Vendor ID:           GenuineIntel
CPU family:          6
Model:               158
Model name:          Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz
Stepping:            10
CPU MHz:             900.433
CPU max MHz:         4000,0000
CPU min MHz:         800,0000
BogoMIPS:            4599.93
Virtualisation:      VT-x
L1d cache:           32K
L1i cache:           32K
L2 cache:            256K
L3 cache:            8192K
NUMA node0 CPU(s):   0-7
Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts
acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts r
ep_good nopl xtopology nonstop_tsc cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3
 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdr
and lahf_lm abm 3dnowprefetch cpuid_fault epb invpcid_single pti ssbd ibrs ibpb stibp tpr_shadow vnmi fle
xpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflus
hopt intel_pt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify hwp_act_window hwp_ep
p md_clear flush_l1d
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~] 2020-02-28 viernes
$
```

```
daniel@daniel-XPS-15-9570:~$ ssh -X e1estudiante18@atcgrid.ugr.es
e1estudiante18@atcgrid.ugr.es's password:
Last login: Tue Feb 25 20:53:07 2020 from vpn-s245207.ugr.es
[DanielMonjasMiguelez e1estudiante18@atcgrid:~] 2020-02-25 martes
$srun -p ac --account ac lscpu
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:               1600.000
CPU max MHz:           2401,0000
CPU min MHz:           1600,0000
BogoMIPS:              4800.38
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
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dt
s acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep
_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16
 xtpr pdcm pcid dca sse4_1 sse4_2 popcnt lahf_lm epb ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriority ep
t vpid dtherm ida arat spec_ctrl intel_stibp flush_l1d
[DanielMonjasMiguelez e1estudiante18@atcgrid:~] 2020-02-25 martes
$
```

(b) ¿Cuántos cores físicos y cuántos cores lógicos tienen los nodos de cómputo de atcgrid y el PC? Razonar las respuestas

RESPUESTA: En mi PC tengo 4 cores físicos y 8 cores lógicos, ya que en el campo de core(s) per socket pone que dispongo de 4 cores por cada socket. En el campo de socket se me indica que mi ordenador dispone de un único socket. Por último Thread(s) per core indica el numero de hebras de procesamiento por core físico, como por cada core dispongo de 2 hebras de procesamiento el total de cores lógicos es 4*2=8, como indica el campo CPU(s). El nodo de cómputo de atcgrid tiene 6 cores físicos y 24 lógicos, ya que tiene 2 sockets, y 2 hilos de procesamiento por core, es decir, 6(cores físicos)*2(sockets)*2(threads)=24 cores lógicos en total.

2. Compilar y ejecutar en el PC el código `HelloOMP.c` del seminario (recordar que se debe usar un directorio independiente para cada ejercicio dentro de `bp0` que contenga todo lo utilizado, implementado o generado durante el desarrollo del mismo, para el presente ejercicio el directorio sería `ejer2`, como se indica en las normas de prácticas).

(a) Adjuntar capturas de pantalla que muestren la compilación y ejecución en el PC.

RESPUESTA:

```
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer2] 2020-02-28 vi
ernes
$gcc -O2 -fopenmp -o HelloOMP HelloOMP.c
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer2] 2020-02-28 vi
ernes
$./HelloOMP
(0:!!!Hello world!!!)(7:!!!Hello world!!!)(3:!!!Hello world!!!)(2:!!!Hello world!!!)(5:!!!Hello world!!!)
(4:!!!Hello world!!!)(1:!!!Hello world!!!)(6:!!!Hello world!!!)[DanielMonjasMiguelez daniel@daniel-XPS-15
-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer2] 2020-02-28 viernes
$
```

(b) Justificar el número de "Hello world" que se imprimen en pantalla teniendo en cuenta la salida que devuelve `lscpu`.

RESPUESTA: Aparecen un total de 8 “Hello World” ya que hay tantos como procesadores lógicos tiene mi PC, donde hemos visto antes que tiene 8 cores lógicos.

3. Copiar el ejecutable de `HelloOMP.c` que ha generado anteriormente y que se encuentra en el directorio `ejer2 del` PC al directorio `ejer2` de su home en el *front-end* de atcgrid. Ejecutar este código en un nodo de cómputo de atcgrid a través de cola ac del gestor de colas (no use ningún *script*) utilizando directamente en línea de comandos:

(a) `srun -p ac –n1 --cpus-per-task=12 --hint=nomultithread HelloOMP`

Adjuntar capturas de pantalla que muestren el envío a la cola de la ejecución y el resultado de esta ejecución tal y como la devuelve el gestor de colas.

RESPUESTA: La primera imagen es del envío del ejecutable, y la segunda de su ejecución en un nodo de cómputo

```
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer2] 2020-02-28 vi
ernes
$sftp e1estudiante18@atcgrid.ugr.es
e1estudiante18@atcgrid.ugr.es's password:
Connected to atcgrid.ugr.es.
sftp> put He
HelloOMP     HelloOMP.c
sftp> put HelloOMP bp0/ejer2/
Uploading HelloOMP to /home/e1estudiante18/bp0/ejer2/HelloOMP
HelloOMP                                                      100% 8688   156.5KB/s   00:00
sftp>
```

```
[DanielMonjasMiguelez e1estudiante18@atcgrid:~/bp0/ejer2] 2020-02-25 martes
$srun -p ac -n1 --cpus-per-task=12 --hint=nomultithread HelloOMP
(3:!!!Hello world!!!)(2:!!!Hello world!!!)(8:!!!Hello world!!!)(4:!!!Hello world!!!)(0:!!!Hello world!!!)
(5:!!!Hello world!!!)(11:!!!Hello world!!!)(9:!!!Hello world!!!)(6:!!!Hello world!!!)(10:!!!Hello world!!
!)(1:!!!Hello world!!!)(7:!!!Hello world!!!)[DanielMonjasMiguelez e1estudiante18@atcgrid:~/bp0/ejer2] 202
0-02-25 martes
$
```

(b) `srun -p ac -n1 --cpus-per-task=24 HelloOMP`

Adjuntar capturas de pantalla que muestren el envío a la cola de la ejecución y el resultado de esta ejecución tal y como la devuelve el gestor de colas.

RESPUESTA:

```
[DanielMonjasMiguelez e1estudiante18@atcgrid:~/bp0/ejer2] 2020-02-25 martes
$ls
HelloOMP
[DanielMonjasMiguelez e1estudiante18@atcgrid:~/bp0/ejer2] 2020-02-25 martes
$srun -p ac -n1 --cpus-per-task=24 HelloOMP
(20:!!!Hello world!!!)(22:!!!Hello world!!!)(9:!!!Hello world!!!)(11:!!!Hello world!!!)(3:!!!Hello world!
!!)(2:!!!Hello world!!!)(17:!!!Hello world!!!)(6:!!!Hello world!!!)(15:!!!Hello world!!!)(12:!!!Hello wor
ld!!!)(10:!!!Hello world!!!)(19:!!!Hello world!!!)(4:!!!Hello world!!!)(0:!!!Hello world!!!)(23:!!!Hello
world!!!)(21:!!!Hello world!!!)(5:!!!Hello world!!!)(1:!!!Hello world!!!)(16:!!!Hello world!!!)(7:!!!Hell
o world!!!)(14:!!!Hello world!!!)(18:!!!Hello world!!!)(8:!!!Hello world!!!)(13:!!!Hello world!!!)[Daniel
MonjasMiguelez e1estudiante18@atcgrid:~/bp0/ejer2] 2020-02-25 martes
$
```

(c) `srun -p ac –n1 HelloOMP`

Adjuntar capturas de pantalla que muestren el envío a la cola de la ejecución y el resultado de esta ejecución tal y como la devuelve el gestor de colas.

RESPUESTA:

```
[DanielMonjasMiguelez e1estudiante18@atcgrid:~/bp0/ejer2] 2020-02-25 martes
$srun -p ac -n1 HelloOMP
(0:!!!Hello world!!!)(1:!!!Hello world!!!)[DanielMonjasMiguelez e1estudiante18@atcgrid:~/bp0/ejer2] 2020-
02-25 martes
$
```

(d) ¿Qué orden `srun` usaría para que HelloOMP utilice los 12 cores físicos de un nodo de cómputo de atcgrid (se debe imprimir un único mensaje desde cada uno de ellos, en total, 12)?

RESPUESTA: Usaría la opción a), en ella se indica que se usen `--cpus-per-task=12`, es decir, un total de 12 cpus, y con la opción `--hint=nomultithread` se indica que se quiere que estos cores sean físicos y no lógicos, pues slurm por defecto considere cpus lógicos en la opcion `--cpus-per-task`.

4. Modificar en su PC `HelloOMP.c` para que se imprima "world" en un `printf` distinto al usado para "Hello", en ambos `printf` se debe imprimir el identificador del thread que escribe en pantalla. Nombrar al código resultante `HelloOMP2.c`. Compilar este nuevo código en el PC y ejecutarlo. Copiar el fichero ejecutable resultante al front-end de atcgrid (directorio `ejer4`). Ejecutar el código en un nodo de cómputo de atcgrid usando el script `script_helloomp.sh` del seminario (el nombre del ejecutable en el script debe ser `HelloOMP2`).

(a) Utilizar: `sbatch -p ac –n1 --cpus-per-task=12 --hint=nomultithread script_helloomp.sh`. Adjuntar capturas de pantalla que muestren el nuevo código, la compilación, el envío a la cola de la ejecución y el resultado de esta ejecución tal y como la devuelve el gestor de colas.

RESPUESTA:

HelloOMP2.c
~/Escritorio/Daniel/AC/PRACTICAS/PRACTICA 1/ejer4

```
#include <stdio.h>
#include <omp.h>

int main(void) {

#pragma omp parallel
        printf ("(%d:!!!Hello ", omp_get_thread_num());

#pragma opm parallel
        printf("%d: world!!!", omp_get_thread_num());

return (0);

}
```

```
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer4] 2020-02-28 vi
ernes
$gcc -O2 -fopenmp -o HelloOMP2 HelloOMP2.c
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer4] 2020-02-28 vi
ernes
$./HelloOMP2
(6:!!!Hello (0:!!!Hello (7:!!!Hello (3:!!!Hello (4:!!!Hello (2:!!!Hello (1:!!!Hello (5:!!!Hello 0: world!
!![DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer4] 2020-02-28
viernes
$
```

```
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer4] 2020-02-28 vi
ernes
$gcc -O2 -fopenmp -o HelloOMP2 HelloOMP2.c
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer4] 2020-02-28 vi
ernes
$./HelloOMP2
(6:!!!Hello (0:!!!Hello (7:!!!Hello (3:!!!Hello (4:!!!Hello (2:!!!Hello (1:!!!Hello (5:!!!Hello 0: world!
!![DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer4] 2020-02-28
viernes
$sftp e1estudiante18@atcgrid.ugr.es
e1estudiante18@atcgrid.ugr.es's password:
Connected to atcgrid.ugr.es.
sftp> puts H
HelloOMP2    HelloOMP2.c
sftp> puts HelloOMP2 bp0/ejer4
Invalid command.
sftp> put HelloOMP2 bp0/ejer4
Uploading HelloOMP2 to /home/e1estudiante18/bp0/ejer4/HelloOMP2
HelloOMP2                                                     100% 8688   159.7KB/s   00:00
sftp>
```

```
[DanielMonjasMiguelez e1estudiante18@atcgrid:~/bp0/ejer4] 2020-02-26 miércoles
$sbatch -p ac -n1 --cpus-per-task=12 --hint=nomultithread script_helloomp.sh
Submitted batch job 10930
[DanielMonjasMiguelez e1estudiante18@atcgrid:~/bp0/ejer4] 2020-02-26 miércoles
$ls
HelloOMP2  script_helloomp.sh  slurm-10845.out  slurm-10930.out
[DanielMonjasMiguelez e1estudiante18@atcgrid:~/bp0/ejer4] 2020-02-26 miércoles
$rm slurm-10845.out
[DanielMonjasMiguelez e1estudiante18@atcgrid:~/bp0/ejer4] 2020-02-26 miércoles
$cat slurm-10930.out
Id. usuario del trabajo: e1estudiante18
Id. del trabajo: 10930
Nombre del trabajo especificado por el usuario: helloOMP2
Directorio trabajo (en el que se ejecuta el script): /home/e1estudiante18/bp0/ejer4
Cola: ac
Nodo que ejecuta este trabajo: atcgrid
Nº de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgrid1
CPUs por nodo:

 1. Ejecución helloOMP una vez sin cambiar nº de threads (valor por defector):

(0:!!!Hello (7:!!!Hello (2:!!!Hello (10:!!!Hello (1:!!!Hello (5:!!!Hello (6:!!!Hello (3:!!!Hello (11:!!!H
ello (8:!!!Hello (4:!!!Hello (9:!!!Hello 0: world!!!
 2. Ejecución helloOMP varias veces con distinto nº de threads:


 - Para 12 threads:
(0:!!!Hello (7:!!!Hello (4:!!!Hello (11:!!!Hello (5:!!!Hello (10:!!!Hello (3:!!!Hello (8:!!!Hello (1:!!!H
ello (9:!!!Hello (6:!!!Hello (2:!!!Hello 0: world!!!
 - Para 6 threads:
(1:!!!Hello (5:!!!Hello (2:!!!Hello (0:!!!Hello (3:!!!Hello (4:!!!Hello 0: world!!!
 - Para 3 threads:
(1:!!!Hello (2:!!!Hello (0:!!!Hello 0: world!!!
 - Para 1 threads:
(0:!!!Hello 0: world!!![DanielMonjasMiguelez e1estudiante18@atcgrid:~/bp0/ejer4] 2020-02-26 miércoles
$
```

(b) ¿Qué nodo de cómputo de atcgrid ha ejecutado el script? Explicar cómo ha obtenido esta información.

RESPUESTA: El nodo que ejecuta el trabajo es atcgrid1. He buscado en internet las variables de entorno de entrada de slurm y que información almacena cada una, llegando a que la variable `SLURM_JOB_NODELIST` indica los nodos que se encargan de la ejecución de la tarea.

NOTA: Utilizar siempre con `sbatch` las opciones `–n1` y `--cpus-per-task`, `--exclusive` y, para usar cores físicos y no lógicos, no olvide incluir `--hint=nomultithread`. Utilizar siempre con `srun`, si lo usa fuera de un script, las opciones `–n1` y `--cpus-per-task` y, para usar cores físicos y no lógicos, no olvide incluir `--hint=nomultithread`. Recordar que los `srun` dentro de un script heredan las opciones utilizadas en el `sbatch` que se usa para enviar el script a la cola slurm. Se recomienda usar `sbatch` en lugar de `srun` para enviar trabajos a ejecutar a través slurm porque éste último deja bloqueada la ventana hasta que termina la ejecución, mientras que usando `sbatch` la ejecución se realiza en segundo plano.

Parte II. Resto de ejercicios

5. Generar en el PC 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`). El comentario inicial del código muestra la orden para compilar (siempre hay que usar `–O2` al compilar como se indica en las normas de prácticas). Incorporar volcados de pantalla que demuestren la compilación y la ejecución correcta del código en el PC (leer lo indicado al respecto en las normas de prácticas).

RESPUESTA:

```
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer5] 2020-02-28 vi
ernes
$gcc -O2 SumaVectoresC.c -o SumaVectoresC -lrt
SumaVectoresC.c: In function ‘main’:
SumaVectoresC.c:45:33: warning: format ‘%u’ expects argument of type ‘unsigned int’, but argument 3 has t
ype ‘long unsigned int’ [-Wformat=]
   printf("Tamaño Vectores:%u (%u B)\n",N, sizeof(unsigned int));
                                ~^
                                %lu
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer5] 2020-02-28 vi
ernes
$gcc -O2 -S SumaVectoresC.c -lrt
SumaVectoresC.c: In function ‘main’:
SumaVectoresC.c:45:33: warning: format ‘%u’ expects argument of type ‘unsigned int’, but argument 3 has t
ype ‘long unsigned int’ [-Wformat=]
   printf("Tamaño Vectores:%u (%u B)\n",N, sizeof(unsigned int));
                                ~^
                                %lu
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer5] 2020-02-28 vi
ernes
$./SumaVectoresC 300000
Tamaño Vectores:300000 (4 B)
Tiempo:0.000814667       / Tamaño Vectores:300000       / V1[0]+V2[0]=V3[0](30000.000000+30000.000000=600
00.000000) / / V1[299999]+V2[299999]=V3[299999](59999.900000+0.100000=60000.000000) /
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer5] 2020-02-28 vi
ernes
$
```

6. En el código del Listado 1 se utiliza la función `clock_gettime()` para obtener el tiempo de ejecución del trozo de código que calcula la suma de vectores. El código se imprime la variable `ncgt`,
(a) ¿qué contiene esta variable?
RESPUESTA: La variable ncgt imprime el tiempo que tarda en ejecutarse la suma de los vectores, es decir, únicamente el tiempo que tarda en ejecutarse el bucle for que realiza la suma de los dos vectores. Esta variable contiene la diferencia de tiempo entre el instante antes del bucle for y el instante justo después del bucle for.
(b) ¿en qué estructura de datos devuelve `clock_gettime()` la información de tiempo (indicar el tipo de estructura de datos, describir la estructura de datos, e indicar los tipos de datos que usa)?

> **RESPUESTA**: Los devuelve en un struct timespec, se trata de una estructura de la biblioteca <time.h>, la cual tiene dos objetos miembros, tv_secs que es de tipo time_t, y contiene los segundos, y tv_nsec que es de tipo long y contiene los nanosegundos.

(c) ¿qué información devuelve exactamente la función `clock_gettime()`en la estructura de datos descrita en el apartado (b)? ¿qué representan los valores numéricos que devuelve?

RESPUESTA: Devuelven el tiempo al que apunta el reloj especificado en el instante de la llamada a la función, es decir, la estructura de datos pasa a contener en segundos y nano segundos el tiempo que indica el reloj especificado como primer argumento, desde Epoch, el 1 de enero de 1970 a las 00:00:00 UT, ya que el reloj indicado es `CLOCK_REAL_TIME`.

7. Rellenar una tabla como la Tabla 1 en una hoja de cálculo con los tiempos de ejecución del código del Listado 1 para vectores locales, globales y dinámicos. Obtener estos resultados usando scripts (partir del script que hay en el seminario). Debe haber una tabla para atcgrid y otra para su PC en la hoja de cálculo. En la columna “Bytes de un vector” hay que poner el total de bytes reservado para un vector. (NOTA: Se recomienda usar en la hoja de cálculo el mismo separador para decimales que usan los códigos al imprimir. Este separador se puede modificar en la hoja de cálculo.)

RESPUESTA:

Tabla 1 . Copiar la tabla de la hoja de cálculo utilizada

Tabla para tiempos atcgrid

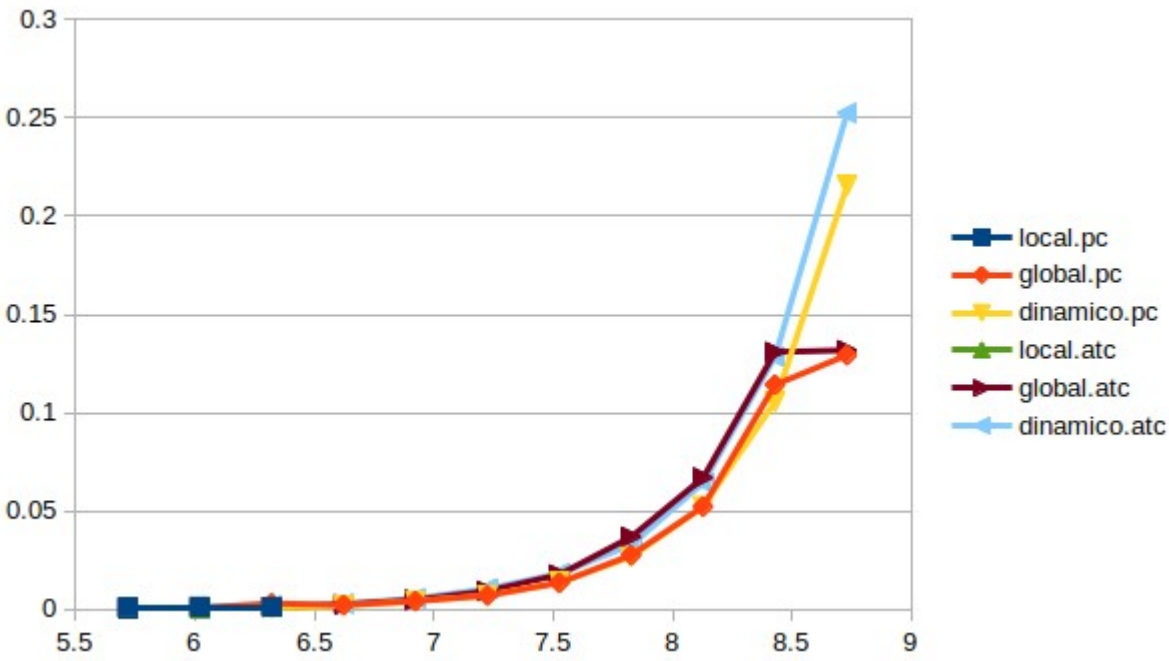
Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000486922	0.000468922	0.000476157
131072	1048576	0.000965785	0.000606296	0.000941882
262144	2097152	0.001907049	0.001227540	0.001897604
524288	4194304		0.002589069	0.002919573
1048576	8388608		0.004914791	0.005638114
2097152	16777216		0.009371723	0.010573060
4194304	33554432		0.017729822	0.018256862
8388608	67108864		0.036905891	0.032870645
16777216	134217728		0.066995720	0.064372822
33554432	268435456		0.130959000	0.128598930
67108864	536870912		0.131847796	0.252460870

Tabla para tiempos pc

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000761630	0.000773551	0.000773139
131072	1048576	0.001218500	0.001137677	0.001161584
262144	2097152	0.001138756	0.003090012	0.000801025
524288	4194304		0.002266604	0.001844603
1048576	8388608		0.004219419	0.004355303
2097152	16777216		0.007121598	0.007061789
4194304	33554432		0.013525487	0.013874854
8388608	67108864		0.027661880	0.026818883
16777216	134217728		0.052327612	0.052400737
33554432	268435456		0.114293997	0.105666481
67108864	536870912		0.129311153	0.215612194

1. Con ayuda de la hoja de cálculo representar **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 (por tanto, los valores de la segunda columna de la tabla, que están en escala logarítmica, deben estar en el eje x). Utilizar escala logarítmica en el eje de ordenadas (eje y). ¿Hay diferencias en los tiempos de ejecución?

   RESPUESTA: Se puede apreciar que los tiempos de ejecución en mi PC, especialmente para pocos bytes por vector son un poco mejores que los alcanzados por atcgrid.



2. **(a)** Cuando se usan vectores locales, ¿se obtiene error para alguno de los tamaños?, ¿a qué cree que es debido lo que ocurre? (Incorporar volcados de pantalla como se indica en las normas de prácticas)

RESPUESTA: Se obtiene el error Segmentation fault (core dumped) a partir del tercer tamaño, es decir cuando los vectores tienen 524288 o más. Esto se deberá muy probablemente a que el tamaño de la pila que almacena las variables de las funciones (main en este caso) se verá superado por el tamaño del vector que se quiere reservar.

Captura atcgrid

```
[DanielMonjasMiguelez e1estudiante18@atcgrid:~/bp0/ejer7] 2020-02-28 viernes
$gcc -O2 SumaVectoresC.c -o SumaVectoresC -lrt
[DanielMonjasMiguelez e1estudiante18@atcgrid:~/bp0/ejer7] 2020-02-28 viernes
$gcc -O2 -S SumaVectoresC.c -lrt
[DanielMonjasMiguelez e1estudiante18@atcgrid:~/bp0/ejer7] 2020-02-28 viernes
$srun -p ac --account ac bash script_ejecucion
Tamaño Vectores:65536 (4 B)
Tiempo:0.000370194       / 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) /
Tamaño Vectores:131072 (4 B)
Tiempo:0.000835120       / Tamaño Vectores:131072       / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=262
14.400000) / / V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tamaño Vectores:262144 (4 B)
Tiempo:0.001694137       / Tamaño Vectores:262144       / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=524
28.800000) / / V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Exited with exit code 139
[DanielMonjasMiguelez e1estudiante18@atcgrid:~/bp0/ejer7] 2020-02-28 viernes
$
```

Caputra PC

```
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer5] 2020-02-28 vi
ernes
$gcc -O2 SumaVectoresC.c -o SumaVectoresC -lrt
SumaVectoresC.c: In function ‘main’:
SumaVectoresC.c:45:33: warning: format ‘%u’ expects argument of type ‘unsigned int’, but argument 3 has t
ype ‘long unsigned int’ [-Wformat=]
   printf("Tamaño Vectores:%u (%u B)\n",N, sizeof(unsigned int));
                                ~^
                                %lu
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer5] 2020-02-28 vi
ernes
$gcc -O2 -S SumaVectoresC.c -lrt
SumaVectoresC.c: In function ‘main’:
SumaVectoresC.c:45:33: warning: format ‘%u’ expects argument of type ‘unsigned int’, but argument 3 has t
ype ‘long unsigned int’ [-Wformat=]
   printf("Tamaño Vectores:%u (%u B)\n",N, sizeof(unsigned int));
                                ~^
                                %lu
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer5] 2020-02-28 vi
ernes
$bash script_ejecucion
Tamaño Vectores:65536 (4 B)
Tiempo:0.000736244       / 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) /
Tamaño Vectores:131072 (4 B)
Tiempo:0.001045718       / Tamaño Vectores:131072       / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=262
14.400000) / / V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tamaño Vectores:262144 (4 B)
Tiempo:0.000750620       / Tamaño Vectores:262144       / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=524
28.800000) / / V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tamaño Vectores:524288 (4 B)
script_ejecucion: line 6:  3179 Segmentation fault      (core dumped) ./SumaVectoresC $P
Tamaño Vectores:1048576 (4 B)
script_ejecucion: line 6:  3181 Segmentation fault      (core dumped) ./SumaVectoresC $P
Tamaño Vectores:2097152 (4 B)
script_ejecucion: line 6:  3183 Segmentation fault      (core dumped) ./SumaVectoresC $P
Tamaño Vectores:4194304 (4 B)
script_ejecucion: line 6:  3185 Segmentation fault      (core dumped) ./SumaVectoresC $P
Tamaño Vectores:8388608 (4 B)
script_ejecucion: line 6:  3187 Segmentation fault      (core dumped) ./SumaVectoresC $P
Tamaño Vectores:16777216 (4 B)
script_ejecucion: line 6:  3189 Segmentation fault      (core dumped) ./SumaVectoresC $P
Tamaño Vectores:33554432 (4 B)
script_ejecucion: line 6:  3191 Segmentation fault      (core dumped) ./SumaVectoresC $P
Tamaño Vectores:67108864 (4 B)
script_ejecucion: line 6:  3193 Segmentation fault      (core dumped) ./SumaVectoresC $P
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer5] 2020-02-28 vi
ernes
$
```

(b) Cuando se usan vectores globales, ¿se obtiene error para alguno de los tamaños?, ¿a qué cree que es debido lo que ocurre? (Incorporar volcados de pantalla como se indica en las normas de prácticas)

RESPUESTA: En mi caso no se produce ningún error, pero si ocurre que en la última ejecución del programa ya no crece más el tamaño del vector, y es que en el código se establece que si el parámetro que se introduce es mayor que un umbral MAX, se establece a MAX el parámetro introducido, de forma que como el último tamaño de vector introducido supera el umbral se establece al máximo valor permitido que es 3554432.

Captura PC

```
$gcc -O2 SumaVectoresC.c -o SumaVectoresC -lrt
SumaVectoresC.c: In function ‘main’:
SumaVectoresC.c:45:33: warning: format ‘%u’ expects argument of type ‘unsigned int’, but argument 3 has t
ype ‘long unsigned int’ [-Wformat=]
   printf("Tamaño Vectores:%u (%u B)\n",N, sizeof(unsigned int));
                               ~^
                               %lu
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer5] 2020-02-28 vi
ernes
$gcc -O2 -S SumaVectoresC.c -lrt
SumaVectoresC.c: In function ‘main’:
SumaVectoresC.c:45:33: warning: format ‘%u’ expects argument of type ‘unsigned int’, but argument 3 has t
ype ‘long unsigned int’ [-Wformat=]
   printf("Tamaño Vectores:%u (%u B)\n",N, sizeof(unsigned int));
                               ~^
                               %lu
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer5] 2020-02-28 vi
ernes
$bash script_ejecucion
Tamaño Vectores:65536 (4 B)
Tiempo:0.000760022       / 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) /
Tamaño Vectores:131072 (4 B)
Tiempo:0.001463666       / Tamaño Vectores:131072       / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=262
14.400000) / / V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tamaño Vectores:262144 (4 B)
Tiempo:0.000813523       / Tamaño Vectores:262144       / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=524
28.800000) / / V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tamaño Vectores:524288 (4 B)
Tiempo:0.001571045       / Tamaño Vectores:524288       / V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104
857.600000) / / V1[524287]+V2[524287]=V3[524287](104857.500000+0.100000=104857.600000) /
Tamaño Vectores:1048576 (4 B)
Tiempo:0.003760231       / Tamaño Vectores:1048576      / V1[0]+V2[0]=V3[0](104857.600000+104857.600000=2
09715.200000) / / V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000) /
Tamaño Vectores:2097152 (4 B)
Tiempo:0.007010698       / Tamaño Vectores:2097152      / V1[0]+V2[0]=V3[0](209715.200000+209715.200000=4
19430.400000) / / V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000) /
Tamaño Vectores:4194304 (4 B)
Tiempo:0.013741388       / Tamaño Vectores:4194304      / V1[0]+V2[0]=V3[0](419430.400000+419430.400000=8
38860.800000) / / V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tamaño Vectores:8388608 (4 B)
Tiempo:0.026673044       / Tamaño Vectores:8388608      / V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1
677721.600000) / / V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000) /
Tamaño Vectores:16777216 (4 B)
Tiempo:0.052451063       / 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) /
Tamaño Vectores:33554432 (4 B)
Tiempo:0.106184163       / 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) /
Tamaño Vectores:67108864 (4 B)
Tiempo:0.104579243       / 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) /
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer5] 2020-02-28 vi
ernes
$
```

Captura atcgrid

```
[DanielMonjasMiguelez e1estudiante18@atcgrid:~/bp0/ejer7] 2020-02-28 viernes
$srun -p ac --account ac bash script_ejecucion
Tamaño Vectores:65536 (4 B)
Tiempo:0.000483933       / 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) /
Tamaño Vectores:131072 (4 B)
Tiempo:0.000630062       / Tamaño Vectores:131072       / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=262
14.400000) / / V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tamaño Vectores:262144 (4 B)
Tiempo:0.001341208       / Tamaño Vectores:262144       / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=524
28.800000) / / V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tamaño Vectores:524288 (4 B)
Tiempo:0.002689615       / Tamaño Vectores:524288       / V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104
857.600000) / / V1[524287]+V2[524287]=V3[524287](104857.500000+0.100000=104857.600000) /
Tamaño Vectores:1048576 (4 B)
Tiempo:0.005417977       / Tamaño Vectores:1048576      / V1[0]+V2[0]=V3[0](104857.600000+104857.600000=2
09715.200000) / / V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000) /
Tamaño Vectores:2097152 (4 B)
Tiempo:0.010144517       / Tamaño Vectores:2097152      / V1[0]+V2[0]=V3[0](209715.200000+209715.200000=4
19430.400000) / / V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000) /
Tamaño Vectores:4194304 (4 B)
Tiempo:0.018416382       / Tamaño Vectores:4194304      / V1[0]+V2[0]=V3[0](419430.400000+419430.400000=8
38860.800000) / / V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tamaño Vectores:8388608 (4 B)
Tiempo:0.035564382       / Tamaño Vectores:8388608      / V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1
677721.600000) / / V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000) /
Tamaño Vectores:16777216 (4 B)
Tiempo:0.067239539       / 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) /
Tamaño Vectores:33554432 (4 B)
Tiempo:0.130535322       / 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) /
Tamaño Vectores:67108864 (4 B)
Tiempo:0.131256119       / 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) /
[DanielMonjasMiguelez e1estudiante18@atcgrid:~/bp0/ejer7] 2020-02-28 viernes
$
```

(c) Cuando se usan vectores dinámicos, ¿se obtiene error para alguno de los tamaños?, ¿a qué cree que es debido lo que ocurre? (Incorporar volcados de pantalla como se indica en las normas de prácticas)

RESPUESTA: No se obtiene ningún error, y es que el tamaño de los vectores se reserva dinámicamente, es decir, durante la ejecución del programa se solicita al sistema operativo memoria para dichos vectores, de forma que, siempre que el computador no se quede sin memoria libre o alcance el limite de memoria (recordemos que el computador siempre reserva un límite de memoria para tener un colchón) se podrán albergar los tamaños de vector que se pidan.

Captura PC

```
$gcc -O2 SumaVectoresC.c -o SumaVectoresC
SumaVectoresC.c: In function ‘main’:
SumaVectoresC.c:45:33: warning: format ‘%u’ expects argument of type ‘unsigned int’, but argument 3 has t
ype ‘long unsigned int’ [-Wformat=]
   printf("Tamaño Vectores:%u (%u B)\n",N, sizeof(unsigned int));
                                ~^
                                %lu
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer5] 2020-02-28 vi
ernes
$gcc -O2 -S SumaVectoresC.c -lrt
SumaVectoresC.c: In function ‘main’:
SumaVectoresC.c:45:33: warning: format ‘%u’ expects argument of type ‘unsigned int’, but argument 3 has t
ype ‘long unsigned int’ [-Wformat=]
   printf("Tamaño Vectores:%u (%u B)\n",N, sizeof(unsigned int));
                                ~^
                                %lu
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer5] 2020-02-28 vi
ernes
$bash script_ejecucion
Tamaño Vectores:65536 (4 B)
Tiempo:0.000829333       / 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) /
Tamaño Vectores:131072 (4 B)
Tiempo:0.000922371       / Tamaño Vectores:131072       / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=262
14.400000) / / V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tamaño Vectores:262144 (4 B)
Tiempo:0.000774019       / Tamaño Vectores:262144       / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=524
28.800000) / / V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tamaño Vectores:524288 (4 B)
Tiempo:0.001766168       / Tamaño Vectores:524288       / V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104
857.600000) / / V1[524287]+V2[524287]=V3[524287](104857.500000+0.100000=104857.600000) /
Tamaño Vectores:1048576 (4 B)
Tiempo:0.003232758       / Tamaño Vectores:1048576      / V1[0]+V2[0]=V3[0](104857.600000+104857.600000=2
09715.200000) / / V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000) /
Tamaño Vectores:2097152 (4 B)
Tiempo:0.006847211       / Tamaño Vectores:2097152      / V1[0]+V2[0]=V3[0](209715.200000+209715.200000=4
19430.400000) / / V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000) /
Tamaño Vectores:4194304 (4 B)
Tiempo:0.014013133       / Tamaño Vectores:4194304      / V1[0]+V2[0]=V3[0](419430.400000+419430.400000=8
38860.800000) / / V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tamaño Vectores:8388608 (4 B)
Tiempo:0.027160877       / Tamaño Vectores:8388608      / V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1
677721.600000) / / V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000) /
Tamaño Vectores:16777216 (4 B)
Tiempo:0.053245946       / 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) /
Tamaño Vectores:33554432 (4 B)
Tiempo:0.105360121       / 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) /
Tamaño Vectores:67108864 (4 B)
Tiempo:0.232062564       / 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) /
```

Captura atcgrid

```
[DanielMonjasMiguelez e1estudiante18@atcgrid:~/bp0/ejer7] 2020-02-28 viernes
$srun -p ac --account ac bash script_ejecucion
Tamaño Vectores:65536 (4 B)
Tiempo:0.000333619       / 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) /
Tamaño Vectores:131072 (4 B)
Tiempo:0.000744794       / Tamaño Vectores:131072       / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=262
14.400000) / / V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tamaño Vectores:262144 (4 B)
Tiempo:0.001579052       / Tamaño Vectores:262144       / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=524
28.800000) / / V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tamaño Vectores:524288 (4 B)
Tiempo:0.002885282       / Tamaño Vectores:524288       / V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104
857.600000) / / V1[524287]+V2[524287]=V3[524287](104857.500000+0.100000=104857.600000) /
Tamaño Vectores:1048576 (4 B)
Tiempo:0.005644124       / Tamaño Vectores:1048576      / V1[0]+V2[0]=V3[0](104857.600000+104857.600000=2
09715.200000) / / V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000) /
Tamaño Vectores:2097152 (4 B)
Tiempo:0.010380735       / Tamaño Vectores:2097152      / V1[0]+V2[0]=V3[0](209715.200000+209715.200000=4
19430.400000) / / V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000) /
Tamaño Vectores:4194304 (4 B)
Tiempo:0.018443185       / Tamaño Vectores:4194304      / V1[0]+V2[0]=V3[0](419430.400000+419430.400000=8
38860.800000) / / V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tamaño Vectores:8388608 (4 B)
Tiempo:0.034930985       / Tamaño Vectores:8388608      / V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1
677721.600000) / / V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000) /
Tamaño Vectores:16777216 (4 B)
Tiempo:0.064333087       / 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) /
Tamaño Vectores:33554432 (4 B)
Tiempo:0.128087695       / 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) /
Tamaño Vectores:67108864 (4 B)
Tiempo:0.254783476       / 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) /
[DanielMonjasMiguelez e1estudiante18@atcgrid:~/bp0/ejer7] 2020-02-28 viernes
$
```

3. **(a)** ¿Cuál es el máximo valor que se puede almacenar en la variable N teniendo en cuenta su tipo? Razonar respuesta.
RESPUESTA: Teniendo encuenta que N es de tipo unsigned int, su tamaño es de 4B, es decir, 32 bits, luego se pueden conseguir un valor maximo de N^(32)-1, pues el 0, está incluido en los valores que se pueden obtener con estos 32bits.
(b) Modificar el código fuente C (en el PC) 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 y generar el ejecutable. ¿Qué ocurre? ¿A qué es debido? (Incorporar volcados de pantalla que muestren lo que ocurre)
RESPUESTA: Da error en la compilación, esto es debido a que lo más probable es que si bien con el primer vector no da error, la suma de los tres superará el espacio permitido por el SO para este proceso. Se requiere de truncamiento.
Captura PC

```
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer5] 2020-02-28 vi
ernes
$gcc -O2 SumaVectoresC.c -o SumaVectoresC -lrt
SumaVectoresC.c: In function ‘main’:
SumaVectoresC.c:45:33: warning: format ‘%u’ expects argument of type ‘unsigned int’, but argument 3 has t
ype ‘long unsigned int’ [-Wformat=]
   printf("Tamaño Vectores:%u (%u B)\n",N, sizeof(unsigned int));
                                ~^
                                %lu
/tmp/ccqIuz0E.o: In function `main':
SumaVectoresC.c:(.text.startup+0x76): relocation truncated to fit: R_X86_64_PC32 against symbol `v2' defi
ned in COMMON section in /tmp/ccqIuz0E.o
SumaVectoresC.c:(.text.startup+0xc9): relocation truncated to fit: R_X86_64_PC32 against symbol `v3' defi
ned in COMMON section in /tmp/ccqIuz0E.o
collect2: error: ld returned 1 exit status
[DanielMonjasMiguelez daniel@daniel-XPS-15-9570:~/Escritorio/Daniel/AC/PRACTICAS/bp0/ejer5] 2020-02-28 vi
ernes
$
```

Entrega del trabajo

Leer lo indicado en las normas de prácticas sobre la entrega del trabajo del bloque práctico en SWAD.

Listado 1 . Código C que suma dos vectores

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

 Para compilar usar (-lrt: real time library, no todas las versiones de gcc necesitan que se incluya
-lrt):
         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()

//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
   if (N<10) {
   printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%lu\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]);


   #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;
}
```