

<p>2º curso / 2º cuatr. Grado Ingeniería Informática</p>	<h2>Arquitectura de Computadores (AC)</h2> <h3>Cuaderno de prácticas.</h3> <h3>Bloque Práctico 0. Entorno de programación</h3> <p>Estudiante (nombre y apellidos): Carmen García ROmero Grupo de prácticas y profesor de prácticas: 2C-C1 Christian Morillas Fecha de entrega: 15/03/21 Fecha evaluación en clase: 16/03/21</p>
--	---

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 (PC = PC del aula de prácticas o su computador personal).

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 x se debe usar con sbatch/srun la opción `--cpus-per-task=x (-cx)`.
- En slurm, por defecto, cpu se refiere a cores lógicos (ej. en la opción `-c`), 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 `--ntasks=1` ó `(-n1)` en sbatch/srun.
- Para que no se ejecute más de un proceso en un nodo de cómputo 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 (partición slurm).
- Las opciones de sbatch se pueden especificar también dentro del *script* (usando `#SBATCH`, ver ejemplos en el script del seminario)

1. Ejecutar `lscpu` en el PC, en atcgrid4 (usar `-p ac4`) y en uno de los restantes nodos de cómputo (atcgrid1, atcgrid2 o atcgrid3, están en la cola ac). (Crear directorio **ej1**)

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

RESPUESTA:

```
CarmenGarciaRomero c1estudiante11@carmen:~/Escritorio/Facultad/2/2Cuatri/AC/BP0 2021-03-09 martes
$lscpu
Arquitectura: x86_64
modo(s) de operación de las CPUs: 32-bit, 64-bit
Orden de los bytes: Little Endian
Address sizes: 39 bits physical, 48 bits virtual
CPU(s): 12
Lista de la(s) CPU(s) en línea: 0-11
Hilo(s) de procesamiento por núcleo: 2
Núcleo(s) por «socket»: 6
«Socket(s)»: 1
Modo(s) NUMA: 1
ID de fabricante: GenuineIntel
Familia de CPU: 6
Modelo: 158
Nombre del modelo: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
Revisión: 13
CPU MHz: 800.035
CPU MHz máx.: 4500,0000
CPU MHz mín.: 800,0000
BogoMIPS: 5199.98
Virtualización: VT-x
Caché L1d: 192 KiB
Caché L1i: 192 KiB
Caché L2: 1,5 MiB
Caché L3: 12 MiB
CPU(s) del nodo NUMA 0: 0-11
Vulnerability Itlb multihit: KVM: Mitigation: VMX disabled
Vulnerability L1tf: Not affected
Vulnerability Mds: Not affected
Vulnerability Meltdown: Not affected
Vulnerability Spec store bypass: Mitigation; Speculative Store Bypass disabled via prctl and seccomp
Vulnerability Spectre v1: Mitigation; usercopy/swapgs barriers and __user pointer sanitization
Vulnerability Spectre v2: Mitigation; Enhanced IBRS, IBPB conditional, RSB filling
Vulnerability Srbds: Mitigation; TSX disabled
Vulnerability Tsx async abort: Not affected
Indicadores: 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 rep_good nopl xtopology non
```

Imagen : PC

```

CarmenGarciaRomero c1estudiante11@atcgrid:/etc 2021-03-09 martes
$srunc -p ac4 lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                64
On-line CPU(s) list:   0-63
Thread(s) per core:    2
Core(s) per socket:    16
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 85
Model name:            Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz
Stepping:              7
CPU MHz:               1171.508
CPU max MHz:           3200,0000
CPU min MHz:           800,0000
BogoMIPS:              4200.00
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              1024K
L3 cache:              22528K
NUMA node0 CPU(s):     0-15,32-47
NUMA node1 CPU(s):     16-31,48-63
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 rep_good nopl xtopology nonst
                        op_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid dca ss
                        e4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch epb cat_l3 cdp_l3
                        invpcid_single intel_ppin intel_pt ssbd mba ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi flexpriority ept vpid fsgsbas
                        e tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm cqm mpx rdt_a avx512f avx512dq rdseed adx smap clflushopt clwb avx
                        512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local dtherm ida arat pln pt
                        s pku ospke avx512_vnni md_clear spec_ctrl intel_stibp flush_l1d arch_capabilities

```

Imagen : atcgrid4

```

CarmenGarciaRomero c1estudiante11@atcgrid:~ 2021-03-09 martes
$srunc -p 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:              4799.93
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 dts 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_t
                        sc 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 ept vpid dtherm ida arat spec_ctrl intel_stibp flush_l1d

```

Imagen : atcgrid

(b) ¿Cuántos cores físicos y cuántos cores lógicos tiene atcgrid4?, ¿cuántos tienen atcgrid1, atcgrid2 y atcgrid3? y ¿cuántos tiene el PC? Razonar las respuestas

RESPUESTA:

	Cores físicos	Cores lógicos
atcgrid4	32	64
atcgrid1, atcgrid2, atcgrid3	12	24
pc	6	12

2. Compilar y ejecutar en el PC el código HelloOMP.c del seminario (recordar que, como se indica en las normas de prácticas, 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**).

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

RESPUESTA:

```
CarmenGarciaRomero ciestudiante11@carmen:~/Escritorio/Facultad/2/2Cuatri/AC/BP0/
ejer2 2021-03-11 jueves
$ ./HelloOMP
(6:!!!Hello world!!!)

(1:!!!Hello world!!!)

(11:!!!Hello world!!!)

(9:!!!Hello world!!!)

(7:!!!Hello world!!!)

(5:!!!Hello world!!!)

(3:!!!Hello world!!!)

(0:!!!Hello world!!!)

(4:!!!Hello world!!!)

(10:!!!Hello world!!!)

(8:!!!Hello world!!!)

(2:!!!Hello world!!!)
```

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

RESPUESTA: 12 apariciones, una por cada núcleo.

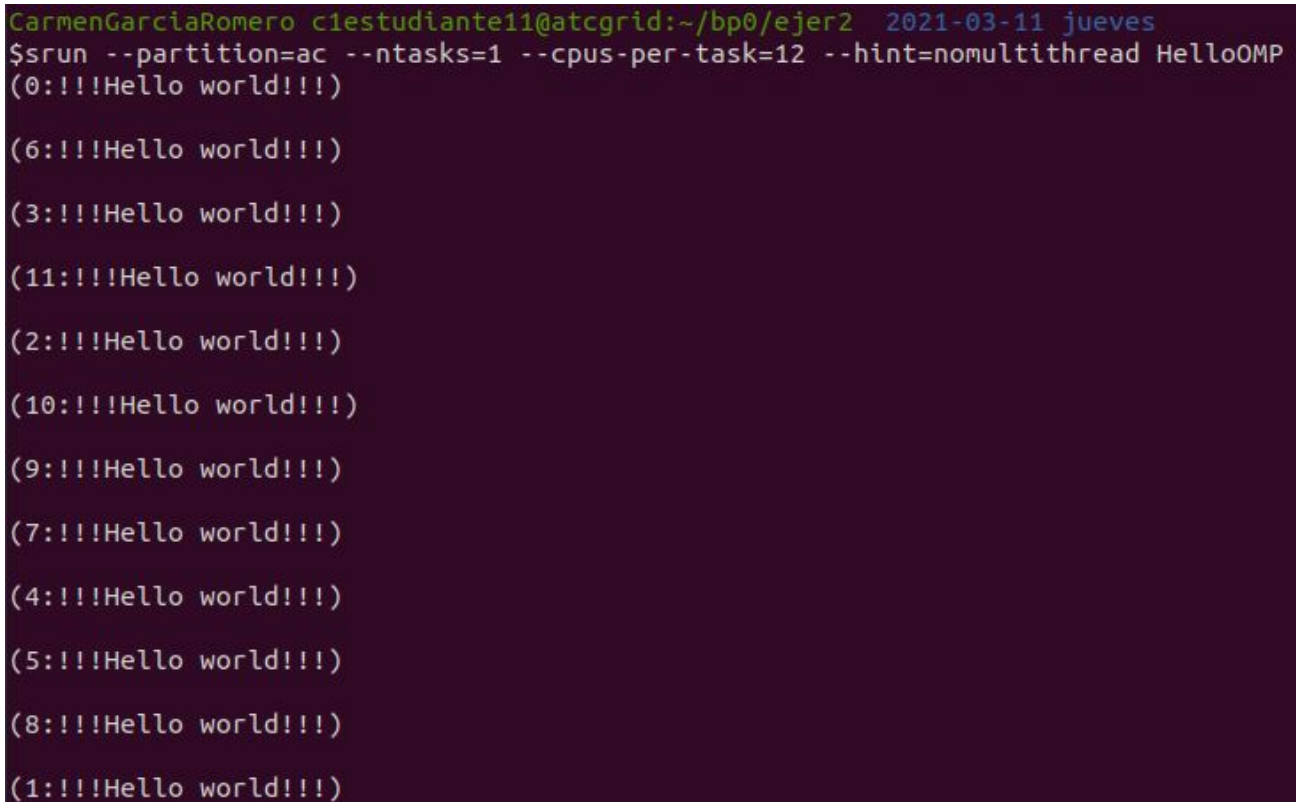
3. Copiar el ejecutable de HelloOMP.c que ha generado anteriormente y que se encuentra en el directorio ejer2 del PC al directorio ejer2 de su home en el *front-end* de atcgrid. Ejecutar este código en un nodo de cómputo de atcgrid (de 1 a 3) a través de cola ac del gestor de colas utilizando directamente en línea de comandos (no use ningún *script*):

(a) `srun --partition=ac --account=ac --ntasks=1 --cpus-per-task=12 --hint=nomultithread HelloOMP`

(Alternativa: `srun -pac -Aac -n1 -c12 --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:



```
CarmenGarciaRomero c1estudiante11@atcgrid:~/bp0/ejer2 2021-03-11 jueves
$srun --partition=ac --ntasks=1 --cpus-per-task=12 --hint=nomultithread HelloOMP
(0:!!!Hello world!!!)

(6:!!!Hello world!!!)

(3:!!!Hello world!!!)

(11:!!!Hello world!!!)

(2:!!!Hello world!!!)

(10:!!!Hello world!!!)

(9:!!!Hello world!!!)

(7:!!!Hello world!!!)

(4:!!!Hello world!!!)

(5:!!!Hello world!!!)

(8:!!!Hello world!!!)

(1:!!!Hello world!!!)
```

(b) `srun -pac -Aac -n1 -c24 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:

```
CarmenGarciaRomero c1estudiante11@atcgrid:~/bp0/ejer2 2021-03-11 jueves
$srunc -pac -Aac -n1 -c24 HelloOMP
(22:!!!Hello world!!!)

(12:!!!Hello world!!!)

(5:!!!Hello world!!!)

(20:!!!Hello world!!!)

(2:!!!Hello world!!!)

(19:!!!Hello world!!!)

(14:!!!Hello world!!!)

(17:!!!Hello world!!!)

(3:!!!Hello world!!!)

(0:!!!Hello world!!!)

(16:!!!Hello world!!!)

(7:!!!Hello world!!!)

(1:!!!Hello world!!!)

(23:!!!Hello world!!!)

(9:!!!Hello world!!!)

(15:!!!Hello world!!!)

(11:!!!Hello world!!!)

(13:!!!Hello world!!!)

(4:!!!Hello world!!!)

(18:!!!Hello world!!!)

(8:!!!Hello world!!!)

(6:!!!Hello world!!!)

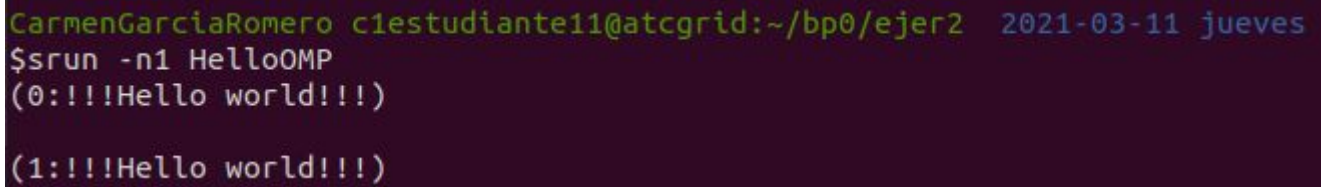
(10:!!!Hello world!!!)

(21:!!!Hello world!!!)
```

(c) `srun -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. ¿Qué partición se está usando?

RESPUESTA: ac



```
CarmenGarciaRomero c1estudiante11@atcgrid:~/bp0/ejer2 2021-03-11 jueves
$ srun -n1 HelloOMP
(0:!!!Hello world!!!)

(1:!!!Hello world!!!)
```

(d) ¿Qué orden `srun` usaría para que HelloOMP utilice todos los cores físicos de atcgrid4 (se debe imprimir un único mensaje desde cada uno de ellos)?

`srun -p ac4 -n2 -c16 --hint=nomultithread HelloOMP`

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 -pac -n1 -c12 --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.



```
1 #include <stdio.h>
2 #include <omp.h>
3
4 int main(void){
5 #pragma omp parallel
6 printf("(%d: !!!Hello)\n", omp_get_thread_num());
7 printf("(%d: world!!)\n", omp_get_thread_num());
8
9 return(0);
10 }
11 |
```

Imagen : Código HelloOMP2.c

```
CarmenGarciaRomero c1estudiante11@atcgrid:~/bp0/ejer4 2021-03-13 sábado
$ sbatch -pac -n1 -c12 --hint=nomultithread script_helloomp.sh
Submitted batch job 67795
CarmenGarciaRomero c1estudiante11@atcgrid:~/bp0/ejer4 2021-03-13 sábado
$ cat slurm-67795.out
Id. usuario del trabajo: c1estudiante11
Id. del trabajo: 67795
Nombre del trabajo especificado por usuario: helloOMP
Directorio de trabajo (en el que se ejecuta el script): /home/c1estudiante11/bp0
/ejer4
Cola: ac
Nodo que ejecuta este trabajo: atcgrid.ugr.es
No de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgrid1
CPUs por nodo: 24

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

(2: !!!Hello)
(8: !!!Hello)
(6: !!!Hello)
(4: !!!Hello)
(0: !!!Hello)
(10: !!!Hello)
(3: !!!Hello)
(9: !!!Hello)
(7: !!!Hello)
(5: !!!Hello)
(11: !!!Hello)
(1: !!!Hello)
(0: world!!)
```


2. Ejecución helloOMP varias veces con distinto no de threads:

- Para 12 threads:

```
(5: !!!Hello)
(10: !!!Hello)
(1: !!!Hello)
(4: !!!Hello)
(2: !!!Hello)
(0: !!!Hello)
(3: !!!Hello)
(9: !!!Hello)
(6: !!!Hello)
(7: !!!Hello)
(8: !!!Hello)
(11: !!!Hello)
(0: world!!)
```

- Para 6 threads:

```
(0: !!!Hello)
(1: !!!Hello)
(2: !!!Hello)
(4: !!!Hello)
(3: !!!Hello)
(5: !!!Hello)
(0: world!!)
```

- Para 3 threads:

```
(1: !!!Hello)
(0: !!!Hello)
(2: !!!Hello)
(0: world!!)
```

- Para 1 threads:

```
(0: !!!Hello)
(0: world!!)
```

RESPUESTA:

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

RESPUESTA: atcgrid1. Obtenido de la var \$SLURM_JOB_NODELIST"

NOTA: Utilizar siempre con sbatch las opciones -n1 y -c, --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 -c y, para usar cores físicos y no lógicos, no olvide incluir --hint=nomultithread. Recordar que los srun dentro de un *script* heredan las opciones incluidas 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:

```
CarmenGarciaRomero c1estudiante11@carmen:~/Escritorio/Facultad/2/2Cuatri/AC/bp0/
ejer5 2021-03-13 sábado
$ ./SumaVectores 10
Tiempo:0.000000381 / Tamaño Vectores:10 / V1[0]+V2[0]=V3[0](0.164323+0.0
10048=0.174371) / / V1[9]+V2[9]=V3[9](1.857823+2.228462=4.086285) /
CarmenGarciaRomero c1estudiante11@carmen:~/Escritorio/Facultad/2/2Cuatri/AC/bp0/
ejer5 2021-03-13 sábado
$ ./SumaVectores 9
Tiempo:0.000000329 / Tamaño Vectores:9
/ V1[0]+V2[0]=V3[0](1.752911+0.535166=2.288078) /
/ V1[1]+V2[1]=V3[1](0.736486+0.027189=0.763675) /
/ V1[2]+V2[2]=V3[2](2.157840+0.925468=3.083308) /
/ V1[3]+V2[3]=V3[3](5.715129+2.973866=8.688996) /
/ V1[4]+V2[4]=V3[4](0.520301+1.019160=1.539461) /
/ V1[5]+V2[5]=V3[5](0.872487+1.895085=2.767572) /
/ V1[6]+V2[6]=V3[6](4.000191+2.394006=6.394197) /
/ V1[7]+V2[7]=V3[7](1.931430+2.121713=4.053143) /
/ V1[8]+V2[8]=V3[8](0.851384+1.301002=2.152386) /
```

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: El tiempo que tardan en sumarse las dos variables, en este caso, los dos vectores.

(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: En una estructura `struct timespec`, que contiene dos variables:

- `tv_sec`, tipo `time_t`, que almacena segundos
- `tv_nsec`, tipo `long`, que almacena 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: La información que devuelve la función `clock_gettime()` es el tiempo real del sistema, la var `tv_sec` es una cuenta en segundos que comenzó el 01/01/1970 00:00:00 UTC y los nanosegundos son el número de nanosegundos expirado en el segundo actual, este valor aumenta en un múltiplo según el reloj del sistema.

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 (se pueden obtener errores en tiempo de ejecución o de compilación, ver ejercicio 9). Obtener estos resultados usando *scripts* (partir del *script* que hay en el seminario). Debe haber una tabla para un nodo de cómputo de atcgrid con procesador Intel Xeon E5645 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 . Tiempos de ejecución de la suma de vectores locales, globales y dinámicos en mi pc.

Mi PC

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000058073	0.000313233	0.000137506
131072	1048576	0.000115132	0.000298348	0.000285850
262144	2097152	0.000390485	0.000611129	0.000607802
524288	4194304		0.001492112	0.001132235
1048576	8388608		0.002650048	0.002501648
2097152	16777216		0.004643862	0.004648320
4194304	33554432		0.009226040	0.009315333
8388608	67108864		0.018134335	0.017959146
16777216	134217728		0.035953521	0.036176094
33554432	268435456		0.073295680	0.074056202
67108864	536870912		0.072105322	0.150019526

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

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000189611	0.000543677	0.000493886
131072	1048576	0.000361425	0.000609016	0.000944515
262144	2097152	0.000731110	0.001427062	0.001882792
524288	4194304		0.002401807	0.002425583
1048576	8388608		0.004672857	0.004774212
2097152	16777216		0.008479454	0.008695735
4194304	33554432		0.016196420	0.016663615
8388608	67108864		0.031791046	0.032471612
16777216	134217728		0.062739695	0.064589576
33554432	268435456		0.129788781	0.128504904
67108864	536870912		0.124319691	0.246983846

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: No hay diferencias considerables en los tiempos. A tener en cuenta que para los vectores locales se produce una violación de segmento a partir del tercer valor.

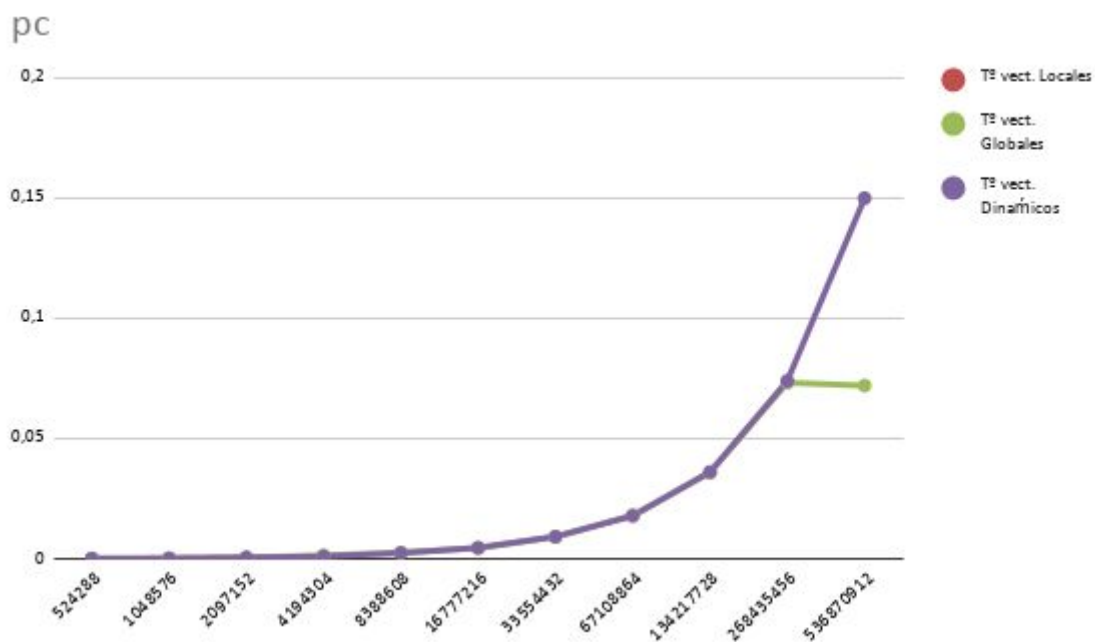


Imagen :Gráfico. Tiempos de ejecución en mi pc

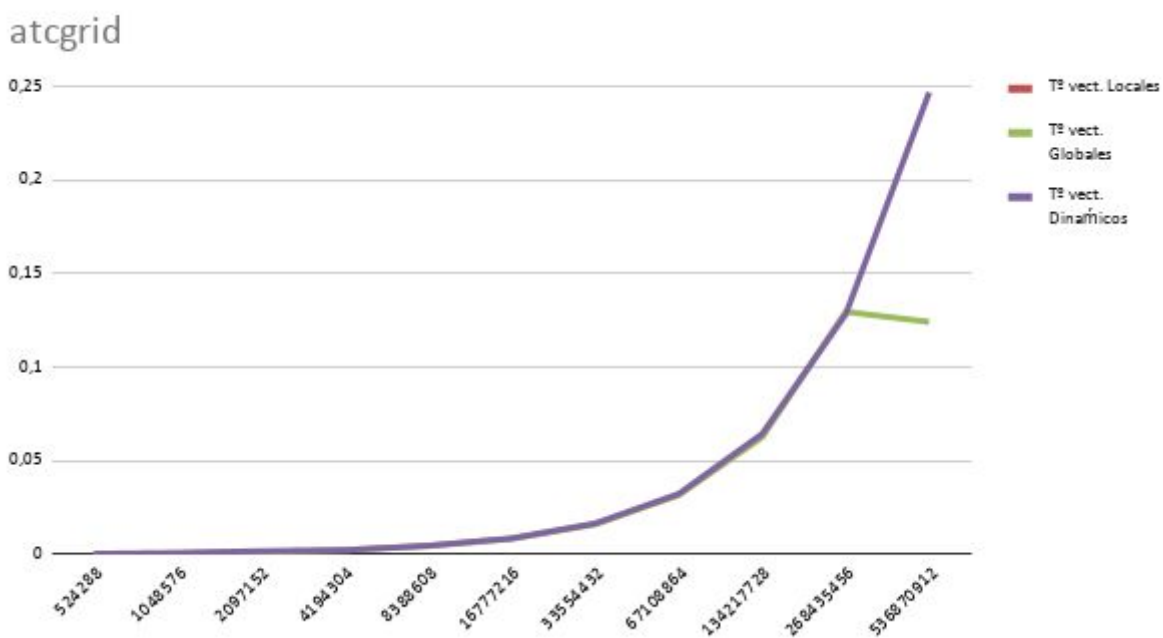


Imagen :Gráfico. Tiempos de ejecución en atcgrid

2. Contestar a las siguientes preguntas:

(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: Si, para los tamaños superiores a 262144. Porque se desborda la pila.

pc

```
CarmenGarciaRomero c1estudiante11@carmen:~/Escritorio/Facultad/2/2Cuatri/AC/bp0/ejer5 2021-03-13 sábado
$ ./SumaVectores.sh
Tiempo:0.000058073 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](5.506495+0.898610=6.405105)
/ / V1[65535]+V2[65535]=V3[65535](5.502388+0.185306=5.687694) /
Tiempo:0.000115132 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](5.506495+0.898610=6.405105)
/ / V1[131071]+V2[131071]=V3[131071](0.401463+0.865296=1.266759) /
Tiempo:0.000390485 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](5.506495+0.898610=6.405105)
/ / V1[262143]+V2[262143]=V3[262143](3.136172+7.482356=10.618528) /
./SumaVectores.sh: línea 15: 9896 Violación de segmento ('core' generado) ./SumaVectores $N
./SumaVectores.sh: línea 15: 9898 Violación de segmento ('core' generado) ./SumaVectores $N
./SumaVectores.sh: línea 15: 9900 Violación de segmento ('core' generado) ./SumaVectores $N
./SumaVectores.sh: línea 15: 9902 Violación de segmento ('core' generado) ./SumaVectores $N
./SumaVectores.sh: línea 15: 9904 Violación de segmento ('core' generado) ./SumaVectores $N
./SumaVectores.sh: línea 15: 9906 Violación de segmento ('core' generado) ./SumaVectores $N
./SumaVectores.sh: línea 15: 9908 Violación de segmento ('core' generado) ./SumaVectores $N
./SumaVectores.sh: línea 15: 9910 Violación de segmento ('core' generado) ./SumaVectores $N
```

Imagen : Resultado ejecución vectores locales en pc.

```
CarmenGarciaRomero c1estudiante11@atcgrid:~/bp0/ejer5 2021-03-14 domingo
$ sbatch -pac -n1 -c12 --hint=nomultithread SumaVectores.sh
Submitted batch job 69201
CarmenGarciaRomero c1estudiante11@atcgrid:~/bp0/ejer5 2021-03-14 domingo
$ cat slurm-69201.out
Id. usuario del trabajo: c1estudiante11
Id. del trabajo: 69201
Nombre del trabajo especificado por usuario: SumaVectores
Directorio de trabajo (en el que se ejecuta el script): /home/c1estudiante11/bp0/ejer5
Cola: ac
Nodo que ejecuta este trabajo: atcgrid.ugr.es
No de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgrid1
Tiempo:0.000189611 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](1.282014+0.214917=1.496931) / / V1[65535]+V2[65535]=V3[65535](0.249813+0.238959=0.488772) /
Tiempo:0.000361425 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](1.282014+0.214917=1.496931) / / V1[131071]+V2[131071]=V3[131071](0.879570+0.314163=1.193732) /
Tiempo:0.000731110 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](1.282014+0.214917=1.496931) / / V1[262143]+V2[262143]=V3[262143](0.468985+7.810068=8.279053) /
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)
```

Imagen : Resultado ejecución vectores locales en atcgrid.

(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: No da error.

```
CarmenGarciaRomero ciestudiante11@carmen:~/Escritorio/Facultad/2/2Cuatri/AC/bp0/ejer5 2021-03-13 sábado
$./SumaVectores.sh
Tiempo:0.000313233 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](0.362891+13.460941=13.823831)
/ / V1[65535]+V2[65535]=V3[65535](2.518793+3.709601=6.228394) /
Tiempo:0.000298348 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](0.362891+13.460941=13.823831)
/ / V1[131071]+V2[131071]=V3[131071](0.256196+0.324071=0.580267) /
Tiempo:0.000611129 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](0.362891+13.460941=13.823831)
/ / V1[262143]+V2[262143]=V3[262143](0.723082+1.272765=1.995847) /
Tiempo:0.001492112 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](0.362891+13.460941=13.823831)
/ / V1[524287]+V2[524287]=V3[524287](0.300455+2.166047=2.466502) /
Tiempo:0.002650048 / Tamaño Vectores:1048576 / V1[0]+V2[0]=V3[0](0.362891+13.460941=13.823831)
/ / V1[1048575]+V2[1048575]=V3[1048575](2.468171+0.272888=2.741060) /
Tiempo:0.004643862 / Tamaño Vectores:2097152 / V1[0]+V2[0]=V3[0](0.362891+13.460941=13.823831)
/ / V1[2097151]+V2[2097151]=V3[2097151](1.803097+5.538167=7.341263) /
Tiempo:0.009226040 / Tamaño Vectores:4194304 / V1[0]+V2[0]=V3[0](0.362891+13.460941=13.823831)
/ / V1[4194303]+V2[4194303]=V3[4194303](2.695750+1.277381=3.973131) /
Tiempo:0.018134335 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](0.362891+13.460941=13.823831)
/ / V1[8388607]+V2[8388607]=V3[8388607](2.299142+0.343381=2.642524) /
Tiempo:0.035953521 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](0.362891+13.460941=13.823831)
/ / V1[16777215]+V2[16777215]=V3[16777215](6.622490+0.222315=6.844806) /
Tiempo:0.073295680 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](9.570041+0.878014=10.448055)
/ / V1[33554431]+V2[33554431]=V3[33554431](1.775898+0.463882=2.239779) /
Tiempo:0.072105322 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](1.012992+0.817681=1.830673) /
/ / V1[33554431]+V2[33554431]=V3[33554431](0.216253+13.136197=13.352450) /
```

Imagen : Resultado ejecución vectores globales en pc.

```
CarmenGarciaRomero ciestudiante11@atcgrid:~/bp0/ejer5 2021-03-14 domingo
$ sbatch -pac -n1 -c12 --hint=nomultithread SumaVectores.sh
Submitted batch job 69214
CarmenGarciaRomero ciestudiante11@atcgrid:~/bp0/ejer5 2021-03-14 domingo
$ cat slurm-69214.out
Id. usuario del trabajo: ciestudiante11
Id. del trabajo: 69214
Nombre del trabajo especificado por usuario: SumaVectores
Directorio de trabajo (en el que se ejecuta el script): /home/ciestudiante11/bp0/ejer5
Cola: ac
Nodo que ejecuta este trabajo:atcgrid.ugr.es
No de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgrid2
Tiempo:0.000543677 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](1.101276+1.070239=
2.171515) / / V1[65535]+V2[65535]=V3[65535](0.847649+1.495151=2.342800) /
Tiempo:0.000609016 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](1.101276+1.070239=
2.171515) / / V1[131071]+V2[131071]=V3[131071](0.424019+0.448635=0.872654) /
Tiempo:0.001427062 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](1.101276+1.070239=
2.171515) / / V1[262143]+V2[262143]=V3[262143](0.198713+4.587871=4.786584) /
Tiempo:0.002401807 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](1.101276+1.070239=
2.171515) / / V1[524287]+V2[524287]=V3[524287](0.379923+1.028449=1.408372) /
Tiempo:0.004672857 / Tamaño Vectores:1048576 / V1[0]+V2[0]=V3[0](1.101276+1.070239=
2.171515) / / V1[1048575]+V2[1048575]=V3[1048575](0.088727+1.493118=1.581845) /
Tiempo:0.008479454 / Tamaño Vectores:2097152 / V1[0]+V2[0]=V3[0](1.101276+1.070239=
2.171515) / / V1[2097151]+V2[2097151]=V3[2097151](2.239340+0.691473=2.930813) /
Tiempo:0.016196420 / Tamaño Vectores:4194304 / V1[0]+V2[0]=V3[0](3.257265+0.430167=
3.687432) / / V1[4194303]+V2[4194303]=V3[4194303](1.532915+0.707898=2.240813) /
Tiempo:0.031791046 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](3.257265+0.430167=
3.687432) / / V1[8388607]+V2[8388607]=V3[8388607](20.876510+1.185724=22.062234) /
Tiempo:0.062739695 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](0.070164+0.259249=
0.329413) / / V1[16777215]+V2[16777215]=V3[16777215](1.375713+0.918534=2.294247) /
Tiempo:0.129788781 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](0.751130+1.840885=
2.592015) / / V1[33554431]+V2[33554431]=V3[33554431](0.937686+1.081361=2.019047) /
Tiempo:0.124319691 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](0.720533+0.146367=
0.866900) / / V1[33554431]+V2[33554431]=V3[33554431](1.073468+1.020398=2.093866) /
```

Imagen : Resultado ejecución vectores globales en atcgrid.

(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:

```
CarmenGarciaRomero ciestudiante11@carmen:~/Escritorio/Facultad/2/2Cuatri/AC/bp0/ejer5 2021-03-13 sábado
$ ./SumaVectores.sh
Tiempo:0.000137506 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](4.531068+0.735514=5.266582) /
/ V1[65535]+V2[65535]=V3[65535](2.723220+2.477151=5.200371) /
Tiempo:0.000285850 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](4.531068+0.735514=5.266582) /
/ V1[131071]+V2[131071]=V3[131071](0.569766+3.143557=3.713323) /
Tiempo:0.000607802 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](4.531068+0.735514=5.266582) /
/ V1[262143]+V2[262143]=V3[262143](1.529874+0.347578=1.877453) /
Tiempo:0.001132235 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](4.531068+0.735514=5.266582) /
/ V1[524287]+V2[524287]=V3[524287](1.588032+0.254271=1.842303) /
Tiempo:0.002501648 / Tamaño Vectores:1048576 / V1[0]+V2[0]=V3[0](4.531068+0.735514=5.266582) /
/ V1[1048575]+V2[1048575]=V3[1048575](0.549071+0.107046=0.656117) /
Tiempo:0.004648320 / Tamaño Vectores:2097152 / V1[0]+V2[0]=V3[0](1.957974+0.802255=2.760229) /
/ V1[2097151]+V2[2097151]=V3[2097151](0.600421+6.984689=7.585109) /
Tiempo:0.009315333 / Tamaño Vectores:4194304 / V1[0]+V2[0]=V3[0](1.957974+0.802255=2.760229) /
/ V1[4194303]+V2[4194303]=V3[4194303](1.423633+0.095780=1.519413) /
Tiempo:0.017959146 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](1.957974+0.802255=2.760229) /
/ V1[8388607]+V2[8388607]=V3[8388607](5.231822+0.777643=6.009466) /
Tiempo:0.036176094 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](1.957974+0.802255=2.760229) /
/ V1[16777215]+V2[16777215]=V3[16777215](0.837835+1.073042=1.910877) /
Tiempo:0.074056202 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](1.957974+0.802255=2.760229) /
/ V1[33554431]+V2[33554431]=V3[33554431](1.049425+1.355984=2.405409) /
Tiempo:0.150019526 / Tamaño Vectores:67108864 / V1[0]+V2[0]=V3[0](0.083175+1.960053=2.043228) /
/ V1[67108863]+V2[67108863]=V3[67108863](0.557772+0.217997=0.775769) /
```

Imagen : Resultado ejecución vectores dinámicos en pc.

```
CarmenGarciaRomero ciestudiante11@atcgrid:~/bp0/ejer5 2021-03-14 domingo
$ sbatch -pac -n1 -c12 --hint=nomultithread SumaVectores.sh
Submitted batch job 69233
CarmenGarciaRomero ciestudiante11@atcgrid:~/bp0/ejer5 2021-03-14 domingo
$ sbatcat slurm-69233.out
Id. usuario del trabajo: ciestudiante11
Id. del trabajo: 69233
Nombre del trabajo especificado por usuario: SumaVectores
Directorio de trabajo (en el que se ejecuta el script): /home/ciestudiante11/bp0/ejer5
Cola: ac
Nodo que ejecuta este trabajo:atcgrid.ugr.es
No de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgrid1
Tiempo:0.000493886 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](0.326883+9.530767=9.857649) /
/ V1[65535]+V2[65535]=V3[65535](1.064454+0.594695=1.659149) /
Tiempo:0.000944515 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](0.326883+9.530767=9.857649) /
/ V1[131071]+V2[131071]=V3[131071](0.632125+0.097604=0.729729) /
Tiempo:0.001882792 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](0.326883+9.530767=9.857649) /
/ V1[262143]+V2[262143]=V3[262143](0.025047+1.384847=1.409893) /
Tiempo:0.002425583 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](0.180567+2.417029=2.597597) /
/ V1[524287]+V2[524287]=V3[524287](5.346417+0.207680=5.554097) /
Tiempo:0.004774212 / Tamaño Vectores:1048576 / V1[0]+V2[0]=V3[0](0.180567+2.417029=2.597597) /
/ V1[1048575]+V2[1048575]=V3[1048575](1.070849+0.453975=1.524824) /
Tiempo:0.008695735 / Tamaño Vectores:2097152 / V1[0]+V2[0]=V3[0](0.180567+2.417029=2.597597) /
/ V1[2097151]+V2[2097151]=V3[2097151](0.784895+1.503216=2.288112) /
Tiempo:0.016663615 / Tamaño Vectores:4194304 / V1[0]+V2[0]=V3[0](0.180567+2.417029=2.597597) /
/ V1[4194303]+V2[4194303]=V3[4194303](99.748269+0.016818=99.765087) /
Tiempo:0.032471612 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](0.006209+1.122807=1.129016) /
/ V1[8388607]+V2[8388607]=V3[8388607](1.058593+1.551457=2.610050) /
Tiempo:0.064589576 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](0.006209+1.122807=1.129016) /
/ V1[16777215]+V2[16777215]=V3[16777215](0.525846+11.833257=12.359102) /
Tiempo:0.128504904 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](0.592900+0.069233=0.662133) /
/ V1[33554431]+V2[33554431]=V3[33554431](1.257659+0.206830=1.464489) /
Tiempo:0.246983846 / Tamaño Vectores:67108864 / V1[0]+V2[0]=V3[0](0.080509+0.209721=0.290230) /
/ V1[67108863]+V2[67108863]=V3[67108863](0.715157+5.567826=6.282983) /
```

Imagen : Resultado ejecución vectores dinámicos en atcgrid.

3. (a) ¿Cuál es el máximo valor que se puede almacenar en la variable N teniendo en cuenta su tipo? Razonar respuesta.

RESPUESTA: $2^{32}-1$ (0xffffffff), porque es el número máximo que se puede representar en un dato de tipo entero de 32 bits (se le resta 1 porque hay que considerar también el 0).

(b) Modificar el código fuente C (en el PC) para que el límite de los vectores cuando se declaren 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 el siguiente error

```
CarmenGarciaRomero c1estudiante11@carmen:~/Escritorio/Facultad/2/2Cuatri/AC/bp0/ejer5 2021-
03-14 domingo
$gcc -O2 SumaVectores.c -o SumaVectores -lrt
/tmp/ccFGgWk8.o: en la función `main':
SumaVectores.c:(.text.startup+0x68): reubicación truncada para ajustar: R_X86_64_PC32 contra
el símbolo `v2' definido en la sección COMMON en /tmp/ccFGgWk8.o
SumaVectores.c:(.text.startup+0xbb): reubicación truncada para ajustar: R_X86_64_PC32 contra
el símbolo `v3' definido en la sección COMMON en /tmp/ccFGgWk8.o
SumaVectores.c:(.text.startup+0x205): reubicación truncada para ajustar: R_X86_64_PC32 contra
el símbolo `v2' definido en la sección COMMON en /tmp/ccFGgWk8.o
collect2: error: ld returned 1 exit status
```

Esto ocurre porque hemos sobrepasado el tamaño permitido y el compilador detecta el desbordamiento.

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. Se generan aleatoriamente las componentes para vectores de tamaño mayor que 8 y se imprimen todas las componentes para vectores menores que 10.

```
/* 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(), rand(), srand(), 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
    if (N < 9)
        for (i = 0; i < N; i++)
        {
            v1[i] = N * 0.1 + i * 0.1;
            v2[i] = N * 0.1 - i * 0.1;
        }
    else
    {
        srand(time(0));
        for (i = 0; i < N; i++)
        {
            v1[i] = rand() / ((double) rand());
            v2[i] = rand() / ((double) rand()); // printf("%d:%f,%f", i, v1[i], v2[i]);
        }
    }

    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.):%11.9ft / 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,v1[i],v2[i],v3[i]);
}
else
printf("Tiempo(seg.):%11.9ft / 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;
}

```