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 Alconchel Vázquez
Grupo de prácticas y profesor de prácticas: 1,
Fecha de entrega:
Fecha evaluación en clase:

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 (-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 `ejer1`)

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

RESPUESTA:

```
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0] 2021-03-10
miércoles
$ 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) i7-8750H CPU @ 2.20GHz
Revisión:                            10
CPU MHz:                             900.048
CPU MHz máx.:                        4100,0000
CPU MHz mín.:                        800,0000
BogoMIPS:                            4399.99
Virtualización:                      VT-x
Caché L1d:                           192 KiB
Caché L1i:                           192 KiB
Caché L2:                            1,5 MiB
Caché L3:                            9 MiB
CPU(s) del nodo NUMA 0:              0-11
Vulnerability Itlb multihit:         KVM: Mitigation: VMX disabled
Vulnerability L1tf:                  Mitigation; PTE Inversion; VMX conditional cache flushes, SMT vulnerable
Vulnerability Mds:                   Mitigation; Clear CPU buffers; SMT vulnerable
Vulnerability Meltdown:              Mitigation; PTI
```

```
[DanielAlconchelVázquez e1estudiante1@atcgrid:~] 2021-03-10 miércoles
$ srun -p ac4 -A ac 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:               1079.223
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 no
```

```
[DanielAlconchelVázquez e1estudiante1@atcgrid:~] 2021-03-10 miércoles
$ srun -p ac -A 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 x
```

(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:

Los **cores físicos** podemos consultarlos mirando el **número del Socket(s)**. Con un rápido vistazo, observamos que atcgrid [1-4] tienen 2 cores físicos por 16 cores por socket dan lugar a 32 cores físicos, mientras que los **cores lógicos** se observan en la **CPU(s),** luego atdgrid[1-3] tienen 24 cores lógicos y atcgrid4 tiene 64.

Por último, mi PC tiene 12 cores lógicos y 6 físico.

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:

```
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0] 2021-03-10
miércoles
$ gcc -O2 -fopenmp -o HelloOMP HelloOMP.c
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0] 2021-03-10
miércoles
$ ./HelloOMP
(8:!!!Hello world!!!)
(6:!!!Hello world!!!)
(2:!!!Hello world!!!)
(3:!!!Hello world!!!)
(10:!!!Hello world!!!)
(7:!!!Hello world!!!)
(5:!!!Hello world!!!)
(1:!!!Hello world!!!)
(4:!!!Hello world!!!)
(9:!!!Hello world!!!)
(11:!!!Hello world!!!)
(0:!!!Hello world!!!)
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0] 2021-03-10
miércoles
$
```

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

RESPUESTA: Imprime la cadena de caracteres 12 veces, ya que dispongo de 12 cores lógicos en mi PC.

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*):

```
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0] 2021-03-10
miércoles
$ sftp  e1estudiante1@atcgrid.ugr.es
e1estudiante1@atcgrid.ugr.es's password:
Connected to atcgrid.ugr.es.
sftp> put
HelloOMP    HelloOMP.c
sftp> put H
HelloOMP    HelloOMP.c
sftp> put HelloOMP.c
Uploading HelloOMP.c to /home/e1estudiante1/HelloOMP.c
HelloOMP.c                                    100%  214     3.5KB/s   00:00
sftp>
```

(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:

```
$ ls
HelloOMP  HelloOMP.c
[DanielAlconchelVázquez e1estudiante1@atcgrid:~/bp0/ejer2] 2021-03-10 miércoles
$ srun -pac -Aac -n1 -c12  --hint=nomultithread HelloOMP
(11:!!!Hello world!!!)
(4:!!!Hello world!!!)
(7:!!!Hello world!!!)
(5:!!!Hello world!!!)
(2:!!!Hello world!!!)
(8:!!!Hello world!!!)
(10:!!!Hello world!!!)
(1:!!!Hello world!!!)
(6:!!!Hello world!!!)
(3:!!!Hello world!!!)
(0:!!!Hello world!!!)
(9:!!!Hello world!!!)
[DanielAlconchelVázquez e1estudiante1@atcgrid:~/bp0/ejer2] 2021-03-10 miércoles
$ srun --partition=ac --account=ac --ntasks=1 --cpus-per-task=12 --hint=nomultithread HelloOMP
(7:!!!Hello world!!!)
(3:!!!Hello world!!!)
(9:!!!Hello world!!!)
(4:!!!Hello world!!!)
(6:!!!Hello world!!!)
(8:!!!Hello world!!!)
(1:!!!Hello world!!!)
(11:!!!Hello world!!!)
(5:!!!Hello world!!!)
(10:!!!Hello world!!!)
(0:!!!Hello world!!!)
(2:!!!Hello world!!!)
[DanielAlconchelVázquez e1estudiante1@atcgrid:~/bp0/ejer2] 2021-03-10 miércoles
```

(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:

```
[DanielAlconchelVázquez e1estudiante1@atcgrid:~/bp0/ejer2] 2021-03-10 miércoles
$ srun -pac -Aac -n1 -c24 HelloOMP
(14:!!!Hello world!!!)
(6:!!!Hello world!!!)
(3:!!!Hello world!!!)
(5:!!!Hello world!!!)
(19:!!!Hello world!!!)
(4:!!!Hello world!!!)
(23:!!!Hello world!!!)
(12:!!!Hello world!!!)
(21:!!!Hello world!!!)
(20:!!!Hello world!!!)
(1:!!!Hello world!!!)
(22:!!!Hello world!!!)
(9:!!!Hello world!!!)
(7:!!!Hello world!!!)
(15:!!!Hello world!!!)
(2:!!!Hello world!!!)
(13:!!!Hello world!!!)
(10:!!!Hello world!!!)
(8:!!!Hello world!!!)
(17:!!!Hello world!!!)
(11:!!!Hello world!!!)
(16:!!!Hello world!!!)
(18:!!!Hello world!!!)
(0:!!!Hello world!!!)
[DanielAlconchelVázquez e1estudiante1@atcgrid:~/bp0/ejer2] 2021-03-10 miércoles
$
```

(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:

```
[DanielAlconchelVázquez e1estudiante1@atcgrid:~/bp0/ejer2] 2021-03-10 miércoles
$ srun -n1 HelloOMP
(1:!!!Hello world!!!)
(0:!!!Hello world!!!)
[DanielAlconchelVázquez e1estudiante1@atcgrid:~/bp0/ejer2] 2021-03-10 miércoles
$
```

Se está usando la cola por defecto, ac.

(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 --partition=ac4 --account=ac --ntasks=1 --cpus-per-task=32`

`--hint=nomultithread HelloOMP`

- **`cpus-per-task`** `indica el número de cores físicos a usar.`
- **`partition`** `indica la cola, que en nuestro caso es atcgrid4.`
- **`hint=nomultithread`** `indica que no queremos usar cores lógicos.`

```
[DanielAlconchelVázquez e1estudiante1@atcgrid:~/bp0/ejer2] 2021-03-10 miércoles
$ srun --partition=ac4 --account=ac --ntasks=1 --cpus-per-task=32 --hint=nomultithread HelloOMP
(11:!!!Hello world!!!)
(3:!!!Hello world!!!)
(24:!!!Hello world!!!)
(1:!!!Hello world!!!)
(10:!!!Hello world!!!)
(15:!!!Hello world!!!)
(26:!!!Hello world!!!)
(27:!!!Hello world!!!)
(8:!!!Hello world!!!)
(22:!!!Hello world!!!)
(19:!!!Hello world!!!)
(0:!!!Hello world!!!)
(16:!!!Hello world!!!)
(20:!!!Hello world!!!)
(30:!!!Hello world!!!)
(12:!!!Hello world!!!)
(17:!!!Hello world!!!)
(21:!!!Hello world!!!)
(2:!!!Hello world!!!)
(6:!!!Hello world!!!)
(23:!!!Hello world!!!)
(9:!!!Hello world!!!)
(28:!!!Hello world!!!)
(18:!!!Hello world!!!)
(25:!!!Hello world!!!)
(4:!!!Hello world!!!)
(31:!!!Hello world!!!)
(5:!!!Hello world!!!)
(13:!!!Hello world!!!)
(7:!!!Hello world!!!)
(29:!!!Hello world!!!)
(14:!!!Hello world!!!)
[DanielAlconchelVázquez e1estudiante1@atcgrid:~/bp0/ejer2] 2021-03-10 miércoles
$
```

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`).

```c
/* Compilar con:
 * gcc -O2 -fopenmp -o HelloOMP HelloOMP.c
 */
#include <stdio.h>
#include <omp.h>

int main(void){
        #pragma omp parallel
        printf("(%d:!!!Hello...)\n",
                omp_get_thread_num());
        printf("(%d:...World!)\n",
                omp_get_thread_num());
        return(0);
}
```

```
NORMAL  HelloOMP.c                                  c  Σ 7% ☰1/14 ㏑:1
```

```
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestr
e/AC/Prácticas/bp0/ejer4] 2021-03-11 jueves
$ gcc -O2 -fopenmp -o HelloOMP2 HelloOMP2.c
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestr
e/AC/Prácticas/bp0/ejer4] 2021-03-11 jueves
$ ./HelloOMP2
(10:!!!Hello...)
(3:!!!Hello...)
(11:!!!Hello...)
(9:!!!Hello...)
(6:!!!Hello...)
(7:!!!Hello...)
(2:!!!Hello...)
(5:!!!Hello...)
(4:!!!Hello...)
(0:!!!Hello...)
(1:!!!Hello...)
(8:!!!Hello...)
(0:...World!)
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestr
e/AC/Prácticas/bp0/ejer4] 2021-03-11 jueves
$
```

```
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestr
e/AC/Prácticas/bp0/ejer4] 2021-03-11 jueves
$ sftp e1estudiante1@atcgrid.ugr.es
e1estudiante1@atcgrid.ugr.es's password:
Connected to atcgrid.ugr.es.
sftp> lls
HelloOMP2  HelloOMP2.c
sftp> put He
HelloOMP2    HelloOMP2.c
sftp> put HelloOMP2
Uploading HelloOMP2 to /home/e1estudiante1/HelloOMP2
HelloOMP2                                     100%   17KB  66.7KB/s   00:00
sftp>
```

```
[DanielAlconchelVázquez e1estudiante1@atcgrid:~/bp0/ejer4] 2021-03-11 jueves
$ ls
HelloOMP2  script_helloomp.sh
[DanielAlconchelVázquez e1estudiante1@atcgrid:~/bp0/ejer4] 2021-03-11 jueves
$ sbatch --job-name=helloMP2 --partition=ac --account=ac


^C
[DanielAlconchelVázquez e1estudiante1@atcgrid:~/bp0/ejer4] 2021-03-11 jueves
$ sbatch -pac -Aac ./script_helloomp.sh
Submitted batch job 65654
[DanielAlconchelVázquez e1estudiante1@atcgrid:~/bp0/ejer4] 2021-03-11 jueves
$ ls
HelloOMP2  script_helloomp.sh  slurm-65654.out
[DanielAlconchelVázquez e1estudiante1@atcgrid:~/bp0/ejer4] 2021-03-11 jueves
$ cat slurm-65654.out
Id. usuario del trabajo: e1estudiante1
Id. del trabajo: 65654
Nombre del trabajo especificado por usuario: helloOMP2
Directorio de trabajo (en el que se ejecuta el script): /home/e1estudiante1/bp0/ejer4
Cola: ac
Nodo que ejecuta este trabajo: atcgrid.ugr.es
Nº de nodos asignadosal trabajo: 1
Nodos asignados al trabajo: atcgrid1
CPUs por nodo: 2

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

(0:!!!Hello...)
(1:!!!Hello...)
(0:...World!)

 2. Ejecución helloOMPvarias veces con distinto nº de threads:
```

```
 2. Ejecución helloOMPvarias veces con distinto nº de threads:


  - Para :
(1:!!!Hello...)
(2:!!!Hello...)
(0:!!!Hello...)
(3:!!!Hello...)
(4:!!!Hello...)
(5:!!!Hello...)
(6:!!!Hello...)
(7:!!!Hello...)
(8:!!!Hello...)
(9:!!!Hello...)
(10:!!!Hello...)
(11:!!!Hello...)
(0:...World!)

  - Para :
(2:!!!Hello...)
(1:!!!Hello...)
(3:!!!Hello...)
(4:!!!Hello...)
(0:!!!Hello...)
(5:!!!Hello...)
(0:...World!)

  - Para :
(2:!!!Hello...)
(1:!!!Hello...)
(0:!!!Hello...)
(0:...World!)

  - Para :
(0:!!!Hello...)
(0:...World!)
[DanielAlconchelVázquez e1estudiante1@atcgrid:~/bp0/ejer4] 2021-03-11 jueves
$
```

(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.

RESPUESTA:

```
[DanielAlconchelVázquez e1estudiante1@atcgrid:~/bp0/ejer4] 2021-03-11 jueves
$ cat script_helloomp.sh
#!/bin/bash
#Órdenes para el Gestor de carga de trabajo:
#1. Asigna al trabajo un nombre
#SBATCH --job-name=helloOMP2
#2. Asignar el trabajo a una partición (cola)
#SBATCH --partition=ac
#2. Asignar el trabajo a un account
#SBATCH --account=ac
#Obtener información de las variables del entorno del Gestor de carga de trabajo:
echo "Id. usuario del trabajo: $SLURM_JOB_USER"
echo "Id. del trabajo: $SLURM_JOBID"
echo "Nombre del trabajo especificado por usuario: $SLURM_JOB_NAME"
echo "Directorio de trabajo (en el que se ejecuta el script): $SLURM_SUBMIT_DIR"
echo "Cola: $SLURM_JOB_PARTITION"
echo "Nodo que ejecuta este trabajo: $SLURM_SUBMIT_HOST"
echo "Nº de nodos asignadosal trabajo: $SLURM_JOB_NUM_NODES"
echo "Nodos asignados al trabajo: $SLURM_JOB_NODELIST"
echo "CPUs por nodo: $SLURM_JOB_CPUS_PER_NODE"
#Instrucciones del script para ejecutar código:
echo -e "\n 1. Ejecución helloOMP una vez sin cambiar nº de threads(valor por defecto):\n"
srun ./HelloOMP2
echo -e "\n 2. Ejecución helloOMPvarias veces con distinto nº de threads:\n"
for ((P=12;P>0;P=P/2))
                do
                                export OMP_NUM_THREADS=$P
                                echo -e "\n  - Para $Pthreads:"
                                srun ./HelloOMP2
                done
```

```
[DanielAlconchelVázquez e1estudiante1@atcgrid:~/bp0/ejer4] 2021-03-11 jueves
$ cat slurm-65659.out
Id. usuario del trabajo: e1estudiante1
Id. del trabajo: 65659
Nombre del trabajo especificado por usuario: helloOMP2
Directorio de trabajo (en el que se ejecuta el script): /home/e1estudiante1/bp0/ejer4
Cola: ac
Nodo que ejecuta este trabajo: atcgrid.ugr.es
Nº de nodos asignadosal trabajo: 1
Nodos asignados al trabajo: atcgrid1
CPUs por nodo: 24

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

(0:!!!Hello...)
(1:!!!Hello...)
(5:!!!Hello...)
(7:!!!Hello...)
(8:!!!Hello...)
(3:!!!Hello...)
(11:!!!Hello...)
(6:!!!Hello...)
(9:!!!Hello...)
(10:!!!Hello...)
(2:!!!Hello...)
(4:!!!Hello...)
(0:...World!)
 2. Ejecución helloOMPvarias veces con distinto nº de threads:


  - Para :
(4:!!!Hello...)
(0:!!!Hello...)
(9:!!!Hello...)
(8:!!!Hello...)
(11:!!!Hello...)
(5:!!!Hello...)
(3:!!!Hello...)
(10:!!!Hello...)
(7:!!!Hello...)
(2:!!!Hello...)
(6:!!!Hello...)
(1:!!!Hello...)
(0:...World!)

  - Para :
(4:!!!Hello...)
(2:!!!Hello...)
(5:!!!Hello...)
(1:!!!Hello...)
(3:!!!Hello...)
(0:!!!Hello...)
(0:...World!)

  - Para :
(1:!!!Hello...)
(0:!!!Hello...)
(2:!!!Hello...)
(0:...World!)

  - Para :
(0:!!!Hello...)
(0:...World!)
[DanielAlconchelVázquez e1estudiante1@atcgrid:~/bp0/ejer4] 2021-03-11 jueves
$
```

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

RESPUESTA: Se ha ejecutado en atcgrid1 tal y como dice la script en el parámetro llamado nodos asignados al trabajo actual.

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:

```
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0/ejer5] 2021-03-11 jueves
$ ls
Listado.c
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0/ejer5] 2021-03-11 jueves
$ gcc -o listado -fopenmp -O2 Listado.c
Listado.c: In function 'main':
Listado.c:86:56: warning: format '%lu' expects argument of type 'long unsigned int', but argument 3 has type 'unsigned int' [-Wformat=]
   86 |     printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%lu\n", ncgt, N);
      |                                                      ~~^   ~
      |                                                        |   |
      |                                                        |   unsigned int
      |                                                        long unsigned int
      |                                                      %u
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0/ejer5] 2021-03-11 jueves
$ ./listado 1000
Tiempo(seg.):0.000013655        / Tamaño Vectores:1000 / V1[0]+V2[0]=V3[0](2.922457+0.218092=3.140549) / /          V1[999]+V2[999]=V3[999](7.721429+2.759517=10.480946) /
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0/ejer5] 2021-03-11 jueves
$
```

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: Contiene la diferencia de tiempo entre que empieza el cálculo de la suma y acaba el mismo.

(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**: clock_gettime() es una función que devuelve el tiempo del reloj especificado por la variable clock_id. Lo almacena en un struct llamado timespec, que contiene dos variables, una llamada tv_sec, que es una variable tipo time_t, que almacena los segundos, y otra llamada tv_nsec, que es de tipo long y almacena 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: Devuelve la información del reloj pasado en el primer parámetro y la almacena en el struct pasado como segundo parámetro. En nuestro caso, le pasamos CLOCK_REALTIME, que es 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 de la ejecución en mi PC:

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

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000227136	0.000744213	0.000739783
131072	1048576	0.000387073	0.000836189	0.001180513
262144	2097152	0.000436373	0.000987118	0.001135666
524288	4194304	seg fault	0.001829190	0.001827518
1048576	8388608	seg fault	0.003409529	0.003769192
2097152	16777216	seg fault	0.006861596	0.007795086
4194304	33554432	seg fault	0.014702092	0.013465917
8388608	67108864	seg fault	0.031986452	0.029440378
16777216	134217728	seg fault	0.064244778	0.053612924
33554432	268435456	seg fault	0.117918090	0.111606577
67108864	536870912	seg fault	0.113628260	0.221783694

Tabla de la ejecución en mi atcgrid:

Tabla 2 . Copiar la tabla de la hoja de cálculo utilizada

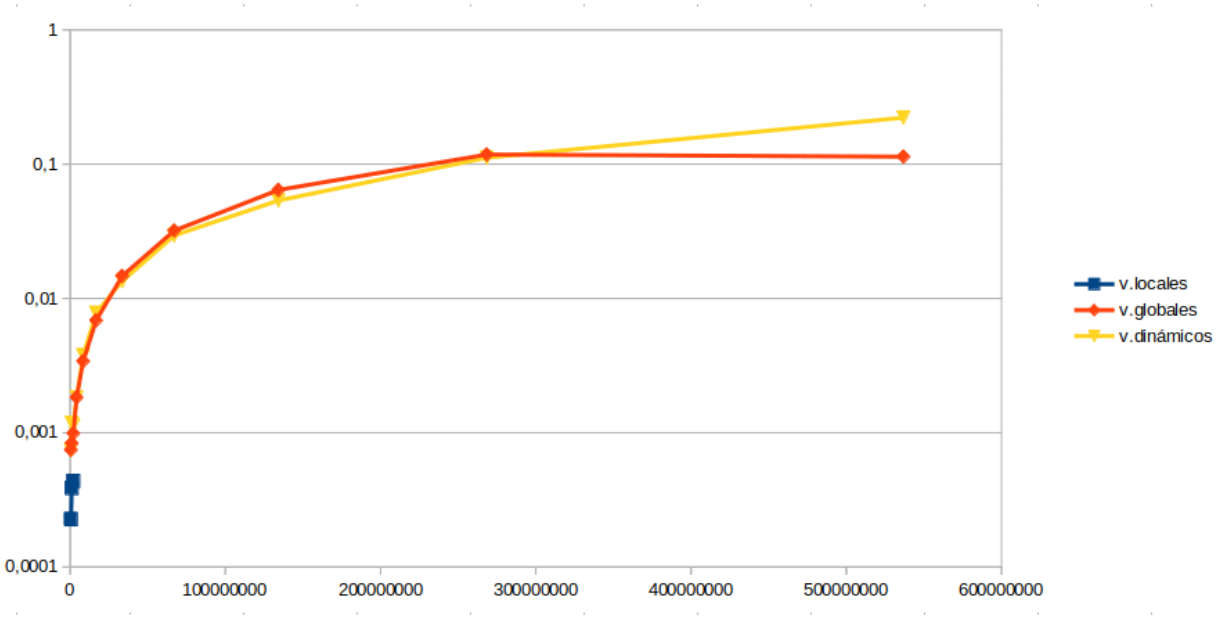
Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000469221	0.000543802	0.000460086
131072	1048576	0.000952945	0.000496844	0.000957977
262144	2097152	0.001556934	0.001447369	0.001759706
524288	4194304	seg fault	0.002463571	0.002540534
1048576	8388608	seg fault	0.004904558	0.004707515
2097152	16777216	seg fault	0.009107574	0.008698365
4194304	33554432	seg fault	0.017641888	0.016685829
8388608	67108864	seg fault	0.033818387	0.032267992
16777216	134217728	seg fault	0.068234203	0.064266470
33554432	268435456	seg fault	0.130884207	0.127241380
67108864	536870912	seg fault	0.131369701	0.246458482

He usado la orden srun -pac -n1 -c12 script.sh

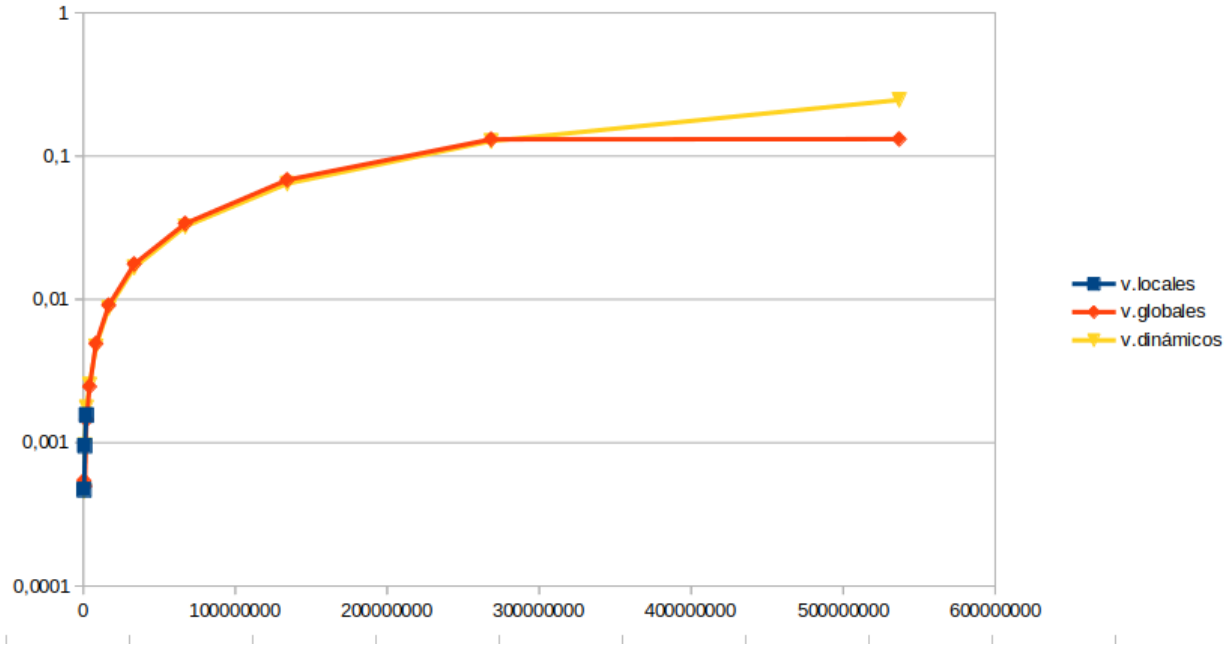
1. Con ayuda de la hoja de cálculo representar **en una misma gráfica** los tiempos de ejecución obtenidos en atcgrid y en su PC para vectores locales, globales y dinámicos (eje y) en función del tamaño en bytes de un vector (por tanto, los valores de la segunda columna de la tabla, que están en escala 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:

- En mi PC:



- En arcgrid:



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:

```
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0/ejer7] 2021-03-11 jueves
$ gcc -o listado -fopenmp -O2 Listado.c
Listado.c: In function ‘main’:
Listado.c:86:56: warning: format ‘%lu’ expects argument of type ‘long unsigned int’, but argument 3 has type ‘unsigned int’ [-Wformat=]
   86 |     printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%lu\n", ncgt, N);
      |                                                      ~~^   ~
      |                                                        |   |
      |                                                        |   unsigned int
      |                                                        long unsigned int
      |                                                      %u
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0/ejer7] 2021-03-11 jueves
$ ./script.sh
Tiempo(seg.):0.000227136        / Tamaño Vectores:65536        / V1[0]+V2[0]=V3[0](0.751800+2.608216=3.360017) / /          V1[65535]+V2[65535]=V3[65535](0.458263+0.397012=0.855275) /
Tiempo(seg.):0.000387073        / Tamaño Vectores:131072       / V1[0]+V2[0]=V3[0](0.751800+2.608216=3.360017) / /          V1[131071]+V2[131071]=V3[131071](1.083926+0.363831=1.447757) /
Tiempo(seg.):0.000436373        / Tamaño Vectores:262144       / V1[0]+V2[0]=V3[0](0.751800+2.608216=3.360017) / /          V1[262143]+V2[262143]=V3[262143](4.943451+1.088628=6.032079) /
./script.sh: línea 4: 18985 Violación de segmento  (`core' generado) ./listado $i
./script.sh: línea 4: 18987 Violación de segmento  (`core' generado) ./listado $i
./script.sh: línea 4: 18989 Violación de segmento  (`core' generado) ./listado $i
./script.sh: línea 4: 18991 Violación de segmento  (`core' generado) ./listado $i
./script.sh: línea 4: 18994 Violación de segmento  (`core' generado) ./listado $i
./script.sh: línea 4: 18996 Violación de segmento  (`core' generado) ./listado $i
./script.sh: línea 4: 18998 Violación de segmento  (`core' generado) ./listado $i
./script.sh: línea 4: 19000 Violación de segmento  (`core' generado) ./listado $i
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0/ejer7] 2021-03-11 jueves
$
```

Los vectores locales se encuentran en el stack, que es un espacio de memoria reducido, luego no podemos reservar vectores cuyo espacio sobrepasen el del stack.

(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:

```
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0/ejer7] 2021-03-11 jueves
$ gcc -o listado -fopenmp -O2 Listado.c
Listado.c: In function ‘main’:
Listado.c:86:56: warning: format ‘%lu’ expects argument of type ‘long unsigned int’, but argument 3 has type ‘unsigned int’ [-Wformat=]
   86 |     printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%lu\n", ncgt, N);
      |                                                      ~~^   ~
      |                                                        |   |
      |                                                        |   unsigned int
      |                                                        long unsigned int
      |                                                      %u
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0/ejer7] 2021-03-11 jueves
$ cat script.sh
#!/bin/bash
sizes=(65536 131072 262144 524288 1048576 2097152 4194304 8388608 16777216 33554432 67108864)

for i in "${sizes[@]}"; do
        ./listado $i
done
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0/ejer7] 2021-03-11 jueves
$ ./script.sh
Tiempo(seg.):0.000744213        / Tamaño Vectores:65536        / V1[0]+V2[0]=V3[0](1.314008+0.038702=1.352710) / /          V1[65535]+V2[65535]=V3[65535](2.651726+0.545315=3.197041) /
Tiempo(seg.):0.000836189        / Tamaño Vectores:131072       / V1[0]+V2[0]=V3[0](1.314008+0.038702=1.352710) / /          V1[131071]+V2[131071]=V3[131071](14.579392+5.197725=19.777117) /
Tiempo(seg.):0.000987118        / Tamaño Vectores:262144       / V1[0]+V2[0]=V3[0](1.314008+0.038702=1.352710) / /          V1[262143]+V2[262143]=V3[262143](2.040764+0.790713=2.831477) /
Tiempo(seg.):0.001829190        / Tamaño Vectores:524288       / V1[0]+V2[0]=V3[0](1.314008+0.038702=1.352710) / /          V1[524287]+V2[524287]=V3[524287](3.619231+7.628835=11.248066) /
Tiempo(seg.):0.003409529        / Tamaño Vectores:1048576      / V1[0]+V2[0]=V3[0](1.314008+0.038702=1.352710) / /          V1[1048575]+V2[1048575]=V3[1048575](1.782376+0.895122=2.677498) /
Tiempo(seg.):0.006861596        / Tamaño Vectores:2097152      / V1[0]+V2[0]=V3[0](1.314008+0.038702=1.352710) / /          V1[2097151]+V2[2097151]=V3[2097151](0.363267+0.429150=0.792417) /
Tiempo(seg.):0.014702092        / Tamaño Vectores:4194304      / V1[0]+V2[0]=V3[0](1.314008+0.038702=1.352710) / /          V1[4194303]+V2[4194303]=V3[4194303](1.028143+1.389216=2.417360) /
Tiempo(seg.):0.031986452        / Tamaño Vectores:8388608      / V1[0]+V2[0]=V3[0](1.314008+0.038702=1.352710) / /          V1[8388607]+V2[8388607]=V3[8388607](140.802239+2.155379=142.957618) /
Tiempo(seg.):0.064244778        / Tamaño Vectores:16777216     / V1[0]+V2[0]=V3[0](0.281303+15.385194=15.666497) / /          V1[16777215]+V2[16777215]=V3[16777215](1.238764+6.936580=8.175344) /
Tiempo(seg.):0.117918090        / Tamaño Vectores:33554432     / V1[0]+V2[0]=V3[0](0.281303+15.385194=15.666497) / /          V1[33554431]+V2[33554431]=V3[33554431](0.189467+1.434802=1.624269) /
Tiempo(seg.):0.113628260        / Tamaño Vectores:33554432     / V1[0]+V2[0]=V3[0](1.028635+2.169666=3.198301) / /          V1[33554431]+V2[33554431]=V3[33554431](0.335512+0.862063=1.197576) /
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0/ejer7] 2021-03-11 jueves
$
```

No, sin embargo el tamaño de estos vectores esta capado por una variable global, es decir, si se supera, el sistema toma el máximo posible como el tamaño del vector.

(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:

```
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0/ejer7] 2021-03-11 jueves
$ gcc -o listado -fopenmp -O2 Listado.c
Listado.c: In function ‘main’:
Listado.c:86:56: warning: format ‘%lu’ expects argument of type ‘long unsigned int’, but argument 3 has type ‘unsigned int’ [-Wformat=]
   86 |     printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%lu\n", ncgt, N);
      |                                                      ~~^   ~
      |                                                        |   |
      |                                                        |   unsigned int
      |                                                        long unsigned int
      |                                                      %u
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0/ejer7] 2021-03-11 jueves
$ ./script.sh
Tiempo(seg.):0.000739783        / Tamaño Vectores:65536        / V1[0]+V2[0]=V3[0](0.587298+0.561778=1.149076) / /          V1[65535]+V2[65535]=V3[65535](48.710902+0.817988=49.528891) /
Tiempo(seg.):0.001180513        / Tamaño Vectores:131072       / V1[0]+V2[0]=V3[0](0.587298+0.561778=1.149076) / /          V1[131071]+V2[131071]=V3[131071](0.443981+1.499369=1.943351) /
Tiempo(seg.):0.001135666        / Tamaño Vectores:262144       / V1[0]+V2[0]=V3[0](0.587298+0.561778=1.149076) / /          V1[262143]+V2[262143]=V3[262143](1.071402+1.135795=2.207197) /
Tiempo(seg.):0.001827518        / Tamaño Vectores:524288       / V1[0]+V2[0]=V3[0](0.587298+0.561778=1.149076) / /          V1[524287]+V2[524287]=V3[524287](1.081826+0.872228=1.954054) /
Tiempo(seg.):0.003769192        / Tamaño Vectores:1048576      / V1[0]+V2[0]=V3[0](0.587298+0.561778=1.149076) / /          V1[1048575]+V2[1048575]=V3[1048575](0.906759+4.201285=5.108044) /
Tiempo(seg.):0.007795086        / Tamaño Vectores:2097152      / V1[0]+V2[0]=V3[0](0.587298+0.561778=1.149076) / /          V1[2097151]+V2[2097151]=V3[2097151](1.397961+0.384864=1.782825) /
Tiempo(seg.):0.013465917        / Tamaño Vectores:4194304      / V1[0]+V2[0]=V3[0](0.587298+0.561778=1.149076) / /          V1[4194303]+V2[4194303]=V3[4194303](23.328059+1.417266=24.745325) /
Tiempo(seg.):0.029440378        / Tamaño Vectores:8388608      / V1[0]+V2[0]=V3[0](0.587298+0.561778=1.149076) / /          V1[8388607]+V2[8388607]=V3[8388607](3.265855+1.840338=5.046193) /
Tiempo(seg.):0.053612924        / Tamaño Vectores:16777216     / V1[0]+V2[0]=V3[0](1.204249+0.440663=1.644912) / /          V1[16777215]+V2[16777215]=V3[16777215](0.877192+0.672738=1.549930) /
Tiempo(seg.):0.111606577        / Tamaño Vectores:33554432     / V1[0]+V2[0]=V3[0](1.204249+0.440663=1.644912) / /          V1[33554431]+V2[33554431]=V3[33554431](0.215155+1.615509=1.830665) /
Tiempo(seg.):0.221783694        / Tamaño Vectores:67108864     / V1[0]+V2[0]=V3[0](0.520137+1.021991=1.542127) / /          V1[67108863]+V2[67108863]=V3[67108863](3.883897+9.889472=13.773369) /
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0/ejer7] 2021-03-11 jueves
$
```

No hay ningún problema, ya que los vectores dinámicos se ubican en el heap, donde tienen suficiente espacio para existir

3. **(a)** ¿Cuál es el máximo valor que se puede almacenar en la variable N teniendo en cuenta su tipo? Razonar respuesta.
RESPUESTA: N se guarda como un tipo int, por lo que su valor máximo es 2147483647 (cada entero son 4 bytes)

(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: El código no compila, ya que al ser el tamaño tan grande, no podemos ubicar los vectores en memoria.

```
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0/ejer7] 2021-03-11 jueves
$ gcc -o listado -fopenmp -O2 Listado.c
Listado.c: In function ‘main’:
Listado.c:86:56: warning: format ‘%lu’ expects argument of type ‘long unsigned int’, but argument 3 has type ‘unsigned int’ [-Wformat=]
   86 |     printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%lu\n", ncgt, N);
      |                                                      ~~^       ~
      |                                                        |       |
      |                                                        |       unsigned int
      |                                                        long unsigned int
      |                                                       %u
/tmp/ccghZJVM.o: en la función `main':
Listado.c:(.text.startup+0x6d): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v2' definido en la sección COMMON en /tmp/ccghZJVM.o
Listado.c:(.text.startup+0xbc): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v3' definido en la sección COMMON en /tmp/ccghZJVM.o
Listado.c:(.text.startup+0x1b5): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v2' definido en la sección COMMON en /tmp/ccghZJVM.o
collect2: error: ld returned 1 exit status
[DanielAlconchelVázquez daniel@daniel-GL63-8SE:~/Git/DGIIM/Segundo/2 Cuatrimestre/AC/Prácticas/bp0/ejer7] 2021-03-11 jueves
$
```

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.):%11.9f\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;
}
```