

Arquitectura de Computadores (AC)

Cuaderno de prácticas.

Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos): Quintín Mesa Romero

Grupo de prácticas y profesor de prácticas: A 1. Carlos Megías

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 que con sbatch se tenga en cuenta `---hint=nomultithread` se debe usar srun dentro del script delante del ejecutable.
- 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)
- Se recomienda escribir las órdenes directamente en la ventana de comandos (*shell*) en lugar de usar copy/paste.

1. Ejecutar `lscpu` en el PC, en atcgrid4 (usar en este caso `-p ac4`) y en uno de los restantes nodos de cómputo (atcgrid1, atcgrid2 o atcgrid3, usar en este caso `-p ac`).

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

RESPUESTA:

```
[ac425@atcgrid ~]$ srun -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:               1199.450
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 dt
s 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 nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est
tm2_5ssse3 sdbg fma cx16 xtpr pdcm pcid dca sse4_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_ppt intel_pt ssbd mba
ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 sme
p bmi2 erms invpcid rtm cqm mpx rdt_a avx512f avx512dq rdseed adx snap clflushopt clwb avx512cd 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
[ac425@atcgrid ~]$
```

```

[ac425@atcgrid ~]$ 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) Xeon(R) CPU E3-1230 v6 @ 3.50GHz
Stepping:                9
CPU MHz:                 1599.395
CPU max MHz:             3900.0000
CPU min MHz:             800.0000
BogoMIPS:                7008.00
Virtualization:          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 dt
s 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 nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est
tm2 sse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx
f16c rdrand lahf_lm abm 3dnowprefetch epb invpcid_single intel_pt ssbd ibrs ibpb stibp tpr_shadow vnmi f
lexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm mpx rdseed adx smap clf
lushopt xsaveopt xsavec xgetbv1 dtherm ida arat pln pts hwp hwp_notify hwp_act_window hwp_epp md_clear sp
ec_ctrl intel_stibp flush_lid
[ac425@atcgrid ~]$

```

```

[ac425@atcgrid ~]$ srun -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:                4799.64
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 sse3 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_lid
[ac425@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:

- En mi PC: 8 lógicos y 4 físicos
- En atcgrid4: 16 y 64 respectivamente
- En atcgrid1, atcgrid2 y atcgrid3: 6 y 24 respectivamente

2. Compilar y ejecutar en el PC el código HelloOMP.c del seminario.

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

```

[ac425@atcgrid clase0]$ gcc -O2 -fopenmp -o HelloOMP HelloOMP.c
[ac425@atcgrid clase0]$ ls
HelloOMP HelloOMP.c srun-115542.out
[ac425@atcgrid clase0]$ ./HelloOMP
(0!!!Hello world!!!)(1!!!Hello world!!!)(6!!!Hello world!!!)(7!!!Hello world!!!)(4!!!Hello world!!!)(3!!!Hello world!!!)(5!!!Hello world!!!)(2!!!Hello world!!!)[ac425@atcgrid clase0]$ ^C
[ac425@atcgrid clase0]$ srun -p ac4 HelloOMP
(0!!!Hello world!!!)(1!!!Hello world!!!)[ac425@atcgrid clase0]$ ^C
[ac425@atcgrid clase0]$ srun -p ac4 -c 12 HelloOMP
(1!!!Hello world!!!)(7!!!Hello world!!!)(10!!!Hello world!!!)(5!!!Hello world!!!)(0!!!Hello world!!!)(9!!!Hello world!!!)(2!!!Hello world!!!)(4!!!Hello world!!!)(8!!!Hello world!!!)(6!!!Hello world!!!)
(3!!!Hello world!!!)(1!!!Hello world!!!)[ac425@atcgrid clase0]$ srun -p ac4 -c 64 HelloOMP
(7!!!Hello world!!!)(45!!!Hello world!!!)(40!!!Hello world!!!)(63!!!Hello world!!!)(51!!!Hello world!!!)(35!!!Hello world!!!)(20!!!Hello world!!!)(4!!!Hello world!!!)(37!!!Hello world!!!)(54!!!Hello wo
rld!!!)(12!!!Hello world!!!)(8!!!Hello world!!!)(10!!!Hello world!!!)(42!!!Hello world!!!)(61!!!Hello world!!!)(18!!!Hello world!!!)(22!!!Hello world!!!)(49!!!Hello world!!!)(58!!!
Hello world!!!)(9!!!Hello world!!!)(39!!!Hello world!!!)(1!!!Hello world!!!)(3!!!Hello world!!!)(43!!!Hello world!!!)(21!!!Hello world!!!)(62!!!Hello world!!!)(11!!!Hello world!!!)(41!!!Hello world!!!)(
56!!!Hello world!!!)(52!!!Hello world!!!)(15!!!Hello world!!!)(16!!!Hello world!!!)(44!!!Hello world!!!)(29!!!Hello world!!!)(55!!!Hello world!!!)(59!!!Hello world!!!)(36!!!Hello w
orld!!!)(34!!!Hello world!!!)(26!!!Hello world!!!)(50!!!Hello world!!!)(13!!!Hello world!!!)(5!!!Hello world!!!)(53!!!Hello world!!!)(60!!!Hello world!!!)(30!!!Hello world!!!)(2!!!Hello world!!!)(32!!!
Hello world!!!)(24!!!Hello world!!!)(46!!!Hello world!!!)(31!!!Hello world!!!)(28!!!Hello world!!!)(23!!!Hello world!!!)(25!!!Hello world!!!)(38!!!Hello world!!!)(48!!!Hello world!!!)(19!!!Hello world
!!!)(6!!!Hello world!!!)(0!!!Hello world!!!)(33!!!Hello world!!!)(14!!!Hello world!!!)(47!!!Hello world!!!)(27!!!Hello world!!!)[ac425@atcgrid clase0]$

```

(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:

```
[ac425@atcgrid clase0]$ srun --partition=ac --account=ac --ntasks=1 --cpus-per-task=12 --hint=nomultithread HelloOMP
(8:!!!Hello world!!!)(4:!!!Hello world!!!)(1:!!!Hello world!!!)(6:!!!Hello world!!!)(9:!!!Hello world!!!)
(11:!!!Hello world!!!)(10:!!!Hello world!!!)(0:!!!Hello world!!!)(5:!!!Hello world!!!)(3:!!!Hello world!!!)
(7:!!!Hello world!!!)(2:!!!Hello world!!!)[ac425@atcgrid clase0]$
```

(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:

```
ac425@atcgrid:~/clase0
[ac425@atcgrid clase0]$ srun -pac -Aac -n1 -c24 HelloOMP
(0:!!!Hello world!!!)(7:!!!Hello world!!!)(10:!!!Hello world!!!)(22:!!!Hello world!!!)(5:!!!Hello world!!!)
(2:!!!Hello world!!!)(16:!!!Hello world!!!)(20:!!!Hello world!!!)(12:!!!Hello world!!!)(19:!!!Hello world!!!)
(21:!!!Hello world!!!)(8:!!!Hello world!!!)(14:!!!Hello world!!!)(18:!!!Hello world!!!)(3:!!!Hello world!!!)
(13:!!!Hello world!!!)(1:!!!Hello world!!!)(6:!!!Hello world!!!)(15:!!!Hello world!!!)(11:!!!Hello world!!!)
(9:!!!Hello world!!!)(4:!!!Hello world!!!)(17:!!!Hello world!!!)(23:!!!Hello world!!!)[ac425@atcgrid clase0]$
```

(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 (cola) se está usando?

RESPUESTA:

```
ac425@atcgrid:~/clase0
[ac425@atcgrid clase0]$ srun -n1 HelloOMP
(1:!!!Hello world!!!)(0:!!!Hello world!!!)[ac425@atcgrid clase0]$
```

(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 ac -c 12 HelloMp`

- 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 `ej4`). 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 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:

```

/**
 *
 * Compilar con gcc -O2 -fopenmp HelloOMP.c -o HelloOMP
 *
 */

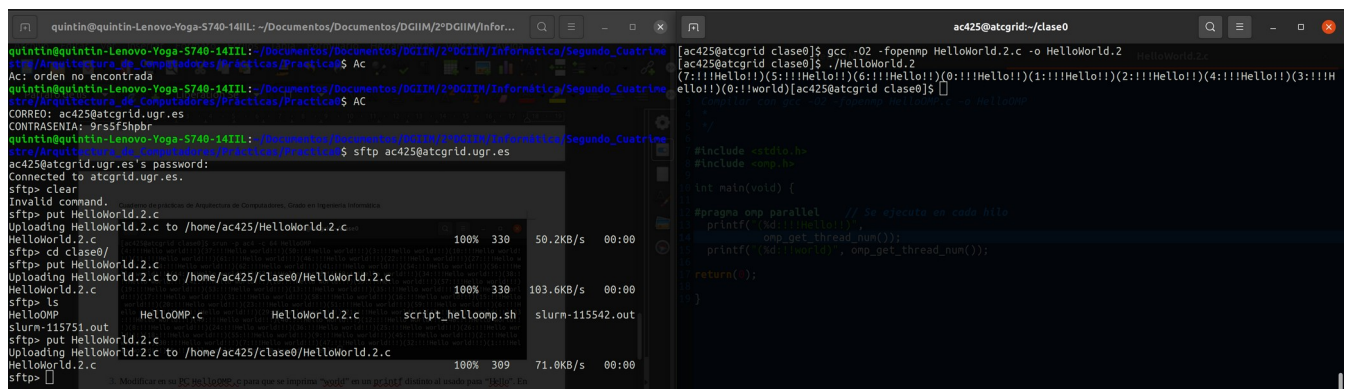
#include <stdio.h>
#include <omp.h>

int main(void) {

#pragma omp parallel // Se ejecuta en cada hilo
printf("(%d:!!!Hello!!!)",
        omp_get_thread_num());
printf("(%d:!!world)", omp_get_thread_num());

return(0);
}

```



```

quintin@quintin-Lenovo-Yoga-5740-14IIL: ~/Documentos/Documentos/DGIIIM/2ºDGIIIM/Infer...
quintin@quintin-Lenovo-Yoga-5740-14IIL: ~/Documentos/Documentos/DGIIIM/2ºDGIIIM/Infer...
quintin@quintin-Lenovo-Yoga-5740-14IIL: ~/Documentos/Documentos/DGIIIM/2ºDGIIIM/Infer...
quintin@quintin-Lenovo-Yoga-5740-14IIL: ~/Documentos/Documentos/DGIIIM/2ºDGIIIM/Infer...
CORREO: ac425@atcgird.ugr.es
CONTRASEÑA: 9rs5fshpbr
quintin@quintin-Lenovo-Yoga-5740-14IIL: ~/Documentos/Documentos/DGIIIM/2ºDGIIIM/Infer...
ac425@atcgird.ugr.es's password:
Connected to atcgird.ugr.es.
sftp> clear
Invalid command.
sftp> put HelloWorld.2.c
Uploading HelloWorld.2.c to /home/ac425/HelloWorld.2.c
100% 330 50.2KB/s 00:00
sftp> cd clase0/
sftp> put HelloWorld.2.c
Uploading HelloWorld.2.c to /home/ac425/clase0/HelloWorld.2.c
100% 330 103.6KB/s 00:00
sftp> ls
HelloOMP.c HelloWorld.2.c script_helloomp.sh slurm-115751.out
sftp> put HelloWorld.2.c
Uploading HelloWorld.2.c to /home/ac425/clase0/HelloWorld.2.c
100% 309 71.0KB/s 00:00
sftp>

```



```

ac425@atcgrid:~/clase0
[ac425@atcgrid clase0]$ sbatch script_helloomp.sh
Submitted batch job 116337
[ac425@atcgrid clase0]$ ls
HelloOMP      HelloWorld.2  script_helloomp.sh  slurm-115751.out  slurm-116337.out
HelloOMP.c    HelloWorld.2.c  slurm-115542.out    slurm-116333.out
[ac425@atcgrid clase0]$ ls -c
slurm-116337.out  HelloWorld.2  slurm-116333.out  HelloOMP  slurm-115542.out
script_helloomp.sh  HelloWorld.2.c  slurm-115751.out  HelloOMP.c
[ac425@atcgrid clase0]$ cat slurm-116337.out
Id. usuario del trabajo: ac425
Id. del trabajo: 116337
Nombre del trabajo especificado por usuario: helloOMP
Directorio de trabajo (en el que se ejecuta el script): /home/ac425/clase0
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)

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

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

- Para 12 threads:

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

(0:!!!Hello world!!!)(3:!!!Hello world!!!)(5:!!!Hello world!!!)(2:!!!Hello world!!!)(4:!!!Hello world!!!)
(1:!!!Hello world!!!) - Para 3 threads:

(0:!!!Hello world!!!)(2:!!!Hello world!!!)(1:!!!Hello world!!!) - Para 1 threads:

(0:!!!Hello world!!!)[ac425@atcgrid clase0]$

```

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

RESPUESTA:

El script lo está ejecutando el nodo **atcgrid1**. Se ve en la captura anterior.

(c) ¿Qué órdenes para el gestor de colas slurm incluye el *script*? Explicar cómo ha obtenido esta información.

RESPUESTA:

(d) Haga los cambios necesarios en el *script* para que se utilice atcgrid4. Comentar los cambios realizados y los motivos por los que se han hecho.

RESPUESTA:

NOTA: Utilizar siempre con `sbatch` las opciones `-n1` y `-c`, `--exclusive` y, para usar cores físicos y no lógicos, no olvidar 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 `SumaVectores.c` 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`). 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:

```
quintin@quintin-Lenovo-Yoga-S740-14IIL: ~/Documentos/Documentos...
quintin@quintin-Lenovo-Yoga-S740-14IIL:~/Documentos/Documentos/DGIIM/2ºDGIIM/Informática/Segundo_Cuatrimestre/Arquitectura_de_Computadores/Prácticas/Practica0/bp0$ gcc -O2 SumaVectores.c -o SumaVectores -lrt
quintin@quintin-Lenovo-Yoga-S740-14IIL:~/Documentos/Documentos/DGIIM/2ºDGIIM/Informática/Segundo_Cuatrimestre/Arquitectura_de_Computadores/Prácticas/Practica0/bp0$ ./SumaVectores
Tamaño Vectores:5 (4 B)
Tiempo:0.000000306 / Tamaño Vectores:5
/ V1[0]+V2[0]=V3[0](0.500000+0.500000=1.000000) /
/ V1[1]+V2[1]=V3[1](0.600000+0.400000=1.000000) /
/ V1[2]+V2[2]=V3[2](0.700000+0.300000=1.000000) /
/ V1[3]+V2[3]=V3[3](0.800000+0.200000=1.000000) /
/ V1[4]+V2[4]=V3[4](0.900000+0.100000=1.000000) /
```

6. En el código `SumaVectores.c` 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 `SumaVectores.c` para vectores locales, globales y dinámicos (se pueden obtener errores en tiempo de ejecución o

T	Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos	ue hay en el E5645 y otras reservas que usan
	65536	4				
	131072					
	262144					
	524288					
	1048576					
	2097152					
	4194304					
	8388608					
	16777216					
	33554432					
	67108864					

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:

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:

(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:

(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:

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

RESPUESTA:

(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:

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. <i>Se generan aleatoriamente las componentes para vectores de tamaño mayor</i>
--

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(), 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)

#ifndef 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
{
    srand48(time(0));
    for (i = 0; i < N; i++)
    {
        v1[i] = drand48();
        v2[i] = drand48(); //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;
}

```