2º curso / 2º cuatr.

Grado Ingeniería Informática

Arquitectura de Computadores (AC)

Cuaderno de prácticas.
Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos): David Martínez Díaz
Grupo de prácticas y profesor de prácticas: Grupo 2 – Juan José Escobar
Fecha de entrega: 15/03/2021
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:

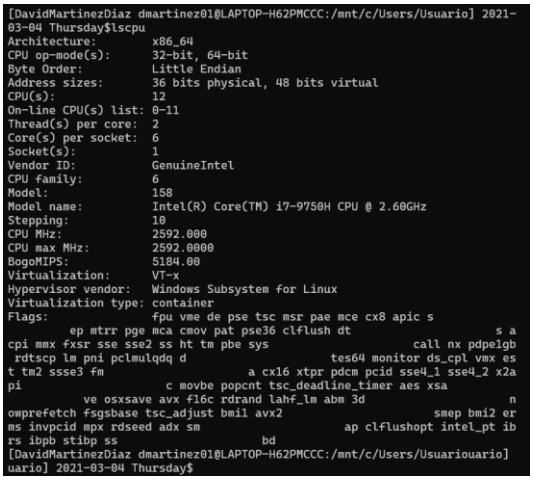
- Slurm está configurado para asignar recursos a los procesos (llamados *tasks* en slurm) a nivel de core físico. Esto significa que por defecto slurm asigna un core a un proceso, para asignar x se debe usar con sbatch/srun la opción --cpus-per-task=x (-cx).
- En slurm, por defecto, cpu se refiere a 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:

Lscpu → pc



Lscpu → ac4

```
[e2estudiante14@atcgrid bp0]$ 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:                 1177.020
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 pse3
6 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc a
rt arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqd
q dtes64 monitor ds_cpl vmx smx est tm2 ssse3 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 ep
b cat_l3 cdp_l3 invpcid_single intel_ppin intel_pt ssbd mba ibrs ibpb stibp ibrs_enhanced tpr
_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid r
tm cqm mpx rdt_a avx512f avx512dq rdseed adx smap clflushopt clwb avx512cd avx512bw avx512vl
xsaveopt xsavec xgetbv1 cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local dtherm ida arat pln
 pts pku ospke avx512_vnni md_clear spec_ctrl intel_stibp flush_l1d arch_capabilities
[e2estudiante14@atcgrid bp0]$
```

Lscpu → ac(1-3)

```
[e2estudiante14@atcgrid bp0]$ 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 pse3
6 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc a
rch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni dtes64 monit
or 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
[e2estudiante14@atcgrid bp0]$
```

Directorio creado → ejer1

```
[e2estudiante14@atcgrid bp0]$ mkdir ejer1
[e2estudiante14@atcgrid bp0]$ ls
HelloOMP  ejer1
[e2estudiante14@atcgrid bp0]$
```

(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:

Justificación: Para saber cuántos cores físicos tengo me fijo en los (cores per socket) y como sabemos que solo tenemos una cpu, directamente es el número de cores físicos. Y para conocer el de los cores lógicos, se obtiene con los números de hilos de procesamiento por nucleo y lo multiploco por el número de físicos que tenga.

Para el atcgrid4: tenemos 16 cores físicos y 16*2 = 32 cores lógicos en el atcgrid4.

Para el atcgrid1-3: tenemos 6 cores físicos y 6*2 = 12 cores lógicos en los atcgrid1-3.

Para el PC: tenemos 6 cores físicos y 12 cores lógicos en el pc.

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:

```
[DavidMartinezDiaz dmartinez01@LAPTOP-H62PMCCC:/mnt/c/Users/Usuario/AC_Pra
cticas/bp0/ejer2] 2021-03-04 Thursday$ gcc -O2 -fopenmp -o HelloOMP HelloO
MP.c
[DavidMartinezDiaz dmartinez01@LAPTOP-H62PMCCC:/mnt/c/Users/Usuario/AC_Pra
cticas/bp0/ejer2] 2021-03-04 Thursday$ls
HelloOMP  HelloOMP.c
[DavidMartinezDiaz dmartinez01@LAPTOP-H62PMCCC:/mnt/c/Users/Usuario/AC_Pra
cticas/bp0/ejer2] 2021-03-04 Thursday$./HelloOMP
(0:!!!Hello world!!!)(4:!!!Hello world!!!)(11:!!!Hello world!!!)(6:!!!Hell
o world!!!)(8:!!!Hello world!!!)(10:!!!Hello world!!!)(1:!!!Hello world!!!
)(7:!!!Hello world!!!)(2:!!!Hello world!!!)(5:!!!Hello world!!!)(3:!!!Hell
o world!!!)(9:!!!Hello world!!!)[DavidMartinezDiaz dmartinez01@LAPTOP-H62P
MCCC:/mnt/c/Users/Usuario/AC_Practicas/bp0/ejer2] 2021-03-04 Thursday$
```

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

RESPUESTA: Utiliza del 0 al 11 hebras, lo que son enrealidad 12 hebras, que se ejecutan en paralelo indicando el numero máximo que puede crear el PC, ya que lo hemos visto anteriormente con el lscpu.

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:

Copiamos el ejecutable al atcgrid y al directorio ejer3:

```
[DavidMartinezDiaz dmartinez01@LAPTOP-H62PMCCC:/mnt/c/Users/Usuario/AC_Practicas/bp0/ejer2]
 2021-03-04 Thursday$ls
HelloOMP  HelloOMP.c
[DavidMartinezDiaz dmartinez01@LAPTOP-H62PMCCC:/mnt/c/Users/Usuario/AC_Practicas/bp0/ejer2]
 2021-03-04 Thursday$sftp e2estudiante14@atcgrid.ugr.es
e2estudiante14@atcgrid.ugr.es's password:
Connected to atcgrid.ugr.es.
sftp> put
HelloOMP     HelloOMP.c
sftp> put He
HelloOMP     HelloOMP.c
sftp> put HelloOMP
Uploading HelloOMP to /home/e2estudiante14/HelloOMP
HelloOMP                                      100%   17KB 164.4KB/s   00:00
sftp>
```

```
[e2estudiante14@atcgrid ejer2]$ cp HelloOMP ../ejer3/
[e2estudiante14@atcgrid ejer2]$ cd ../ejer3
[e2estudiante14@atcgrid ejer3]$ ls
HelloOMP
[e2estudiante14@atcgrid ejer3]$
```

Ejecución del apartado a:

```
[e2estudiante14@atcgrid ejer3]$ srun --partition=ac --account=ac --ntasks=1 --cpus-per-task=12 --hint=nomultithrea
d HelloOMP
(0:!!!Hello world!!!)(8:!!!Hello world!!!)(2:!!!Hello world!!!)(5:!!!Hello world!!!)(7:!!!Hello world!!!)(4:!!!Hel
lo world!!!)(1:!!!Hello world!!!)(11:!!!Hello world!!!)(6:!!!Hello world!!!)(9:!!!Hello world!!!)(10:!!!Hello worl
d!!!)(3:!!!Hello world!!!)[e2estudiante14@atcgrid ejer3]$ s
```

(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:

```
[e2estudiante14@atcgrid ejer3]$ srun -p ac -n1 --cpus-per-task=24 HelloOMP
(22:!!!Hello world!!!)(15:!!!Hello world!!!)(1:!!!Hello world!!!)(5:!!!Hello world!!!)(23:!!!Hello world!!!)(21:!!
!Hello world!!!)(7:!!!Hello world!!!)(12:!!!Hello world!!!)(16:!!!Hello world!!!)(4:!!!Hello world!!!)(8:!!!Hello
world!!!)(13:!!!Hello world!!!)(6:!!!Hello world!!!)(10:!!!Hello world!!!)(20:!!!Hello world!!!)(14:!!!Hello world
!!!)(2:!!!Hello world!!!)(11:!!!Hello world!!!)(17:!!!Hello world!!!)(19:!!!Hello world!!!)(9:!!!Hello world!!!)(3
:!!!Hello world!!!)(0:!!!Hello world!!!)(18:!!!Hello world!!!)[e2estudiante14@atcgrid ejer3]$
```

(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:

Creo que se esta usando la partición del atcgrid1-3

```
[e2estudiante14@atcgrid ejer3]$ srun -n1 HelloOMP
(0:!!!Hello world!!!)(1:!!!Hello world!!!)[e2estudiante14@atcgrid ejer3]$
```

(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 –n1 –cpus-per-task=32 –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.

RESPUESTA:

HelloOMP2.c HelloOMP.c

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

int main (void){

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

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

    return (0);
}
```

→ Código HelloOMP2

→ Compilacion del HelloOMP2

```
[DavidMartinezDiaz dmartinez01@LAPTOP-H62PMCCC:/mnt/c/Users/Usuario/AC_Practicas/bp0/ejer2]
 2021-03-04 Thursday$ls
HelloOMP  HelloOMP.c  HelloOMP2  HelloOMP2.c
[DavidMartinezDiaz dmartinez01@LAPTOP-H62PMCCC:/mnt/c/Users/Usuario/AC_Practicas/bp0/ejer2]
 2021-03-04 Thursday$s
```

```
[DavidMartinezDiaz dmartinez01@LAPTOP-H62PMCCC:/mnt/c/Users/Usuario/AC_Practicas/bp0/ejer2]
 2021-03-04 Thursday$./HelloOMP2
(0:!!!Hello!!!)(6:!!!Hello!!!)(9:!!!Hello!!!)(5:!!!Hello!!!)(1:!!!Hello!!!)(4:!!!Hello!!!)(
8:!!!Hello!!!)(10:!!!Hello!!!)(11:!!!Hello!!!)(2:!!!Hello!!!)(7:!!!Hello!!!)(3:!!!Hello!!!)
(3:!!!World!!!)(0:!!!World!!!)(2:!!!World!!!)(9:!!!World!!!)(11:!!!World!!!)(6:!!!World!!!)
(1:!!!World!!!)(10:!!!World!!!)(8:!!!World!!!)(5:!!!World!!!)(7:!!!World!!!)(4:!!!World!!!)
[DavidMartinezDiaz dmartinez01@LAPTOP-H62PMCCC:/mnt/c/Users/Usuario/AC_Practicas/bp0/ejer2]
 2021-03-04 Thursday$
```

→ Lo subimos a atcgrid

```
[DavidMartinezDiaz dmartinez01@LAPTOP-H62PMCCC:/mnt/c/Users/Usuario/AC_Practicas/bp0/ejer2]
 2021-03-04 Thursday$sftp e2estudiante14@atcgrid.ugr.es
e2estudiante14@atcgrid.ugr.es's password:
Connected to atcgrid.ugr.es.
sftp> put He
HelloOMP     HelloOMP.c   HelloOMP2    HelloOMP2.c
sftp> put HelloOMP2
Uploading HelloOMP2 to /home/e2estudiante14/HelloOMP2
HelloOMP2                                     100%   17KB 169.9KB/s   00:00
sftp>
```

```
sftp> put script_helloomp.sh
Uploading script_helloomp.sh to /home/e2estudiante14/script_helloomp.sh
script_helloomp.sh                            100% 1187    23.5KB/s   00:00
sftp>
```

→ Lo ejecutamos:

```
[e2estudiante14@atcgrid ejer4]$ sbatch -p ac -n1 --cpus-per-task=12 --hint=nomultithread script_helloomp.sh
Submitted batch job 59237
[e2estudiante14@atcgrid ejer4]$ cat slurm-59237.out
Id. usuario del trabajo: e2estudiante14
Id. del trabajo: 59237
Nombre del trabajo especificado por usuario: helloOMP
Directorio de trabajo (en el que se ejecuta el script): /home/e2estudiante14/AC/bp0/ejer4
Cola: ac
Nodo que ejecuta este trabajo:atcgrid.ugr.es
N° de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgrid1
CPUs por nodo: 24

 1. Ejecución helloOMP una vez sin cambiar n° de threads (valor por defecto):

(4:!!!Hello!!!)(0:!!!Hello!!!)(11:!!!Hello!!!)(3:!!!Hello!!!)(5:!!!Hello!!!)(8:!!!Hello!!!)(2:!!!Hello!!!)(1:!!!He
llo!!!)(6:!!!Hello!!!)(9:!!!Hello!!!)(7:!!!Hello!!!)(10:!!!Hello!!!)(4:!!!World!!!)(10:!!!World!!!)(2:!!!World!!!)
(6:!!!World!!!)(0:!!!World!!!)(11:!!!World!!!)(3:!!!World!!!)(7:!!!World!!!)(5:!!!World!!!)(9:!!!World!!!)(8:!!!Wo
rld!!!)(1:!!!World!!!)
 2. Ejecución helloOMP varias veces con distinto n° de threads:


 - Para 12 threads:
(11:!!!Hello!!!)(10:!!!Hello!!!)(2:!!!Hello!!!)(1:!!!Hello!!!)(0:!!!Hello!!!)(4:!!!Hello!!!)(9:!!!Hello!!!)(6:!!!H
ello!!!)(5:!!!Hello!!!)(7:!!!Hello!!!)(3:!!!Hello!!!)(8:!!!Hello!!!)(4:!!!World!!!)(5:!!!World!!!)(6:!!!World!!!)(
10:!!!World!!!)(2:!!!World!!!)(9:!!!World!!!)(11:!!!World!!!)(3:!!!World!!!)(8:!!!World!!!)(0:!!!World!!!)(1:!!!Wo
rld!!!)(7:!!!World!!!)
 - Para 6 threads:
(3:!!!Hello!!!)(1:!!!Hello!!!)(4:!!!Hello!!!)(0:!!!Hello!!!)(2:!!!Hello!!!)(5:!!!Hello!!!)(3:!!!World!!!)(2:!!!Wor
ld!!!)(5:!!!World!!!)(1:!!!World!!!)(4:!!!World!!!)(0:!!!World!!!)
 - Para 3 threads:
(1:!!!Hello!!!)(0:!!!Hello!!!)(2:!!!Hello!!!)(1:!!!World!!!)(0:!!!World!!!)(2:!!!World!!!)
 - Para 1 threads:
(0:!!!Hello!!!)(0:!!!World!!!)[e2estudiante14@atcgrid ejer4]$
```

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

RESPUESTA: El nodo del computo atcgrid1, mostrado en el script en la variables $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:

```
[DavidMartinezDiaz dmartinez01@LAPTOP-H62PMCCC:/mnt/c/User
s/Usuario/AC_Practicas/bp0/ejer5] 2021-03-05 Friday$gcc -O
2 SumaVectoresC.c -o SumaVectores -lrt
[DavidMartinezDiaz dmartinez01@LAPTOP-H62PMCCC:/mnt/c/User
s/Usuario/AC_Practicas/bp0/ejer5] 2021-03-05 Friday$
```

```
[DavidMartinezDiaz dmartinez01@LAPTOP-H62PMCCC:/mnt/c/User
s/Usuario/AC_Practicas/bp0/ejer5] 2021-03-05 Friday$./Suma
Vectores 100
Tamanio Vectores:100 (4 B)
Tiempo:0.000001400       / Tamanio Vectores:100 / V1[0]+V2
[0]=V3[0](10.000000+10.000000=20.000000) / / V1[99]+V2[99]
=V3[99](19.900000+0.100000=20.000000) /
[DavidMartinezDiaz dmartinez01@LAPTOP-H62PMCCC:/mnt/c/User
s/Usuario/AC_Practicas/bp0/ejer5] 2021-03-05 Friday$
```

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:
Esta variable que es de tipo double contiene el tiempo que hay ya que es la suma de segundos más los nanosegundos, y se guarda en ncgt. Por asi decirlo, es el tiempo que tarda en hacer la suma de los 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:
Esta se trata de una estructura → "struct timespec", y tiene dos elementos: primero tenemos uno de tipo time_t que almacena en segundos y el segundo de tipo long que almacena en 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 funcion clock es la siguiente gettime(clockid_t clk_id, struct timespec *tp):

Devuelve 0 si se ha ejecutado correctamente, y 1 si no se ha ejecutado correctamente.

Donde clockid_t clk_id → En este caso seria CLOCK_REALTIME que es el instante de tiempo.

Struct timespec *tp → Como lo hemos definido posteriormente, guarda el tiempo en segundos y en nanosegundos.

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 . Copiar la tabla de la hoja de cálculo utilizada → PARA PC

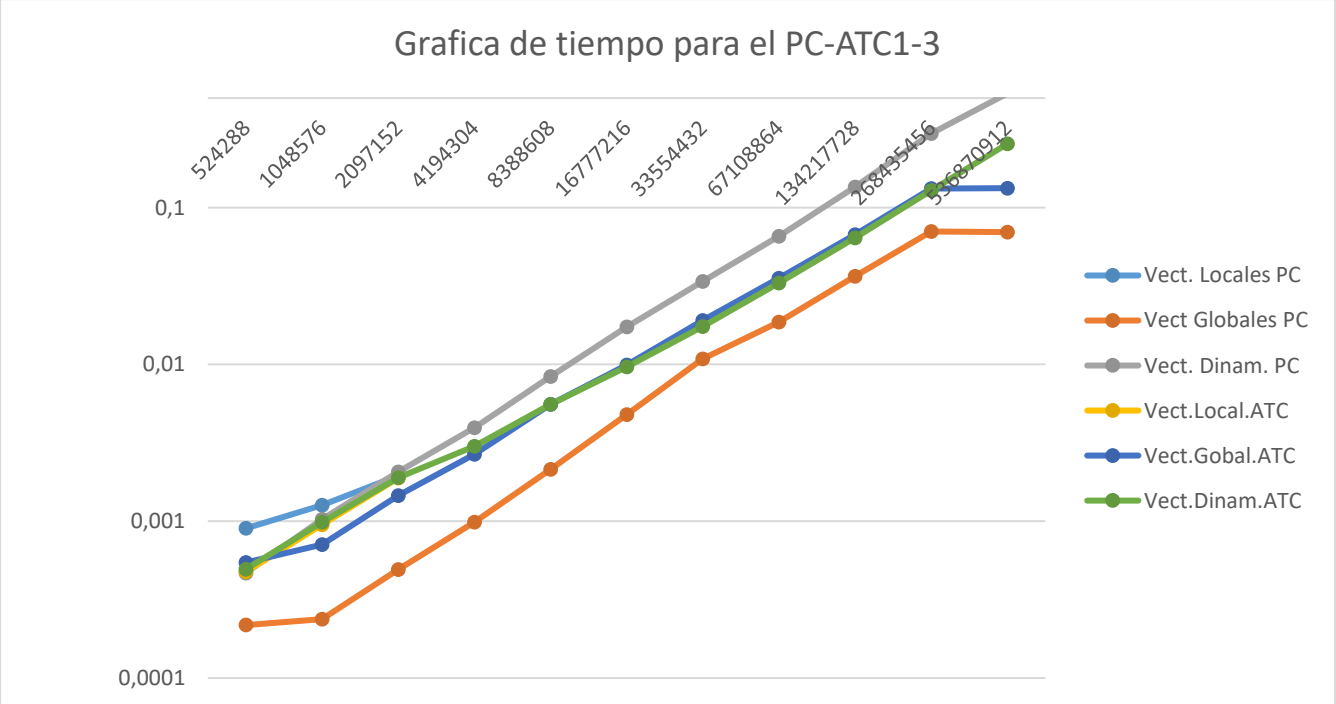
Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000900800	0.000217700	0.000467700
131072	1048576	0.001267300	0.000238000	0.001028100
262144	2097152	0.001953200	0.000492400	0.002066100
524288	4194304	0	0.000988700	0.003943400
1048576	8388608	0	0.002141500	0.008372900
2097152	16777216	0	0.004787800	0.017377300
4194304	33554432	0	0.010839300	0.033843400
8388608	67108864	0	0.018607200	0.065611400
16777216	134217728	0	0.036447300	0.135473200
33554432	268435456	0	0.070451900	0.296745100
67108864	536870912	0	0.069915100	0.534960600

Tabla 2 . Copiar la tabla de la hoja de cálculo utilizada → PARA ATC1-3 → XEON E5645

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000475597	0.000545191	0.000494366
131072	1048576	0.000948897	0.000709896	0.000987051
262144	2097152	0.001884045	0.001456963	0.001894494
524288	4194304	0	0.002676603	0.003001805
1048576	8388608	0	0.005566595	0.005575649
2097152	16777216	0	0.009910724	0.009629597
4194304	33554432	0	0.019056654	0.017408069
8388608	67108864	0	0.035541328	0.033119654
16777216	134217728	0	0.067253318	0.064271252
33554432	268435456	0	0.132486755	0.128775737
67108864	536870912	0	0.133150120	0.255313222

8. 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:



Si hay ciertas diferencias entre los tiempos de ejecución, por lo que en general, si influye la velacidad de procesamiento.

9. 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 hay ciertos errores para los locales, esto se produce porque hay un desbordamiento de pila, ya que esta esta limitada y por eso cuando metemos valores muy grandes se produce lo que se llama “segmentation fault”, en nuestro caso se produce a partir del valor 524288.

```
[e2estudiante14@atcgrid ejer7]$ sbatch -p ac -n1 SumaVectores_ATC.sh
Submitted batch job 60041
[e2estudiante14@atcgrid ejer7]$ cat slurm-60041.out
Id. usuario del trabajo:
Id. del trabajo:
Nombre  del trabajo especificado por usuario:
Nodo que ejecuta qsub:
Directorio en el que se ha ejecutado qsub:
Cola:
Nodos asignados al trabajo:
Tamanio Vectores:65536 (4 B)
Tiempo:0.000492924      / Tamanio Vectores:65536      / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) /
/ V1[65535]+V2[65535]=V3[65535](13107.100000+0.100000=13107.200000) /
Tamanio Vectores:131072 (4 B)
Tiempo:0.000950800      / Tamanio Vectores:131072     / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.400000)
/ / V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tamanio Vectores:262144 (4 B)
Tiempo:0.001847918      / Tamanio Vectores:262144     / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.800000)
/ / V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
/var/spool/slurmd/job60041/slurm_script: line 23:  9978 Segmentation fault      (core dumped) ./SumaVectoresC $N
/var/spool/slurmd/job60041/slurm_script: line 23:  9980 Segmentation fault      (core dumped) ./SumaVectoresC $N
/var/spool/slurmd/job60041/slurm_script: line 23:  9985 Segmentation fault      (core dumped) ./SumaVectoresC $N
/var/spool/slurmd/job60041/slurm_script: line 23:  9987 Segmentation fault      (core dumped) ./SumaVectoresC $N
/var/spool/slurmd/job60041/slurm_script: line 23:  9989 Segmentation fault      (core dumped) ./SumaVectoresC $N
/var/spool/slurmd/job60041/slurm_script: line 23:  9991 Segmentation fault      (core dumped) ./SumaVectoresC $N
/var/spool/slurmd/job60041/slurm_script: line 23:  9993 Segmentation fault      (core dumped) ./SumaVectoresC $N
/var/spool/slurmd/job60041/slurm_script: line 23:  9995 Segmentation fault      (core dumped) ./SumaVectoresC $N
[e2estudiante14@atcgrid ejer7]$
```

(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 ya que no está limitada con en los locales y tiene un control de desbordamiento ya si se pone un valor muy alto se establece el valor de 33554432 * 4Bytes, así que en ningún caso se va a poder desbordar y dar ese error de violación de segmento.

```
[e2estudiante14@atcgrid ejer7]$ cat slurm-59691.out
Id. usuario del trabajo:
Id. del trabajo:
Nombre  del trabajo especificado por usuario:
Nodo que ejecuta qsub:
Directorio en el que se ha ejecutado qsub:
Cola:
Nodos asignados al trabajo:
Tamanio Vectores:65536 (4 B)
Tiempo:0.000545191      / Tamanio Vectores:65536      / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) /
/ V1[65535]+V2[65535]=V3[65535](13107.100000+0.100000=13107.200000) /
Tamanio Vectores:131072 (4 B)
Tiempo:0.000709896      / Tamanio Vectores:131072     / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.400000)
/ / V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tamanio Vectores:262144 (4 B)
Tiempo:0.001456963      / Tamanio Vectores:262144     / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.800000)
/ / V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tamanio Vectores:524288 (4 B)
Tiempo:0.002676603      / Tamanio Vectores:524288     / V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000)
/ / V1[524287]+V2[524287]=V3[524287](104857.500000+0.100000=104857.600000) /
Tamanio Vectores:1048576 (4 B)
Tiempo:0.005566595      / Tamanio Vectores:1048576    / V1[0]+V2[0]=V3[0](104857.600000+104857.600000=209715.20000
0) / / V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000) /
Tamanio Vectores:2097152 (4 B)
Tiempo:0.009910724      / Tamanio Vectores:2097152    / V1[0]+V2[0]=V3[0](209715.200000+209715.200000=419430.40000
0) / / V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000) /
Tamanio Vectores:4194304 (4 B)
Tiempo:0.019856654      / Tamanio Vectores:4194304    / V1[0]+V2[0]=V3[0](419430.400000+419430.400000=838860.80000
0) / / V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tamanio Vectores:8388608 (4 B)
Tiempo:0.035541328      / Tamanio Vectores:8388608    / V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1677721.6000
00) / / V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000) /
Tamanio Vectores:16777216 (4 B)
Tiempo:0.067253318      / Tamanio Vectores:16777216    / V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.20
0000) / / V1[16777215]+V2[16777215]=V3[16777215](3355443.100000+0.100000=3355443.200000) /
Tamanio Vectores:33554432 (4 B)
Tiempo:0.132486755      / Tamanio Vectores:33554432    / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.40
0000) / / V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.400000) /
Tamanio Vectores:67108864 (4 B)
Tiempo:0.133150120      / Tamanio Vectores:33554432    / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.40
0000) / / V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.400000) /
[e2estudiante14@atcgrid ejer7]$
```

(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:

En los dinámicos tampoco se produces desbordamiento, aunque en esto si hay establecido un limite es mayor que el de los locales por eso no se produce ningún "segmentation fault".

```
[e2estudiante14@atcgrid ejer7]$ sbatch -p ac -n1 SumaVectores_ATC.sh
Submitted batch job 60042
[e2estudiante14@atcgrid ejer7]$ cat slurm-60042.out
Id. usuario del trabajo:
Id. del trabajo:
Nombre  del trabajo especificado por usuario:
Nodo que ejecuta qsub:
Directorio en el que se ha ejecutado qsub:
Cola:
Nodos asignados al trabajo:
Tamanio Vectores:65536 (4 B)
Tiempo:0.000475228      / Tamanio Vectores:65536      / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) /
/ V1[65535]+V2[65535]=V3[65535](13107.100000+0.100000=13107.200000) /
Tamanio Vectores:131072 (4 B)
Tiempo:0.000928291      / Tamanio Vectores:131072     / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.400000)
/ / V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tamanio Vectores:262144 (4 B)
Tiempo:0.001902688      / Tamanio Vectores:262144     / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.800000)
/ / V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tamanio Vectores:524288 (4 B)
Tiempo:0.002962125      / Tamanio Vectores:524288     / V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000)
 / / V1[524287]+V2[524287]=V3[524287](104857.500000+0.100000=104857.600000) /
Tamanio Vectores:1048576 (4 B)
Tiempo:0.005607252      / Tamanio Vectores:1048576    / V1[0]+V2[0]=V3[0](104857.600000+104857.600000=209715.20000
0) / / V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000) /
Tamanio Vectores:2097152 (4 B)
Tiempo:0.010313934      / Tamanio Vectores:2097152    / V1[0]+V2[0]=V3[0](209715.200000+209715.200000=419430.40000
0) / / V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000) /
Tamanio Vectores:4194304 (4 B)
Tiempo:0.019214581      / Tamanio Vectores:4194304    / V1[0]+V2[0]=V3[0](419430.400000+419430.400000=838860.80000
0) / / V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tamanio Vectores:8388608 (4 B)
Tiempo:0.035509705      / Tamanio Vectores:8388608    / V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1677721.6000
00) / / V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000) /
Tamanio Vectores:16777216 (4 B)
Tiempo:0.065445382      / Tamanio Vectores:16777216   / V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.20
0000) / / V1[16777215]+V2[16777215]=V3[16777215](3355443.100000+0.100000=3355443.200000) /
Tamanio Vectores:33554432 (4 B)
Tiempo:0.127933828      / Tamanio Vectores:33554432   / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.40
0000) / / V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.400000) /
Tamanio Vectores:67108864 (4 B)
Tiempo:0.257111078      / Tamanio Vectores:67108864   / V1[0]+V2[0]=V3[0](6710886.400000+6710886.400000=13421772.8
00000) / / V1[67108863]+V2[67108863]=V3[67108863](13421772.700000+0.100000=13421772.800000) /
[e2estudiante14@atcgrid ejer7]$
```

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

RESPUESTA:

Si nos metemos en el código vemos que la variable N es lo siguiente:
unsigned int N = atoi(argv[1]);
Al ser un unsigned int tiene un tamaño de 4 Bytes = 32 Bits entonces su → Máximo N =2^32-1=4294967295

(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:

```
[DavidMartinezDiaz dmartinez01@LAPTOP-H62PMCCC:/mnt/c/User
s/Usuario/AC_Practicas/bp0/ejer10] 2021-03-06 Saturday$gcc
  -O2  SumaVectoresC.c -o SumaVectoresC -lrt
SumaVectoresC.c: In function 'main':
SumaVectoresC.c:68:18: warning: iteration 2147483647 invok
es undefined behavior [-Waggressive-loop-optimizations]
   68 |   for(i=0; i<N; i++){
      |                 ~^~
SumaVectoresC.c:68:3: note: within this loop
   68 |   for(i=0; i<N; i++){
      |   ^~~
SumaVectoresC.c:74:18: warning: iteration 2147483647 invok
es undefined behavior [-Waggressive-loop-optimizations]
   74 |   for(i=0; i<N; i++)
      |                 ~^~
SumaVectoresC.c:74:3: note: within this loop
   74 |   for(i=0; i<N; i++)
      |   ^~~
/tmp/cc7p8YFb.o: in function `main':
SumaVectoresC.c:(.text.startup+0x56): relocation truncated
 to fit: R_X86_64_PC32 against symbol `v2' defined in COMM
ON section in /tmp/cc7p8YFb.o
collect2: error: ld returned 1 exit status
[DavidMartinezDiaz dmartinez01@LAPTOP-H62PMCCC:/mnt/c/User
s/Usuario/AC_Practicas/bp0/ejer10] 2021-03-06 Saturday$
```

El error se encuentra en que aunque se pueda crear el Tam de N que es de 4294967295, cuando se crean los vectores, se quedarian asi v[4294967295], los cuales si los multiplicas por 8 Bytes, quedando un tamaño de 32 GB, el tamaño maximo de los vectores se desborda debido a la falta de espacio en memoria.

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;
}
```