

Arquitectura de Computadores (AC)

Cuaderno de prácticas.

Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos): Salvador Romero Cortés

Grupo de prácticas y profesor de prácticas: A1 Niceto Rafel Luque

Fecha de entrega: 16/03/2021

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:

```

bash /home/salva
[SalvadorRomeroCortés salva@pop-os:~] 2021-03-03 miércoles
$lscpu
Arquitectura:                x86_64
modo(s) de operación de las CPUs: 32-bit, 64-bit
Orden de los bytes:          Little Endian
Address sizes:                43 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:             AuthenticAMD
Familia de CPU:                23
Modelo:                       113
Nombre del modelo:             AMD Ryzen 5 3600 6-Core Processor
Revisión:                      0
Frequency boost:               enabled
CPU MHz:                       2117.103
CPU MHz máx.:                  3600,0000
CPU MHz mín.:                  2200,0000
BogoMIPS:                      7186.20
Virtualización:                AMD-V
Caché L1d:                     192 KiB
Caché L1i:                     192 KiB
Caché L2:                      3 MiB
Caché L3:                      32 MiB
CPU(s) del nodo NUMA 0:        0-11
Vulnerability Itlb multihit:    Not affected
Vulnerability L1tf:             Not affected
Vulnerability Mds:              Not affected
Vulnerability Meltdown:         Not affected
Vulnerability Spec store bypass: Mitigation; Speculative Store Bypass disabled via prctl and seccomp
Vulnerability Spectre v1:       Mitigation; usercopy/swapgs barriers and __user pointer sanitization
Vulnerability Spectre v2:       Mitigation; Full AMD retpoline, IBPB conditional, STIBP conditional, RSB filling
Vulnerability Srbds:            Not affected
Vulnerability Tsx async abort:  Not affected
Indicadores:                    fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt pdpe1gb rdtscp lm constant_tsc rep_good nopl nonstop_tsc cpuid extd_apicid aperfmperf pni pclmulqdq monitor ssse3 fma cx16 sse4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand lahf_lm cmp_legacy svm extapic cr8_legacy abm sse4a misalignsse 3dnowprefetch osvw ibs skinit wdt tce topoext perfctr_core perfctr_nb bpext perfctr_llc mwaitx cpb cat_l3 cdp_l3 hw_pstate sme ssbd mba sev ibpb stibp vmmcall fsgsbase bmi1 avx2 smep bmi2 cqm rdt_a rdseed adx smap clflushopt clwb sha_ni xsaveopt xsavec xgetbv1 xsaves cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local clzero irperf xsaveerptr rdpru wbnoinvd arat npt lbrv svm_lock nrip_save tsc_scale vmcb_clean flushbyasid decodeassists pausefilter pfthreshold avic v_vmsave_vmload vgif umip rdpid overflow_recover succor smca
[SalvadorRomeroCortés salva@pop-os:~] 2021-03-03 miércoles
$

```

Mi PC

```

a1estudiante23@atcgrid:~$ 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:                1170.996
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
h dtb 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 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 epb cat_l3 cdp_l3 invpcid_single intel_ppi
n 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 rtm cqm mpx rdt_a avx512f avx512dq rdseed adx smap clflush
hopt 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
[a1estudiante23@atcgrid ~]$

```

atcgrid4

```

a1estudiante23@atcgrid:~$ 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
                        h dtb 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 tm
                        2 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
a1estudiante23@atcgrid:~$
  
```

atcgrid[1-3]

(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: Mi PC tiene 6 cores físicos (1 socket y 6 núcleos por socket) y 12 lógicos (2 threads por core). Atcgrid4 tiene 32 cores físicos (2 sockets y 16 cores por socket) y 64 lógicos (2 threads por core). Atcgrid[1-3] tiene 12 cores físicos (2 sockets y 6 cores por socket) y 24 lógicos (2 threads por core).

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:

```

bash /home/salva
[SalvadorRomeroCortés salva@pop-os:~/Uni/2Info-C2/AC/bp0/ejer2] 2021-03-03 miércoles
$vim HelloOMP.c
[SalvadorRomeroCortés salva@pop-os:~/Uni/2Info-C2/AC/bp0/ejer2] 2021-03-03 miércoles
$gcc -O2 -fopenmp -o HelloOMP HelloOMP.c
[SalvadorRomeroCortés salva@pop-os:~/Uni/2Info-C2/AC/bp0/ejer2] 2021-03-03 miércoles
$./HelloOMP
(0:!!!Hello world!!!)
(6:!!!Hello world!!!)
(8:!!!Hello world!!!)
(10:!!!Hello world!!!)
(4:!!!Hello world!!!)
(2:!!!Hello world!!!)
(11:!!!Hello world!!!)
(9:!!!Hello world!!!)
(5:!!!Hello world!!!)
(3:!!!Hello world!!!)
(1:!!!Hello world!!!)
(7:!!!Hello world!!!)
[SalvadorRomeroCortés salva@pop-os:~/Uni/2Info-C2/AC/bp0/ejer2] 2021-03-03 miércoles
$

```

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

Tiene 12 “Hello World” porque mi PC tiene 12 hilos (6 cores con hyperthreading).

RESPUESTA:

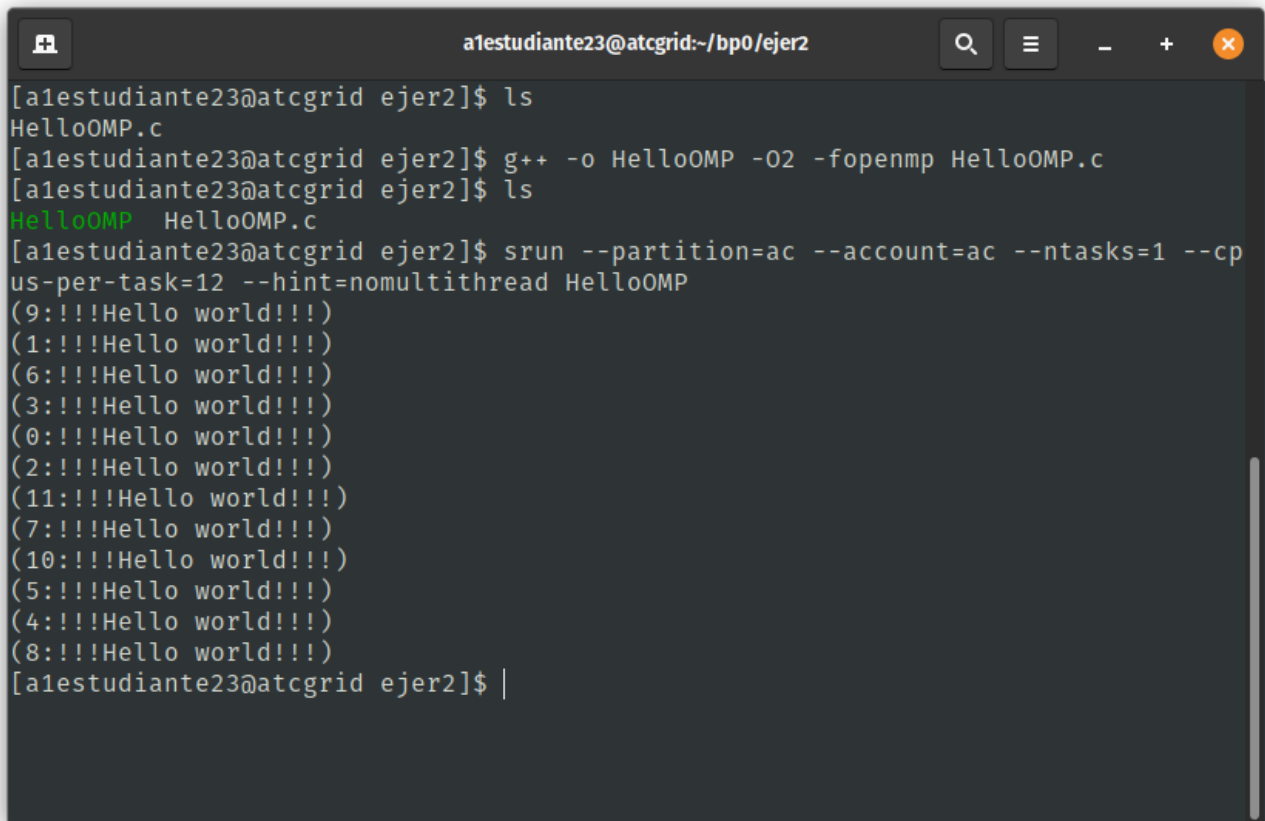
3. Copiar el ejecutable de `HelloOMP.c` que ha generado anteriormente y que se encuentra en el directorio `ejer2` del PC al directorio `ejer2` de su home en el *front-end* de `atcgrid`. Ejecutar este código en un nodo de cómputo de `atcgrid` (de 1 a 3) a través de cola `ac` del gestor de colas utilizando 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:

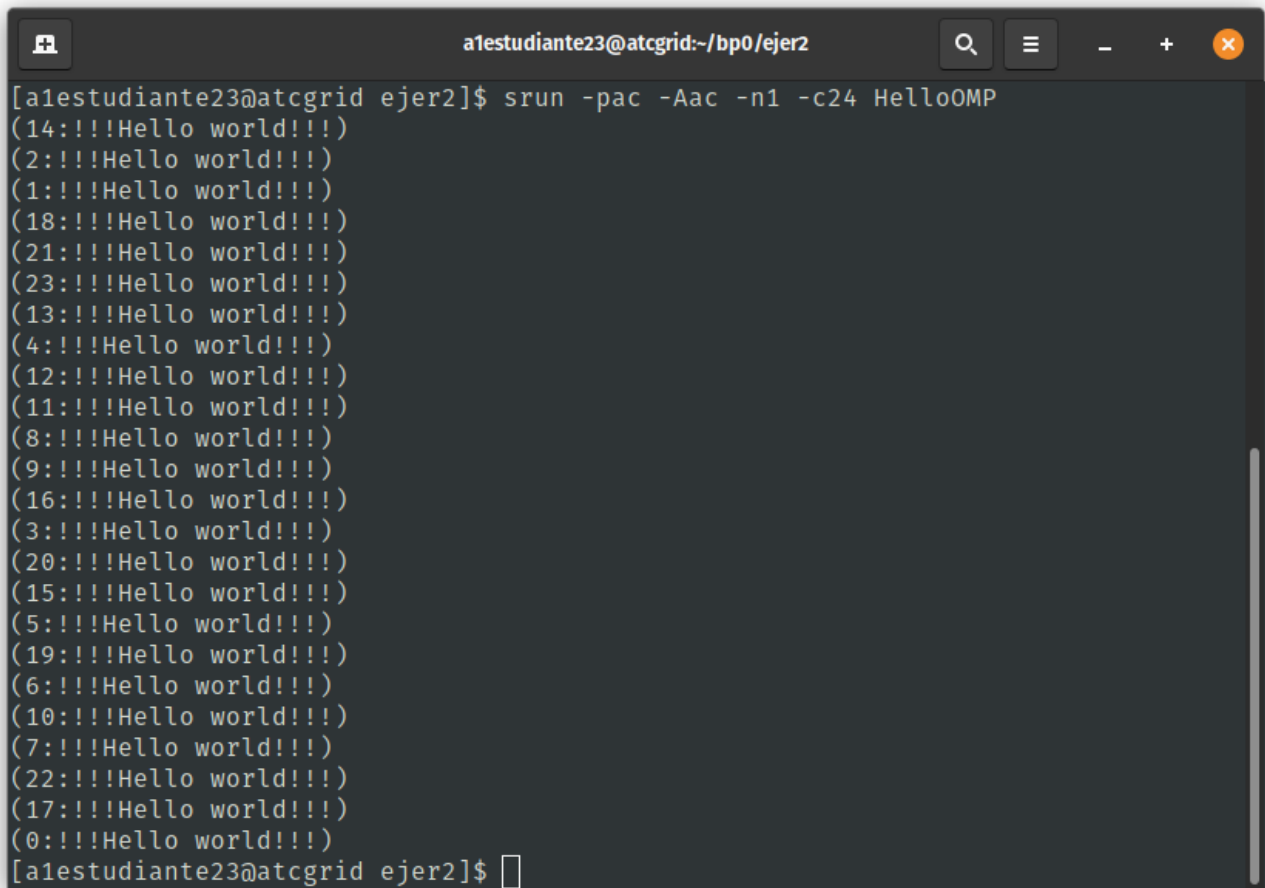


```
a1estudiante23@atcgrid:~/bp0/ejer2
[a1estudiante23@atcgrid ejer2]$ ls
HelloOMP.c
[a1estudiante23@atcgrid ejer2]$ g++ -o HelloOMP -O2 -fopenmp HelloOMP.c
[a1estudiante23@atcgrid ejer2]$ ls
HelloOMP  HelloOMP.c
[a1estudiante23@atcgrid ejer2]$ srun --partition=ac --account=ac --ntasks=1 --cp
us-per-task=12 --hint=nomultithread HelloOMP
(9:!!!Hello world!!!)
(1:!!!Hello world!!!)
(6:!!!Hello world!!!)
(3:!!!Hello world!!!)
(0:!!!Hello world!!!)
(2:!!!Hello world!!!)
(11:!!!Hello world!!!)
(7:!!!Hello world!!!)
(10:!!!Hello world!!!)
(5:!!!Hello world!!!)
(4:!!!Hello world!!!)
(8:!!!Hello world!!!)
[a1estudiante23@atcgrid ejer2]$ |
```

(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:

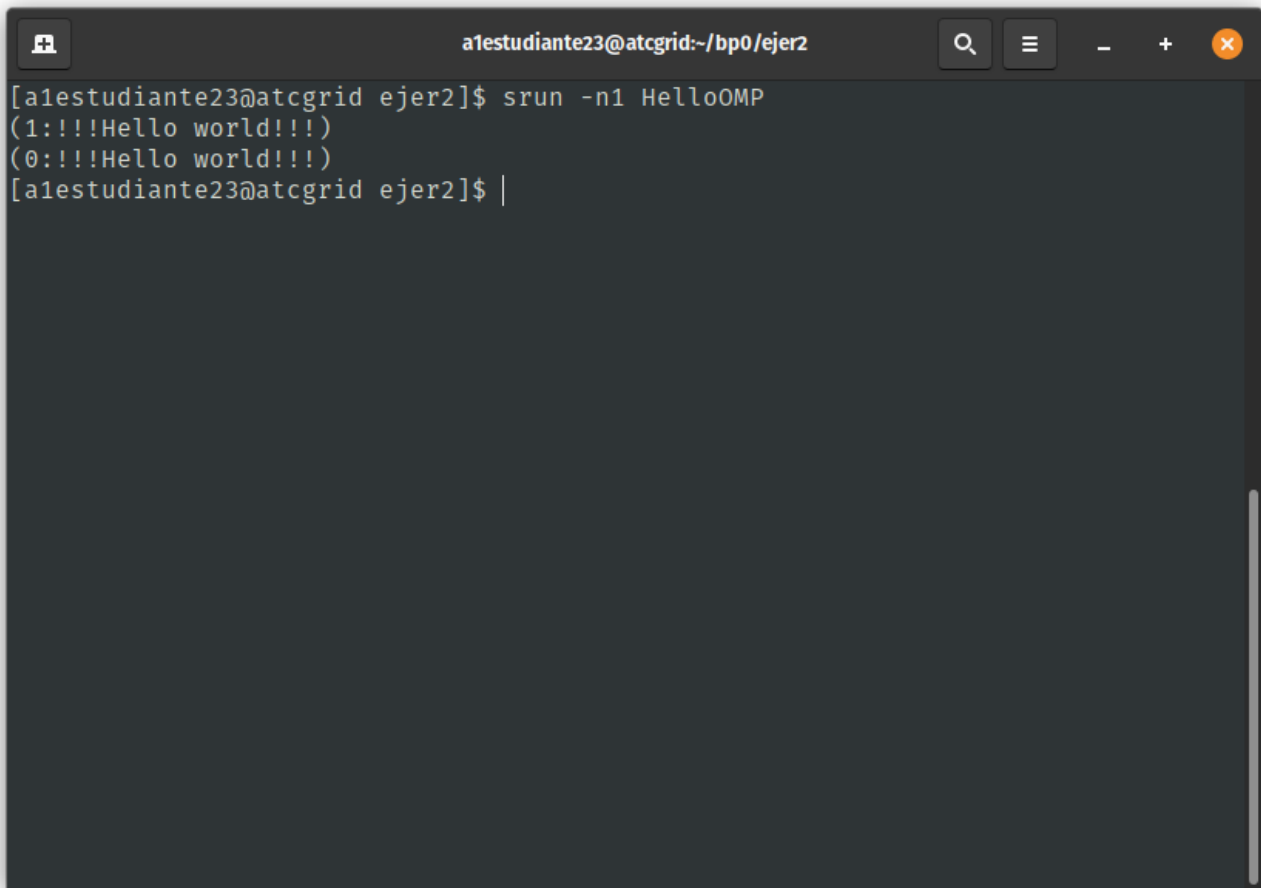


```
a1estudiante23@atcgrid:~/bp0/ejer2
[a1estudiante23@atcgrid ejer2]$ srun -pac -Aac -n1 -c24 HelloOMP
(14:!!!Hello world!!!)
(2:!!!Hello world!!!)
(1:!!!Hello world!!!)
(18:!!!Hello world!!!)
(21:!!!Hello world!!!)
(23:!!!Hello world!!!)
(13:!!!Hello world!!!)
(4:!!!Hello world!!!)
(12:!!!Hello world!!!)
(11:!!!Hello world!!!)
(8:!!!Hello world!!!)
(9:!!!Hello world!!!)
(16:!!!Hello world!!!)
(3:!!!Hello world!!!)
(20:!!!Hello world!!!)
(15:!!!Hello world!!!)
(5:!!!Hello world!!!)
(19:!!!Hello world!!!)
(6:!!!Hello world!!!)
(10:!!!Hello world!!!)
(7:!!!Hello world!!!)
(22:!!!Hello world!!!)
(17:!!!Hello world!!!)
(0:!!!Hello world!!!)
[a1estudiante23@atcgrid ejer2]$
```

(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:

A terminal window titled 'a1estudiante23@atcgrid:~/bp0/ejer2'. The prompt is '[a1estudiante23@atcgrid ejer2]\$'. The user has entered 'srun -n1 HelloOMP'. The output shows two lines: '(1:!!!Hello world!!!)' and '(0:!!!Hello world!!!)'. The prompt is now '[a1estudiante23@atcgrid ejer2]\$ |' with a cursor.

```
a1estudiante23@atcgrid:~/bp0/ejer2
[a1estudiante23@atcgrid ejer2]$ srun -n1 HelloOMP
(1:!!!Hello world!!!)
(0:!!!Hello world!!!)
[a1estudiante23@atcgrid ejer2]$ |
```

Usa la partición 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 -Aac -pac4 --cpus-per-task=32 --hint=nomultithread HelloOMP
```



```

a1estudiante23@atcgrid:~/bp0/ejer2
[a1estudiante23@atcgrid ejer2]$ srun -pac4 -Aac -n1 -c32 HelloOMP
(11:!!!Hello world!!!)
(14:!!!Hello world!!!)
(18:!!!Hello world!!!)
(26:!!!Hello world!!!)
(24:!!!Hello world!!!)
(27:!!!Hello world!!!)
(15:!!!Hello world!!!)
(30:!!!Hello world!!!)
(5:!!!Hello world!!!)
(28:!!!Hello world!!!)
(29:!!!Hello world!!!)
(31:!!!Hello world!!!)
(21:!!!Hello world!!!)
(0:!!!Hello world!!!)
(7:!!!Hello world!!!)
(9:!!!Hello world!!!)
(22:!!!Hello world!!!)
(4:!!!Hello world!!!)
(19:!!!Hello world!!!)
(16:!!!Hello world!!!)
(13:!!!Hello world!!!)
(10:!!!Hello world!!!)
(2:!!!Hello world!!!)
(6:!!!Hello world!!!)
(25:!!!Hello world!!!)
(23:!!!Hello world!!!)
(8:!!!Hello world!!!)
(3:!!!Hello world!!!)
(12:!!!Hello world!!!)
(17:!!!Hello world!!!)
(20:!!!Hello world!!!)
(1:!!!Hello world!!!)
[a1estudiante23@atcgrid ejer2]$ |

```

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:

```

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

int main(void){
#pragma omp parallel
    printf("[[%d]] -> Hello\n",omp_get_thread_num());
    printf("<- World [[%d]]\n",omp_get_thread_num());

    return(0);
}
~
~
~
~
~
~
~
~
HelloOMP2.c 7,50-57 Todo
"HelloOMP2.c" 10L, 191C escritos
    
```

Código nuevo

```

[SalvadorRomeroCortés salva@pop-os:~/Uni/2Info-C2/AC/bp0/ejer4] 2021-03-08 lunes
$gcc -O2 HelloOMP2.c -o HelloOMP2 -fopenmp
[SalvadorRomeroCortés salva@pop-os:~/Uni/2Info-C2/AC/bp0/ejer4] 2021-03-08 lunes
$./HelloOMP2
[[4]] -> Hello
[[8]] -> Hello
[[10]] -> Hello
[[5]] -> Hello
[[3]] -> Hello
[[9]] -> Hello
[[11]] -> Hello
[[1]] -> Hello
[[7]] -> Hello
[[2]] -> Hello
[[0]] -> Hello
[[6]] -> Hello
<- World [[0]]
[SalvadorRomeroCortés salva@pop-os:~/Uni/2Info-C2/AC/bp0/ejer4] 2021-03-08 lunes
$
    
```

Compilación y ejecución en mi pc

```
a1estudiante23@atcgrid:~/bp0/ejer4
[a1estudiante23@atcgrid ejer4]$ srun ./script_helloomp.sh
Id. usuario del trabajo: a1estudiante23
Id. del trabajo: 61904
Nombre del trabajo especificado por usuario: script_helloomp.sh
Directorio de trabajo (en el que se ejecuta el script): /home/a1estudiante23/bp0/ejer4
Cola: ac
Nodo que ejecuta este trabajo: atcgrid.ugr.es
Nº de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgrid1
CPUs por nodo: 2

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

[[1]] -> Hello
[[0]] -> Hello
<- World [[0]]

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

- Para 12 threads:
[[1]] -> Hello
[[2]] -> Hello
[[0]] -> Hello
[[3]] -> Hello
[[4]] -> Hello
[[5]] -> Hello
[[6]] -> Hello
[[7]] -> Hello
[[8]] -> Hello
[[9]] -> Hello
[[10]] -> Hello
[[11]] -> Hello
<- World [[0]]

- Para 6 threads:
[[2]] -> Hello
[[1]] -> Hello
[[3]] -> Hello
[[4]] -> Hello
[[0]] -> Hello
[[5]] -> Hello
<- World [[0]]

- Para 3 threads:
[[2]] -> Hello
[[1]] -> Hello
[[0]] -> Hello
<- World [[0]]

- Para 1 threads:
[[0]] -> Hello
<- World [[0]]
[a1estudiante23@atcgrid ejer4]$
```

Ejecución del script con srun

Depto. Arquitectura y Tecnología de Computadores

```

a1estudiante23@atcgrid:~/bp0/ejer4
[a1estudiante23@atcgrid ejer4]$ sbatch -pac -n1 -c12 --hint=nomultithread script_helloomp.sh
Submitted batch job 61883
[a1estudiante23@atcgrid ejer4]$ ls
HelloOMP2 HelloOMP2.c script_helloomp.sh slurm-61883.out
[a1estudiante23@atcgrid ejer4]$ cat slurm-61883.out
Id. usuario del trabajo: a1estudiante23
Id. del trabajo: 61883
Nombre del trabajo especificado por usuario: helloOMP
Directorio de trabajo (en el que se ejecuta el script): /home/a1estudiante23/bp0/ejer4
Cola: ac
Nodo que ejecuta este trabajo: atcgrid.ugr.es
Nº de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgrid1
CPUs por nodo: 24

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

[[2]] -> Hello
[[8]] -> Hello
[[4]] -> Hello
[[0]] -> Hello
[[6]] -> Hello
[[10]] -> Hello
[[3]] -> Hello
[[5]] -> Hello
[[11]] -> Hello
[[1]] -> Hello
[[7]] -> Hello
[[9]] -> Hello
<- World [[0]]

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

- Para 12 threads:
[[4]] -> Hello
[[7]] -> Hello
[[1]] -> Hello
[[0]] -> Hello
[[2]] -> Hello
[[3]] -> Hello
[[5]] -> Hello
[[8]] -> Hello
[[6]] -> Hello
[[10]] -> Hello
[[11]] -> Hello
[[9]] -> Hello
<- World [[0]]

- Para 6 threads:
[[1]] -> Hello
[[5]] -> Hello
[[2]] -> Hello
[[4]] -> Hello
[[3]] -> Hello
[[0]] -> Hello
<- World [[0]]

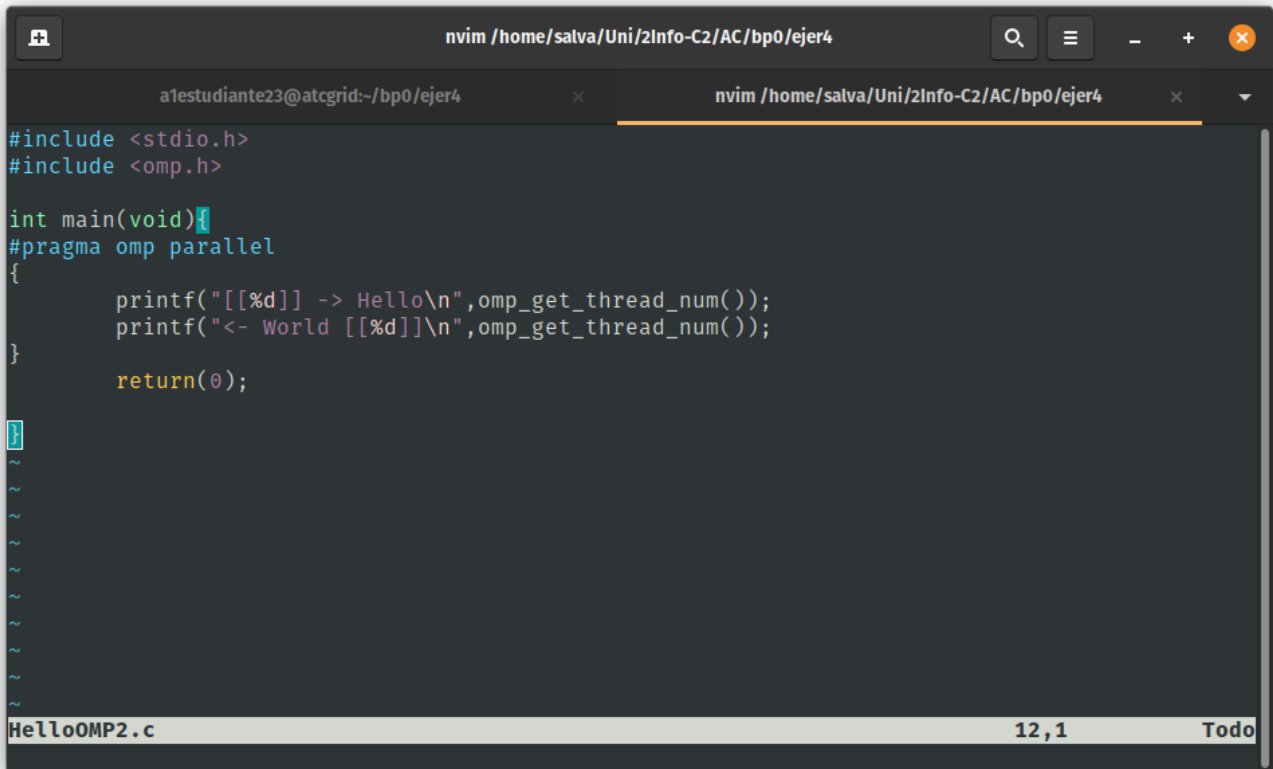
- Para 3 threads:
[[1]] -> Hello
[[0]] -> Hello
[[2]] -> Hello
<- World [[0]]

- Para 1 threads:
[[0]] -> Hello
<- World [[0]]

```

Envío a la cola y ejecución con instrucciones del apartado (sbatch)

Vemos que el segundo printf sólo lo ejecuta el primer núcleo. Esto es porque tenemos que indicar a la directiva OMP que incluya los dos printf. Eso lo hacemos encerrándolo entre llaves.



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

int main(void){
#pragma omp parallel
{
    printf("[[%d]] -> Hello\n",omp_get_thread_num());
    printf("<- World [[%d]]\n",omp_get_thread_num());
}

    return(0);
}
```

Código entre llaves

Y si ejecutamos esta versión:

```

a1estudiante23@atcgrid:~/bp0/ejer4

[a1estudiante23@atcgrid ejer4]$ sbatch -pac -n1 -c12 --hint=nomultithread script_helloomp.sh
Submitted batch job 63095
[a1estudiante23@atcgrid ejer4]$ cat slurm-63095.out
Id. usuario del trabajo: a1estudiante23
Id. del trabajo: 63095
Nombre del trabajo especificado por usuario: helloOMP
Directorio de trabajo (en el que se ejecuta el script): /home/a1estudiante23/bp0/ejer4
Cola: ac
Nodo que ejecuta este trabajo: atcgrid.ugr.es
Nº de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgrid1
CPUs por nodo: 24

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

[[2]] -> Hello
[[11]] -> Hello
<- World [[2]]
<- World [[11]]
[[7]] -> Hello
[[9]] -> Hello
<- World [[7]]
<- World [[9]]
[[10]] -> Hello
<- World [[10]]
[[0]] -> Hello
[[8]] -> Hello
<- World [[0]]
<- World [[8]]
[[6]] -> Hello
<- World [[6]]
[[3]] -> Hello
<- World [[3]]
[[4]] -> Hello
<- World [[4]]
[[5]] -> Hello
<- World [[5]]
[[1]] -> Hello
<- World [[1]]

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

- Para 12 threads:
[[11]] -> Hello
<- World [[11]]
[[2]] -> Hello
<- World [[2]]
[[1]] -> Hello
<- World [[1]]
[[6]] -> Hello
<- World [[6]]
[[3]] -> Hello
<- World [[3]]
[[7]] -> Hello
<- World [[7]]
[[5]] -> Hello
<- World [[5]]
[[10]] -> Hello
<- World [[10]]
[[9]] -> Hello
<- World [[9]]
[[0]] -> Hello
<- World [[0]]
[[8]] -> Hello
<- World [[8]]
[[4]] -> Hello
<- World [[4]]

- Para 6 threads:
[[0]] -> Hello
<- World [[0]]
[[2]] -> Hello
<- World [[2]]
[[5]] -> Hello
<- World [[5]]
[[3]] -> Hello
<- World [[3]]
[[4]] -> Hello
<- World [[4]]
[[1]] -> Hello
<- World [[1]]

- Para 3 threads:
[[1]] -> Hello
<- World [[1]]
[[0]] -> Hello
<- World [[0]]
[[2]] -> Hello
<- World [[2]]

- Para 1 threads:
[[0]] -> Hello
<- World [[0]]
[a1estudiante23@atcgrid ejer4]$ |

```

Ahora se ejecuta correctamente.

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

RESPUESTA:

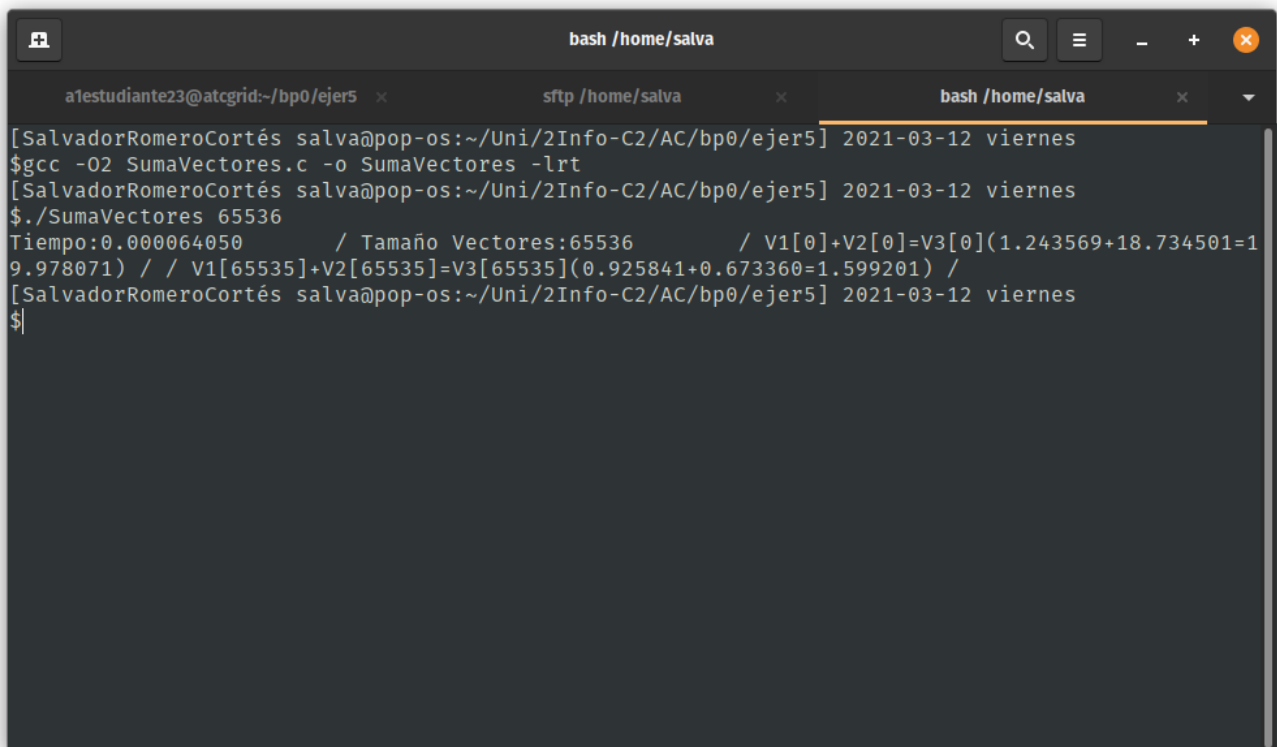
Usa uno de los nodos de cómputo del atcgrid1. Lo sé porque en el script, en la configuración de SBATCH aparece `-partition=ac:` indicando que la partición usada es la ac. Dentro de esta partición podemos ver (en la captura de pantalla) en la línea de nodos asignados al trabajo que es el atcgrid1 el que realiza el trabajo.

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:



```

bash /home/salva
a1estudiante23@atcgrid:~/bp0/ejer5 x sftp /home/salva x bash /home/salva x
[SalvadorRomeroCortés salva@pop-os:~/Uni/2Info-C2/AC/bp0/ejer5] 2021-03-12 viernes
$gcc -O2 SumaVectores.c -o SumaVectores -lrt
[SalvadorRomeroCortés salva@pop-os:~/Uni/2Info-C2/AC/bp0/ejer5] 2021-03-12 viernes
$./SumaVectores 65536
Tiempo:0.000064050 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](1.243569+18.734501=1
9.978071) / / V1[65535]+V2[65535]=V3[65535](0.925841+0.673360=1.599201) /
[SalvadorRomeroCortés salva@pop-os:~/Uni/2Info-C2/AC/bp0/ejer5] 2021-03-12 viernes
$|

```

Compilación y ejecución en mi pc

6. En el código del Listado 1 se utiliza la función `clock_gettime()` para obtener el tiempo de ejecución del trozo de código que calcula la suma de vectores. El código se imprime la variable `ncgt`,

(a) ¿Qué contiene esta variable?

RESPUESTA:

La variable `ncgt` es un `double` que almacena el tiempo que ha tardado en ejecutarse la suma de vectores (la diferencia entre el tiempo final y el inicial).

(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()` devuelve una estructura del tipo `timespec` (por referencia). Esta estructura está compuesta por `tv_sec` de tipo `time_t` y por `tv_nsec` de tipo `long`. El primero representa los segundos enteros y el segundo, 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?

Devuelve el tiempo en el que se llama a la función. Esto es, un número, que representa los segundos que han pasado desde el 1 de enero de 1970. Se conoce como tiempo Unix.

RESPUESTA:

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 . Resultados ATCGRID

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0,000470351	0,000566736	0.000361375
131072	1048576	0,000550487	0,000694664	0.000883653
262144	2097152	0,001830576	0,001353386	0.001881336
524288	4194304	Segmentation fault	0,002677504	0.002729792
1048576	8388608	Segmentation fault	0,004969129	0.004933475
2097152	16777216	Segmentation fault	0,009025316	0.008827536
4194304	33554432	Segmentation fault	0,017378560	0.016668563
8388608	67108864	Segmentation fault	0,034468847	0.032550488
16777216	134217728	Segmentation fault	0,066656231	0.064483873
33554432	268435456	Segmentation fault	0,130933260	0.126980703
67108864	536870912	Segmentation fault	0,131841598	0.245421535

RESULTADOS PC

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000195528	0.000102823	0.000110297
131072	1048576	0.000321425	0.000286449	0.000421784
262144	2097152	0.000622842	0.000458172	0.000540166
524288	4194304	Segmentation fault	0.001123835	0.001307160
1048576	8388608	Segmentation fault	0.002619039	0.002578332
2097152	16777216	Segmentation fault	0.005486005	0.005210996
4194304	33554432	Segmentation fault	0.010761692	0.008899047
8388608	67108864	Segmentation fault	0.024422571	0.019097119
16777216	134217728	Segmentation fault	0.044430703	0.042916344
33554432	268435456	Segmentation fault	0.087699550	0.075320815
67108864	536870912	Segmentation fault	0.093542466	0.150668802

```

a1estudiante23@atcgrid:~/bp0/ejer7
a1estudiante23@atcgrid:~/bp0/ejer7
[alestudiante23@atcgrid:~/bp0/ejer7]$ cat slurm-67115.out
Id. usuario del trabajo: alestudiante23
Id. del trabajo: 67115
Nombre del trabajo especificado por usuario: SumaLocal
Directorio de trabajo (en el que se ejecuta el script): /home/alestudiante23/bp0/ejer7
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: 2
VECTORES LOCALES
Tiempo(seg.):0.000470351 / Tamaño Vectores:65536 / V1[0]-V2[0]=V3[0](0.541424+0.357051-0.898474) / V1[65535]-V2[65535]=V3[65535](0.069264+1.327851-1.397115) /
Tiempo(seg.):0.000550487 / Tamaño Vectores:131072 / V1[0]-V2[0]=V3[0](0.541424+0.357051-0.898474) / V1[131071]-V2[131071]=V3[131071](1.285722+158.427513-159.713235) /
Tiempo(seg.):0.001830576 / Tamaño Vectores:262144 / V1[0]-V2[0]=V3[0](0.541424+0.357051-0.898474) / V1[262143]-V2[262143]=V3[262143](1.095415+33.210798+3.306213) /
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
VECTORES GLOBALES
Tiempo:0.00056736 / Tamaño Vectores:65536 / V1[0]-V2[0]=V3[0](0.644198+0.437666-1.081864) / V1[65535]-V2[65535]=V3[65535](1.213433+1.099811+2.313243) /
Tiempo:0.000694664 / Tamaño Vectores:131072 / V1[0]-V2[0]=V3[0](0.644198+0.437666-1.081864) / V1[131071]-V2[131071]=V3[131071](0.455610-1.448392+1.984002) /
Tiempo:0.001353386 / Tamaño Vectores:262144 / V1[0]-V2[0]=V3[0](0.644198+0.437666-1.081864) / V1[262143]-V2[262143]=V3[262143](3.940103-0.337368+4.277471) /
Tiempo:0.002677504 / Tamaño Vectores:524288 / V1[0]-V2[0]=V3[0](0.644198+0.437666-1.081864) / V1[524287]-V2[524287]=V3[524287](0.761456-0.712901-1.474358) /
Tiempo:0.004969129 / Tamaño Vectores:1048576 / V1[0]-V2[0]=V3[0](0.644198+0.437666-1.081864) / V1[1048575]-V2[1048575]=V3[1048575](0.973836-0.589758-1.562794) /
Tiempo:0.009025316 / Tamaño Vectores:2097152 / V1[0]-V2[0]=V3[0](0.644198+0.437666-1.081864) / V1[2097151]-V2[2097151]=V3[2097151](0.900230+0.242839+1.434069) /
Tiempo:0.017378560 / Tamaño Vectores:4194304 / V1[0]-V2[0]=V3[0](0.383180+2.000943+2.384043) / V1[4194303]-V2[4194303]=V3[4194303](0.349641+0.950134+1.299775) /
Tiempo:0.034468847 / Tamaño Vectores:8388608 / V1[0]-V2[0]=V3[0](0.383180+2.000943+2.384043) / V1[8388607]-V2[8388607]=V3[8388607](4.585693+0.512539+5.098232) /
Tiempo:0.066656231 / Tamaño Vectores:16777216 / V1[0]-V2[0]=V3[0](1.321152+1.050347+2.371500) / V1[16777215]-V2[16777215]=V3[16777215](0.880689+0.908405+1.789094) /
Tiempo:0.130933260 / Tamaño Vectores:33554432 / V1[0]-V2[0]=V3[0](1.279303+0.329551+1.608854) / V1[33554431]-V2[33554431]=V3[33554431](0.958630+0.645257+1.603887) /
Tiempo:0.131841598 / Tamaño Vectores:33554432 / V1[0]-V2[0]=V3[0](0.567897+0.980098+1.547995) / V1[33554431]-V2[33554431]=V3[33554431](1.524753+0.742637+2.267389) /
VECTORES DINÁMICOS
Tiempo:0.000361375 / Tamaño Vectores:65536 / V1[0]-V2[0]=V3[0](0.341085+1.951054+2.292139) / V1[65535]-V2[65535]=V3[65535](1.108545+1.230100+2.338645) /
Tiempo:0.000883653 / Tamaño Vectores:131072 / V1[0]-V2[0]=V3[0](0.341085+1.951054+2.292139) / V1[131071]-V2[131071]=V3[131071](0.160284+0.420613+0.580897) /
Tiempo:0.001881336 / Tamaño Vectores:262144 / V1[0]-V2[0]=V3[0](0.341085+1.951054+2.292139) / V1[262143]-V2[262143]=V3[262143](2.559816+1.222443+3.782259) /
Tiempo:0.002729792 / Tamaño Vectores:524288 / V1[0]-V2[0]=V3[0](0.341085+1.951054+2.292139) / V1[524287]-V2[524287]=V3[524287](1.167549+1.499672+2.667221) /
Tiempo:0.004933475 / Tamaño Vectores:1048576 / V1[0]-V2[0]=V3[0](0.341085+1.951054+2.292139) / V1[1048575]-V2[1048575]=V3[1048575](0.995989+0.760738+1.756727) /
Tiempo:0.008027536 / Tamaño Vectores:2097152 / V1[0]-V2[0]=V3[0](0.341085+1.951054+2.292139) / V1[2097151]-V2[2097151]=V3[2097151](1.645681+0.559865+2.205466) /
Tiempo:0.016685653 / Tamaño Vectores:4194304 / V1[0]-V2[0]=V3[0](0.341085+1.951054+2.292139) / V1[4194303]-V2[4194303]=V3[4194303](1.328735+1.955231+3.323966) /
Tiempo:0.032550488 / Tamaño Vectores:8388608 / V1[0]-V2[0]=V3[0](2.830196+0.924213+3.754408) / V1[8388607]-V2[8388607]=V3[8388607](1.897647+0.655207+2.552854) /
Tiempo:0.064483873 / Tamaño Vectores:16777216 / V1[0]-V2[0]=V3[0](2.830196+0.924213+3.754408) / V1[16777215]-V2[16777215]=V3[16777215](16.906297+0.131709+17.038006) /
Tiempo:0.120980703 / Tamaño Vectores:33554432 / V1[0]-V2[0]=V3[0](3.520279+0.393090+3.913369) / V1[33554431]-V2[33554431]=V3[33554431](0.319515+6.708789+7.028303) /
Tiempo:0.245421535 / Tamaño Vectores:67108864 / V1[0]-V2[0]=V3[0](0.738534+2.483869+3.222403) / V1[67108863]-V2[67108863]=V3[67108863](1.756286+11.760470+13.516756) /
[alestudiante23@atcgrid:~/bp0/ejer7]$

```

Ejecución en atcgrid

```

bash /home/salva
a1estudiante23@atcgid:~/bp0/ejer7
[SalvadorRomeroCortés salva@pop-os:~/Uni/2Info-C2/AC/bp0/ejer7] 2021-03-12 viernes
$.script_sumav.sh
Id. usuario del trabajo:
Id. del trabajo:
Nombre del trabajo especificado por usuario:
Directorio de trabajo (en el que se ejecuta el script):
Cola:
Nodo que ejecuta este trabajo:
No. de nodos asignados al trabajo:
Nodos asignados al trabajo:
CPUs por nodo:
VECTORES LOCALES
Tiempo(seg.):0.000195528 / Tamaño Vectores:65536 / V1[0].V2[0]=V3[0](0.762031+0.864346+1.626377) // V1[65535].V2[65535]=V3[65535](0.606630+0.766826+1.373456) /
Tiempo(seg.):0.000321425 / Tamaño Vectores:131072 / V1[0].V2[0]=V3[0](0.762031+0.864346+1.626377) // V1[131071].V2[131071]=V3[131071](0.758182+0.980458+1.738640) /
Tiempo(seg.):0.000622842 / Tamaño Vectores:262144 / V1[0].V2[0]=V3[0](0.762031+0.864346+1.626377) // V1[262143].V2[262143]=V3[262143](5.282924+0.044714+5.327639) /
./script_sumav.sh: línea 25: 12800 Violación de segmento ('core' generado) ./SumaVectores $P
./script_sumav.sh: línea 25: 12802 Violación de segmento ('core' generado) ./SumaVectores $P
./script_sumav.sh: línea 25: 12806 Violación de segmento ('core' generado) ./SumaVectores $P
./script_sumav.sh: línea 25: 12808 Violación de segmento ('core' generado) ./SumaVectores $P
./script_sumav.sh: línea 25: 12810 Violación de segmento ('core' generado) ./SumaVectores $P
./script_sumav.sh: línea 25: 12812 Violación de segmento ('core' generado) ./SumaVectores $P
./script_sumav.sh: línea 25: 12814 Violación de segmento ('core' generado) ./SumaVectores $P
VECTORES GLOBALES
Tiempo(seg.):0.000102823 / Tamaño Vectores:65536 / V1[0].V2[0]=V3[0](0.762031+0.864346+1.626377) // V1[65535].V2[65535]=V3[65535](0.606630+0.766826+1.373456) /
Tiempo(seg.):0.000286449 / Tamaño Vectores:131072 / V1[0].V2[0]=V3[0](0.762031+0.864346+1.626377) // V1[131071].V2[131071]=V3[131071](0.758182+0.980458+1.738640) /
Tiempo(seg.):0.000458172 / Tamaño Vectores:262144 / V1[0].V2[0]=V3[0](0.762031+0.864346+1.626377) // V1[262143].V2[262143]=V3[262143](5.282924+0.044714+5.327639) /
Tiempo(seg.):0.001123835 / Tamaño Vectores:524288 / V1[0].V2[0]=V3[0](0.762031+0.864346+1.626377) // V1[524287].V2[524287]=V3[524287](2.222507+0.847155+3.069662) /
Tiempo(seg.):0.002610039 / Tamaño Vectores:1048576 / V1[0].V2[0]=V3[0](0.762031+0.864346+1.626377) // V1[1048575].V2[1048575]=V3[1048575](1.792122+322.433269+324.223391) /
Tiempo(seg.):0.005486005 / Tamaño Vectores:2097152 / V1[0].V2[0]=V3[0](0.762031+0.864346+1.626377) // V1[2097151].V2[2097151]=V3[2097151](0.450660+1.606172+2.264840) /
Tiempo(seg.):0.010761692 / Tamaño Vectores:4194304 / V1[0].V2[0]=V3[0](0.817314+1.869433+2.686748) // V1[4194303].V2[4194303]=V3[4194303](1.915967+0.445630+2.361507) /
Tiempo(seg.):0.024422571 / Tamaño Vectores:8388608 / V1[0].V2[0]=V3[0](0.817314+1.869433+2.686748) // V1[8388607].V2[8388607]=V3[8388607](0.626106+2.425228+3.051334) /
Tiempo(seg.):0.044430703 / Tamaño Vectores:16777216 / V1[0].V2[0]=V3[0](0.817314+1.869433+2.686748) // V1[16777215].V2[16777215]=V3[16777215](0.779731+2.074273+2.854004) /
Tiempo(seg.):0.087699550 / Tamaño Vectores:33554432 / V1[0].V2[0]=V3[0](0.817314+1.869433+2.686748) // V1[33554431].V2[33554431]=V3[33554431](0.725081+0.765033+1.490114) /
Tiempo(seg.):0.093542466 / Tamaño Vectores:33554432 / V1[0].V2[0]=V3[0](0.108600+2.382112+2.490712) // V1[33554431].V2[33554431]=V3[33554431](8.872370+5.890860+14.770438) /
VECTORES DINÁMICOS
Tiempo(seg.):0.00010297 / Tamaño Vectores:65536 / V1[0].V2[0]=V3[0](0.596015+1.608840+2.204855) // V1[65535].V2[65535]=V3[65535](0.869393+0.839753+1.709146) /
Tiempo(seg.):0.000421784 / Tamaño Vectores:131072 / V1[0].V2[0]=V3[0](0.596015+1.608840+2.204855) // V1[131071].V2[131071]=V3[131071](0.636242+2.782635+3.418877) /
Tiempo(seg.):0.000549166 / Tamaño Vectores:262144 / V1[0].V2[0]=V3[0](0.596015+1.608840+2.204855) // V1[262143].V2[262143]=V3[262143](2.046249+1.033253+3.079502) /
Tiempo(seg.):0.001307160 / Tamaño Vectores:524288 / V1[0].V2[0]=V3[0](0.596015+1.608840+2.204855) // V1[524287].V2[524287]=V3[524287](1.320972+0.741879+2.062850) /
Tiempo(seg.):0.002578332 / Tamaño Vectores:1048576 / V1[0].V2[0]=V3[0](0.596015+1.608840+2.204855) // V1[1048575].V2[1048575]=V3[1048575](0.240496+1.082076+1.322572) /
Tiempo(seg.):0.005210996 / Tamaño Vectores:2097152 / V1[0].V2[0]=V3[0](0.596015+1.608840+2.204855) // V1[2097151].V2[2097151]=V3[2097151](1.012876+4422.965120+4423.978005) /
Tiempo(seg.):0.008890967 / Tamaño Vectores:4194304 / V1[0].V2[0]=V3[0](0.596015+1.608840+2.204855) // V1[4194303].V2[4194303]=V3[4194303](0.450653+0.286695+0.737248) /
Tiempo(seg.):0.019097119 / Tamaño Vectores:8388608 / V1[0].V2[0]=V3[0](0.596015+1.608840+2.204855) // V1[8388607].V2[8388607]=V3[8388607](2.071171+1.208612+3.369782) /
Tiempo(seg.):0.042916344 / Tamaño Vectores:16777216 / V1[0].V2[0]=V3[0](12.686793+0.829805+13.516598) // V1[16777215].V2[16777215]=V3[16777215](0.647859+0.717813+1.365672) /
Tiempo(seg.):0.075328815 / Tamaño Vectores:33554432 / V1[0].V2[0]=V3[0](12.686793+0.829805+13.516598) // V1[33554431].V2[33554431]=V3[33554431](1.434192+0.182707+1.616899) /
Tiempo(seg.):0.150668802 / Tamaño Vectores:67108864 / V1[0].V2[0]=V3[0](0.206200+2.433874+2.640075) // V1[67108863].V2[67108863]=V3[67108863](0.951141+0.237273+1.188414) /
[SalvadorRomeroCortés salva@pop-os:~/Uni/2Info-C2/AC/bp0/ejer7] 2021-03-12 viernes
$

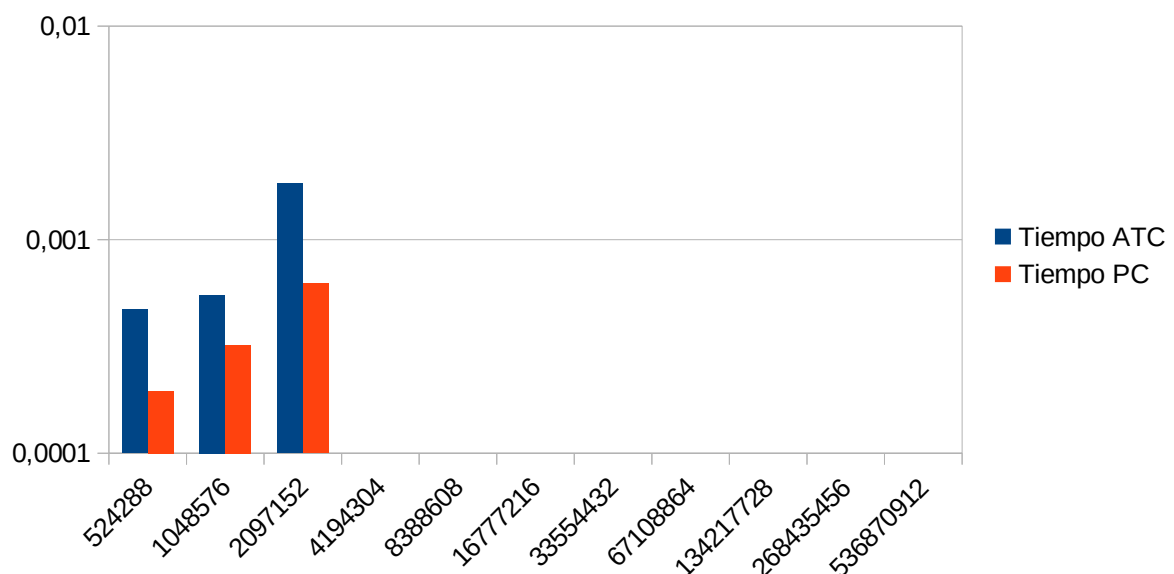
```

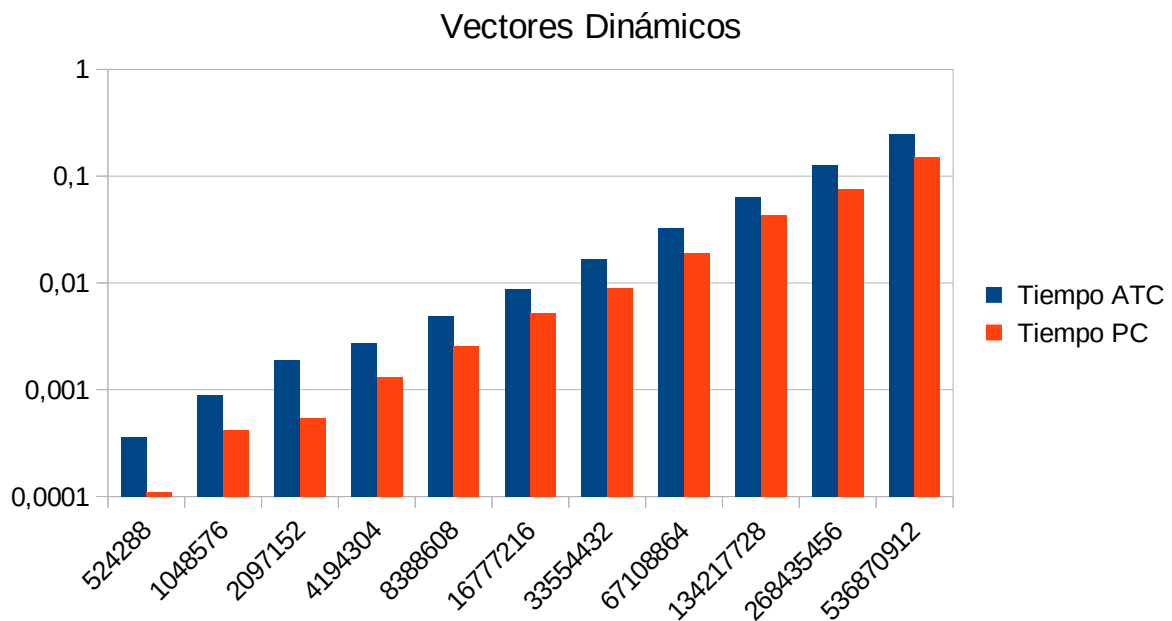
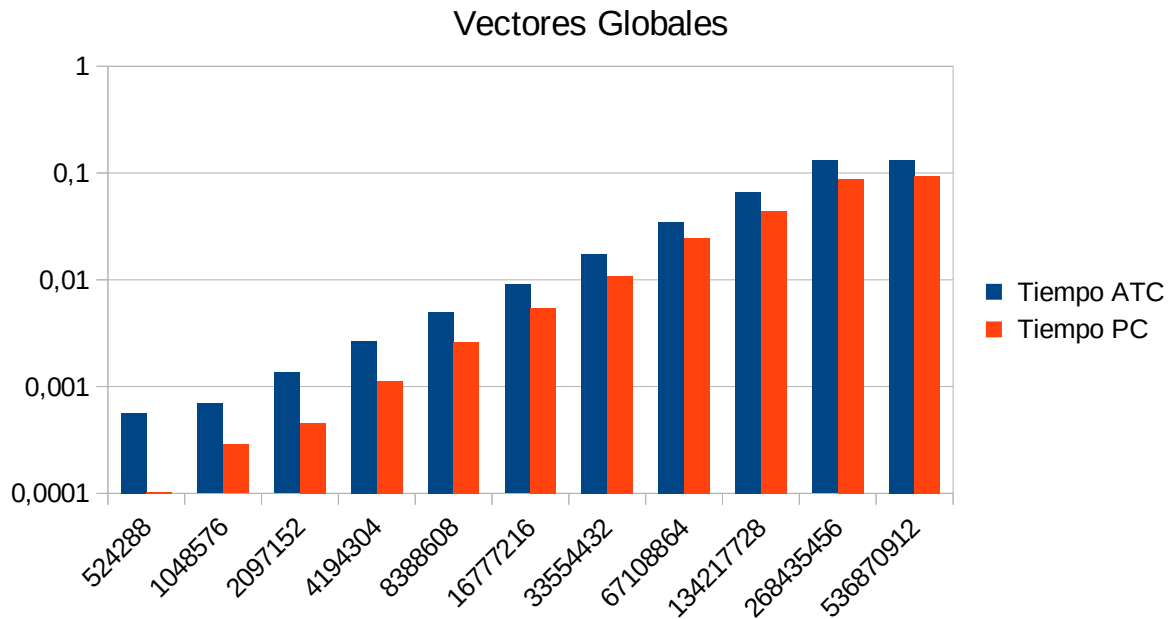
Ejecución en mi pc

- Con ayuda de la hoja de cálculo representar **en una misma gráfica** los tiempos de ejecución obtenidos en atcgid 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:

Vectores Locales





Vemos que en mi caso, mi ordenador es más rápido en todos los casos que el atcgrid.

2. Contestar a las siguientes preguntas:

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

RESPUESTA:

A partir de de tamaño de bytes de 2^{22} se produce segmentation fault en los programas. Esto ocurre puesto que este vector se almacena en la pila y esta tiene un tamaño limitado.

```

a1estudiante23@atcgrid:~/bp0/ejer7
[a1estudiante23@atcgrid ejer7]$ srun -Aac -pac SumaVectores 262144
Tiempo(seg.):0.001725923 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](0.792537+1.258343=2.050881) /
/ V1[262143]+V2[262143]=V3[262143](0.321585+2.555353=2.876938) /
[a1estudiante23@atcgrid ejer7]$ srun -Aac -pac SumaVectores 524288
srun: error: atcgrid1: task 0: Segmentation fault (core dumped)
[a1estudiante23@atcgrid ejer7]$ |

```

Vemos que a partir de cierto número de elementos el programa falla (ATCGRID)

```

bash /home/salva
a1estudiante23@atcgrid:~/bp0/ejer7
[SalvadorRomeroCortés salva@pop-os:~/Uni/2Info-C2/AC/bp0/ejer7] 2021-03-13 sábado
$ ./SumaVectores 262144
Tiempo(seg.):0.000626249 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](0.553623+0.814270=1.367893) /
/ V1[262143]+V2[262143]=V3[262143](1.595453+0.615089=2.210542) /
[SalvadorRomeroCortés salva@pop-os:~/Uni/2Info-C2/AC/bp0/ejer7] 2021-03-13 sábado
$ ./SumaVectores 524288
Violación de segmento ('core' generado)
[SalvadorRomeroCortés salva@pop-os:~/Uni/2Info-C2/AC/bp0/ejer7] 2021-03-13 sábado
$|

```

Vemos la misma situación anterior (PC)

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

RESPUESTA:

En este caso no se produce ningún error porque se almacena en la sección de código. Se puede ver en la captura del ejercicio 7 que no falla para ningún tamaño de 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:

En este caso no se produce ningún error porque se almacena en el heap, que se va reservando más a medida que se va necesitando. Se puede ver en la captura del ejercicio 7 que no falla para ningún tamaño de vector.

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

RESPUESTA:

Debido a que es de tipo `unsigned int` la variable ocupa 4 bytes. Como no tiene signo, esto quiere decir que el rango de valores es de 0 a $2^{32} - 1$.

- (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:

Hay que cambiar el `#define MAX 33554432` a `#define MAX 4294967295`.

```

bash /home/salva

//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 variab
les ...
// dinámicas (memoria reutilizable durante la ejecución)

#ifdef VECTOR_GLOBAL
//#define MAX 33554432 // = 2^25
#define MAX 4294967295 // = 2^32 - 1

double v1[MAX], v2[MAX], v3[MAX];
#endif
int main(int argc, char** argv){

SumaVectores.c 29,1 15%

```

Nuevo define con el nuevo límite

```

bash /home/salva

a1estudiante23@atcgriid~/bp0/ejer7
[SalvadorRomeroCortés salva@pop-os:~/Uni/2Info-C2/AC/bp0/ejer10] 2021-03-13 sábado
$gcc -O2 SumaVectores.c -o SumaVectores -lrt
/tmp/ccFmrVw0.o: en la función 'main':
SumaVectores.c:(.text.startup+0x68): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v2' definido en la sección COMMON en /tmp/ccFmrVw0.o
SumaVectores.c:(.text.startup+0xbb): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v3' definido en la sección COMMON en /tmp/ccFmrVw0.o
SumaVectores.c:(.text.startup+0x205): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v2' definido en la sección COMMON en /tmp/ccFmrVw0.o
collect2: error: ld returned 1 exit status
[SalvadorRomeroCortés salva@pop-os:~/Uni/2Info-C2/AC/bp0/ejer10] 2021-03-13 sábado
$

```

Resultados de compilar con el nuevo límite

Vemos como ahora ocurre un error de compilación puesto que el enlazador nos avisa de que no hay espacio suficiente para el tamaño que deseamos en la sección de código para variables globales.

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;
}

```