

BP0RomanAlvarezCarlos.pdf



BrokenQuagga



Arquitectura de Computadores



2º Grado en Ingeniería Informática



Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación
Universidad de Granada



Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.



Antes de comenzar a realizar el trabajo de este cuaderno consultar el fichero con las normas de prácticas que se encuentra en SWAD

Parte I. Ejercicios balogout

sados 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:

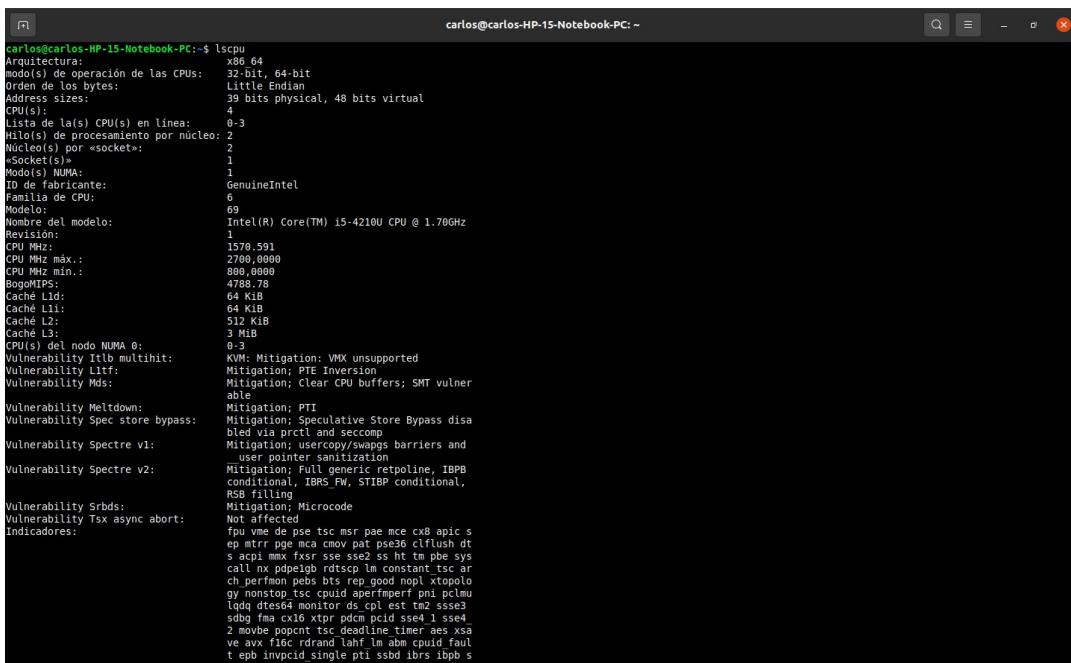
- Slurm está configurado para asignar recursos a los procesos (llamados *tasks* en slurm) a nivel de core físico. Esto significa que por defecto slurm asigna un core a un proceso, para asignar x se debe usar con sbatch/srun la opción --cpus-per-task=x (-cx).
- En slurm, por defecto, cpu se refiere a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

En mi PC



```
carlos@carlos-HP-15-Notebook-PC:~$ lscpu
Arquitectura: x86_64
Modo(s) de operación de las CPUs: 32-bit, 64-bit
ordenador de los bytes: littleEndian
Address sizes: 39 bits physical, 48 bits virtual
CPU(s): 4
Lista de la(s) CPU(s) en linea: 0-3
Hilo(s) de procesamiento por núcleo: 2
Núcleo(s) por «socket»: 2
«Socket(s»: 1
Modo(s) NUMA: 1
ID CPU: 0
Fabricante: GenuineIntel
Familia de CPU: 6
Modelo: 69
Nombre del modelo: Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz
Revisión: 1
CPU MHz: 1570.591
CPU MHz máx.: 2700.0000
CPU MHz min.: 800.0000
BogomIPS: 4785.78
Processor Id: 0
Caché L1i: 64 KiB
Caché L2: 512 KiB
Caché L3: 3 MiB
CPU(s) del nodo NUMA 0: 0-3
Vulnerability Itlb multihit: KVM: Mitigation: VMX unsupported
Vulnerability L1tf: Mitigation: PTE Inversion
Vulnerability L1tf: Mitigation: clear CPU buffers; SMT vulner
Vulnerability Mds: Mitigation: IBRS
Vulnerability Meltdown: Mitigation: PTI
Vulnerability Spec store bypass: Mitigation: Speculative Store Bypass disa
Vulnerability Spectre v1: bled via prctl and seccomp
Vulnerability Spectre v2: Mitigation: usercopy/swaps barriers and
Vulnerability Spectre v2: user pointer sanitization
Vulnerability Spectre v2: Mitigation: Full generic retpoline, IBPB
Vulnerability Spectre v2: conditional, IBRS_FW, STIBP conditional,
Vulnerability Spectre v2: RSB fill
Vulnerability Spectre v2: Mitigation: Microcode
Vulnerability Spectre v2: Not affected
Vulnerability Tsx async abort: fpu vme de pse tsc msr pae mce cx8 apic s
Vulnerability Tsx async abort: ep mtrr pge mca cmov pat pse36 clflush dt
Vulnerability Tsx async abort: s acpi mmx fxsr sse sse2 ss ht tm pbe sys
Vulnerability Tsx async abort: call nx pdpe1gb rdtsclm constant_tsc ar
Vulnerability Tsx async abort: ch perfmon pebs bits rep_good nopl xtopolo
Vulnerability Tsx async abort: gy nonstop_tsc cpuid aperfmperf pn1 pcmu
Vulnerability Tsx async abort: gdc rtm tsc_lar tsc_lar_tsc_lar_tsc_lar_tsc
Vulnerability Tsx async abort: srdt srdt_xtr pdce pcid sse4_1 sse4_2
Vulnerability Tsx async abort: movebe popcnt tsc deadline timer aes fsha
Vulnerability Tsx async abort: ve avx f16c rdrandlahf lm abm cpuid faul
Vulnerability Tsx async abort: t epb invpcid single pti ssbd ibrs ibpb s
```

En ac4

```

a2estudiante23@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:               1600.036
CPU max MHz:          3900.0000
CPU min MHz:          800.0000
BogoMIPS:              7000.00
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 dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm
                       constant_tsc arch_perfmon pebs bts rep_good nopl xtTopology nonstop_tsc aperfmpf eagerfpu
                       pni dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 pop
                       cnt lahf_lm epb ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriority ept vpid dtherm ida ar
                       at spec_ctrl intel stibp flush l1d
Last failed login: Mon Mar  1 17:48:04 CET 2021 from vpn-s246182.ugr.es on ssh:notty
There were 10 failed logins since the last successful login.
Last login: Mon Mar  1 17:43:59 2021 from vpn-s246182.ugr.es
(a2estudiante23@atcgrid:~)$

```

En ac

```

[a2estudiante23@atcgrid home]$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:             Little Endian
CPU(s):                24
On-line CPU(s) list:  0-23
Thread(s) per core:   2
Core(s) per socket:   6
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 44
Model name:            Intel(R) Xeon(R) CPU E5645 @ 2.40GHz
Stepping:              2
CPU MHz:               1600.000
CPU max MHz:          2401.0000
CPU min MHz:          1600.0000
BogoMIPS:              4799.93
Virtualization:       VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              12288K
NUMA node0 CPU(s):    0-5,12-17
NUMA node1 CPU(s):    6-11,18-23
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
                       pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm
                       constant_tsc arch_perfmon pebs bts rep_good nopl xtTopology nonstop_tsc aperfmpf eagerfpu
                       pni dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 pop
                       cnt lahf_lm epb ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriority ept vpid dtherm ida ar
                       at spec_ctrl intel stibp flush l1d

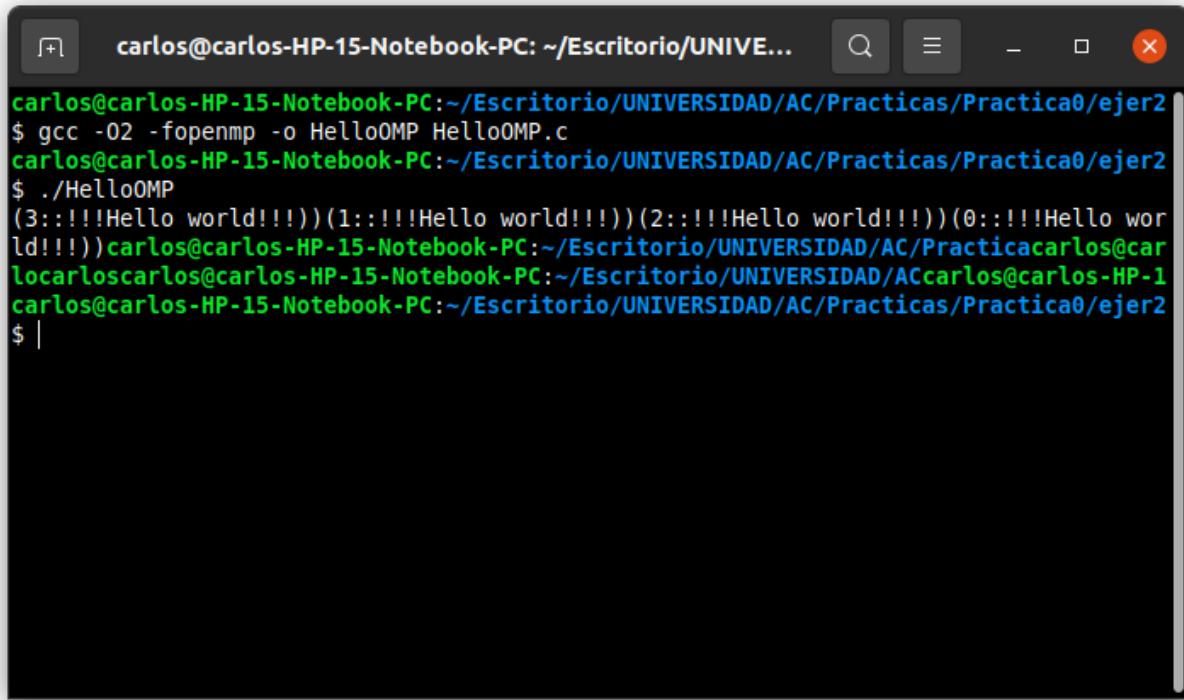
```

(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 4 cores lógicos y 4 físicos. El ac4 tiene 8 lógicos y 4 físicos. El ac tiene 24 lógicos y 3 físicos.

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.



```
carlos@carlos-HP-15-Notebook-PC:~/Escritorio/UNIVERSIDAD/AC/Practicas/Practica0/ejer2
$ gcc -O2 -fopenmp -o HelloOMP HelloOMP.c
carlos@carlos-HP-15-Notebook-PC:~/Escritorio/UNIVERSIDAD/AC/Practicas/Practica0/ejer2
$ ./HelloOMP
(3:!!!!Hello world!!!)(1:!!!!Hello world!!!)(2:!!!!Hello world!!!)(0:!!!!Hello wor
ld!!!)carlos@carlos-HP-15-Notebook-PC:~/Escritorio/UNIVERSIDAD/AC/Practicacarlos@car
locarloscarlos@carlos-HP-15-Notebook-PC:~/Escritorio/UNIVERSIDAD/ACcarlos@carlos-HP-1
carlos@carlos-HP-15-Notebook-PC:~/Escritorio/UNIVERSIDAD/AC/Practicas/Practica0/ejer2
$ |
```

RESPUESTA:

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

RESPUESTA: Aparecen 4 “Hello world” ya que es el numero de cores físicos que tengo

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:

```
[a2estudiante23@atcgrid ejer2]$ srun --partition=ac --account=ac --ntasks=1 --cpus-per-task=12 --hint=nomultithread HelloOMP
(0:!!!Hello world!!!)(2:!!!Hello world!!!)(4:!!!Hello world!!!)(8:!!!Hello world!!!)(10:!!!Hello world!!!)(9:!!!Hello world!!!)(3:!!!Hello world!!!)(5:!!!Hello wor
ld!!!)(7:!!!Hello world!!!)(11:!!!Hello world!!!)(1:!!!Hello world!!!)(6:!!!Hello world!!!)[a2estudiante23@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:

```
[a2estudiante23@atcgrid ejer2]$ srun -pac -Aac -n1 -c24 HelloOMP
(12:!!!Hello world!!!)(23:!!!Hello world!!!)(16:!!!Hello world!!!)(14:!!!Hello world!!!)(0:!!!Hello world!!!)(21:!!!Hello world!!!)(4:!!!Hello world!!!)(5:!!!Hello
world!!!)(20:!!!Hello world!!!)(19:!!!Hello world!!!)(11:!!!Hello world!!!)(9:!!!Hello world!!!)(10:!!!Hello world!!!)(7:!!!Hello world!!!)(22:!!!Hello world!!!)(1
1:!!!Hello world!!!)(13:!!!Hello world!!!)(15:!!!Hello world!!!)(8:!!!Hello world!!!)(6:!!!Hello world!!!)(1:!!!Hello world!!!)(22:!!!Hello world!!!)(3:!!!Hello wo
rld!!!)(18:!!!Hello world!!!)[a2estudiante23@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:

```
[a2estudiante23@atcgrid ejer2]$ srun -n1 HelloOMP
(0:!!!Hello world!!!)(1:!!!Hello world!!!)[a2estudiante23@atcgrid ejer2]$ |
```

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

```
srun --partition=ac --account=ac --ntasks=1 --cpus-per-task=32 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:

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

int main(void){
    #pragma omp parallel
    printf("(%d::::Hello world!!!) \n", omp_get_thread_num());

    #pragma omp parallel
    printf("(%d::::WORLD!!!) \n", omp_get_thread_num());

    return (0);
}
```

```
carlos@carlos-HP-15-Notebook-PC:~/Escritorio/UNIVERSIDAD/AC/Practicas/Practica0/ejer2$ gcc -o2 -fopenmp -o HelloOMP2 HelloOMP.c
carlos@carlos-HP-15-Notebook-PC:~/Escritorio/UNIVERSIDAD/AC/Practicas/Practica0/ejer2$ ./HelloOMP2
(0::::!Hello world!!!)
(2::::!Hello world!!!)
(3::::!Hello world!!!)
(1::::!Hello world!!!)
(2::::::WORLD!!!)
(3::::::WORLD!!!)
(1::::::WORLD!!!)
(0::::::WORLD!!!)
```

```
[a2estudiante23@atcgrid ejer4]$ sbatch -pac -n1 -c12 -h=nomultithread script_helloomp.sh
Submitted batch job 58421
[a2estudiante23@atcgrid ejer4]$ cat slurm-58421.out
Id. usuario del trabajo: a2estudiante23
Id. del trabajo: 58421
Nombre del trabajo especificado por usuario: helloOMP
Directorio de trabajo (en el que se ejecuta el script): /home/a2estudiante23/ejer4
Cola: ac
Nodo que ejecuta este trabajo:atcgrid.ugr.es
Nº de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atgrid1
CPUs por nodo: 24

1. Ejecución HelloOMP2 una vez sin cambiar nº de threads (valor por defecto):

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

```
2. Ejecución HhelloOMP2 varias veces con distinto nº de threads:

- Para 12 threads:
(6::::::Hello world!!!!)
(1::::::Hello world!!!!)
(3::::::Hello world!!!!)
(11::::::Hello world!!!!)
(2::::::Hello world!!!!)
(10::::::Hello world!!!!)
(0::::::Hello world!!!!)
(5::::::Hello world!!!!)
(4::::::Hello world!!!!)
(9::::::Hello world!!!!)
(7::::::Hello world!!!!)
(8::::::Hello world!!!!)
(4::::::world!!!!)
(1::::::world!!!!)
(2::::::world!!!!)
(6::::::world!!!!)
(10::::::world!!!!)
(7::::::world!!!!)
(3::::::world!!!!)
(8::::::world!!!!)
(11::::::world!!!!)
(5::::::world!!!!)
(0::::::world!!!!)
(9::::::world!!!!)

- Para 6 threads:
(0::::::Hello world!!!!)
(3::::::Hello world!!!!)
(2::::::Hello world!!!!)
(1::::::Hello world!!!!)
(5::::::Hello world!!!!)
(4::::::Hello world!!!!)
(2::::::world!!!!)
(4::::::world!!!!)
(0::::::world!!!!)
(3::::::world!!!!)
(5::::::world!!!!)
(1::::::world!!!!)

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

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

RESPUESTA: En el nodo 1

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:

```
carlos@carlos-HP-15-Notebook-PC:~/Escritorio/UNIVERSIDAD/AC/Practicas/Practica0$ gcc -O2 SumaVectoresC.c -o SumaVectores -lrt
carlos@carlos-HP-15-Notebook-PC:~/Escritorio/UNIVERSIDAD/AC/Practicas/Practica0$ ./SumaVectores 2000
Tiempo(seg.): 0.000004767 / TamañoVectores: 2000
/ V1[0] + V2[0] = V3[0](1.545014 + 0.662076 = 2.207090) /
/ V1[1999] + V2[1999] = V3[1999](0.361336 + 5.408826 = 5.770162) /
```

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

(a) ¿Qué contiene esta variable?

RESPUESTA: El tiempo transcurrido durante la ejecución de la suma de vectores, es decir, la diferencia de tiempo desde que se llama por primera vez a *clock_gettime()* hasta que se llama por segunda vez a dicha función. Esta diferencia se calcula como diferencia del segundo instante de tiempo menos el primero con una precisión de nanosegundos tal y como se muestra en la sentencia:

```
ncgt=(double)(cgt2.tv_sec-cgt1.tv_sec)+(double)((cgt2.tv_nsec-cgt1.tv_nsec)/(1.c+9));
```

(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 la información de tiempo en la estructura de datos *timespec*, tal y como se especifica en *<time.h>*:

```
struct timespec{
    time_t    tv_sec;
    long      tv_nsec;
};
```

(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: El inicio de este contador está en el 1 de Enero de 1970 a las 00:00 (como valor numérico tendría 0). Las funciones *clock_gettime()* y *clock_settime()* recuperan y establecen el tiempo de un determinado reloj *clk_id*.

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
PC

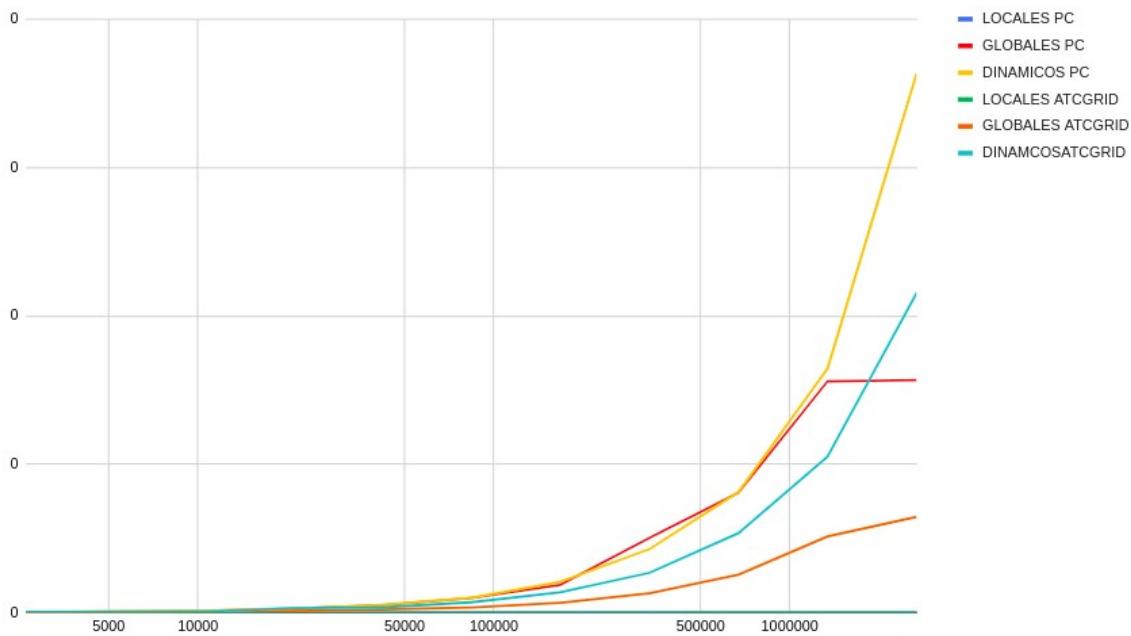
Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	262144	0,000096048	0,000292294	0,000290681
131072	524288	0,000253729	0,000799936	0,000896586
262144	1048576	0,000747021	0,001217882	0,001397063
524288	2097152	0	0,002879978	0,002478763
1048576	4194304	0	0,005092727	0,005326327
2097152	8388608	0	0,009863346	0,009984320
4194304	16777216	0	0,018832727	0,020847209
8388608	33554432	0	0,050389873	0,042803732
16777216	67108864	0	0,080786699	0,081283573
33554432	134217717	0	0,155889708	0,164464554
67108864	268435456	0	0,156748881	0,362926686

AC

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	262144	0,000110564	0,000207829	0,000445294
131072	524288	0,000198995	0,000203200	0,000394520
262144	1048576	0,000201396	0,000667637	0,000828547
524288	2097152	0	0,001132832	0,003053195
1048576	4194304	0	0,002240018	0,003588083
2097152	8388608	0	0,003418938	0,007029226
4194304	16777216	0	0,006754465	0,013725061
8388608	33554432	0	0,012988905	0,026847504
16777216	67108864	0	0,025664904	0,053602446
33554432	134217717	0	0,051321226	0,105248458
67108864	268435456	0	0,06454635	0,215436218

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: Si hay diferencia entre los tiempos, de tal forma que influye la velocidad de procesamiento



2. Contestar a las siguientes preguntas:

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

RESPUESTA: Si se produce un desbordamiento de pila, esto quiere decir que el tamaño de la pila con los vectores locales está limitada y por eso sale “Violación del segmento” cuando metemos un tamaño superior a 524288. De tal forma se utiliza para valoren que no sean muy grandes y puedan caber en la pila.

(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, puesto que no está limitada el tamaño de pila y tiene un control máximo en el programa del que si se supera se interpreta que es 33554432 * 4Bytes el tamaño del vector, de tal forma que se controla el tamaño en el programa. Tanto si se controla o no el tamaño no se produciría desbordamiento de pila.

(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: Tampoco se produce desbordamiento algo similar al anterior, pero el tamaño de pila esta limitado, la diferencia a la local es que el tamaño en el que esta limitado el vector dinámico es mayor al local y por eso no produce desbordamiento de pila.

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

RESPUESTA: Como N es definido como unsigned int, su tamaño es de 4Bytes = 32bits, entonces $N = 2^{32} - 1$.

(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: El compilador da error, porque supera el tamaño permitido para reservar memoria en un vector de $2^{32}-1$ posiciones, es decir son $(2^{32}-1) * 8\text{Bytes}$ por posición = $3,436 * 10^{10}$ Bytes = 31,999GB ocuparía el vector, de tal forma que habría desbordamiento de memoria.

```
carlos@carlos-HP-15-Notebook-PC:~/Escritorio/UNIVERSIDAD/AC/Practicas/Practicas$ gcc -O2 SumaVectoresC2.c -o SumaVectores2 -lrt
/tmp/ccIRGbKz.o: en la función `main':
SumaVectoresC2.c:(.text.startup+0x60): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v1' definido en la sección .bss en /tmp/ccIRGbKz.o
SumaVectoresC2.c:(.text.startup+0x67): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v2' definido en la sección .bss en /tmp/ccIRGbKz.o
SumaVectoresC2.c:(.text.startup+0x1e3): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v1' definido en la sección .bss en /tmp/ccIRGbKz.o
SumaVectoresC2.c:(.text.startup+0x1ea): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v2' definido en la sección .bss en /tmp/ccIRGbKz.o
collect2: error: ld returned 1 exit status
```

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 (arg<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

//Iniciarizar 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,&cg1);
//Calcular suma de vectores
for(i=0; i<N; i++)
    v3[i] = v1[i] + v2[i];

clock_gettime(CLOCK_REALTIME,&cg2);
ncgt=(double) (cg2.tv_sec-cg1.tv_sec) +
(double) ((cg2.tv_nsec-cg1.tv_nsec)/(1.e+9));

//Imprimir resultado de la suma y el tiempo de ejecución
if (N<10) {
    printf("Tiempo(seg.):%11.9f\n / 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.):%11.9ft / Tamaño Vectores:%u\n/ V1[0]+V2[0]=V3[0](%8.6f+%8.6f=%8.6f) //\nV1[%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]);\n\n#endif VECTOR_DYNAMIC\nfree(v1); // libera el espacio reservado para v1\nfree(v2); // libera el espacio reservado para v2\nfree(v3); // libera el espacio reservado para v3\n#endif\nreturn 0;\n}
```