2º curso / 2º cuatr.

Grado Ingeniería Informática

Arquitectura de Computadores (AC)

Cuaderno de prácticas.
Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos): Álvaro Vega Romero
Grupo de prácticas y profesor de prácticas: B2 – Niceto Rafael Luque Sola
Fecha de entrega: - 11/03/2021
Fecha evaluación en clase: - 19/03/2021

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
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
Vendor ID:           GenuineIntel
CPU family:          6
Model:               126
Model name:          Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz
Stepping:            5
CPU MHz:             1498.000
CPU max MHz:         1498.0000
BogoMIPS:            2996.00
Hypervisor vendor:   Windows Subsystem for Linux
Virtualization type: container
Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr s
se sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4_1 sse4
_2 movbe popcnt aes xsave osxsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch fsgsbase tsc_adjust bmi1 avx2 sme
p bmi2 erms invpcid avx512f avx512dq rdseed adx smap avx512ifma clflushopt intel_pt avx512cd sha_ni avx512bw avx512vl av
x512vbmi umip avx512_vbmi2 gfni vaes vpclmulqdq avx512_vnni avx512_bitalg avx512_vpopcntdq rdpid ibrs ibpb stibp ssbd
[AlvaroVega zuki@LAPTOP-9IDMGVGU:~] 2021-02-27 Saturday
```

```
[ÁlvaroVega b2estudiante24@atcgrid:~/bp0/ejer1] 2021-02-27 Saturday
$sbatch -p ac4 --wrap "lscpu"
Submitted batch job 57006
[ÁlvaroVega b2estudiante24@atcgrid:~/bp0/ejer1] 2021-02-27 Saturday
$cat slurm-57006.out
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:               799.932
CPU max MHz:           3200.0000
CPU min MHz:           800.0000
BogoMIPS:              4200.00
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              1024K
L3 cache:              22528K
NUMA node0 CPU(s):     0-15,32-47
NUMA node1 CPU(s):     16-31,48-63
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts
rep_good 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 a
vx f16c rdrand lahf_lm abm 3dnowprefetch epb 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 b
mi2 erms invpcid rtm 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 pk
u ospke avx512_vnni md_clear spec_ctrl intel_stibp flush_l1d arch_capabilities
[ÁlvaroVega b2estudiante24@atcgrid:~/bp0/ejer1] 2021-02-27 Saturday
```

```
[ÁlvaroVega b2estudiante24@atcgrid:~/bp0/ejer1] 2021-02-27 Saturday
$sbatch -p ac --wrap "lscpu"
Submitted batch job 57007
[ÁlvaroVega b2estudiante24@atcgrid:~/bp0/ejer1] 2021-02-27 Saturday
$cat slurm-57007
cat: slurm-57007: No such file or directory
[ÁlvaroVega b2estudiante24@atcgrid:~/bp0/ejer1] 2021-02-27 Saturday
$ls
slurm-57006.out  slurm-57007.out
[ÁlvaroVega b2estudiante24@atcgrid:~/bp0/ejer1] 2021-02-27 Saturday
$cat slurm-57007.out
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 xtopology nonstop_tsc aperfmperf eagerfpu pni dtes64 monitor 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
[ÁlvaroVega b2estudiante24@atcgrid:~/bp0/ejer1] 2021-02-27 Saturday
```

(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:

ac4 tiene 64 cores lógicos y cómo podemos observar y al tener dos threads por cada core pues vemos como es la mitad de cores físicos que es 32.

En cambio, ac1 tiene 24 cores lógicos y 12 físicos (ac2 y ac3 son iguales a al otro)

Por último, la computadora que uso dispone de 8 cores lógicos y 4 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.

RESPUESTA:

```
[AlvaroVega zuki@LAPTOP-9IDMGVGU:~/bp0/ejer2] 2021-02-27 Saturday
$gcc -O2 -fopenmp -o HelloOMP HelloOMP.c
[AlvaroVega zuki@LAPTOP-9IDMGVGU:~/bp0/ejer2] 2021-02-27 Saturday
$./HelloOMP
(7:!!!Hello world!!!)(4:!!!Hello world!!!)(3:!!!Hello world!!!)(0:!!!Hello world!!!)(1:!!!Hello world!!!)(5:!!!Hello world!!!)(2:!!!Hello world!!!)(6:!!!Hello world!!!)[AlvaroVega zuki@LAPTOP-9IDMGVGU:~/bp0/ejer
2] 2021-02-27 Saturday
$./HelloOMP > solucion
[AlvaroVega zuki@LAPTOP-9IDMGVGU:~/bp0/ejer2] 2021-02-27 Saturday
$cat solucion
(0:!!!Hello world!!!)(5:!!!Hello world!!!)(7:!!!Hello world!!!)(3:!!!Hello world!!!)(6:!!!Hello world!!!)(1:!!!Hello world!!!)(4:!!!Hello world!!!)(2:!!!Hello world!!!)[AlvaroVega zuki@LAPTOP-9IDMGVGU:~/bp0/ejer
2] 2021-02-27 Saturday
$cat HelloOMP.c
/* Compilar con:
gcc -O2 -fopenmp -o HelloOMP HelloOMP.c
*/
#include <stdio.h>
#include <omp.h>

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

return(0);
}

[AlvaroVega zuki@LAPTOP-9IDMGVGU:~/bp0/ejer2] 2021-02-27 Saturday
```

(Cabe destacar que para que su formato sea un Hello World por cada línea, deberíamos poner un /n al final de cada Hello World).

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

RESPUESTA:

Imprime 8 "Hello world" que coincide con el número de cores lógicos ya que son 8 hebras al tener 4 cores físicos y tener dos hebras por cada core físico

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:

```
[ÁlvaroVega b2estudiante24@atcgrid:~/bp0/ejer3] 2021-02-27 Saturday
$chmod +x HelloOMP
[ÁlvaroVega b2estudiante24@atcgrid:~/bp0/ejer3] 2021-02-27 Saturday
$srun --partition=ac --account=ac --ntasks=1 --cpus-per-task=12 --hint=nomultithread HelloOMP
(2:!!!Hello world!!!)(1:!!!Hello world!!!)(8:!!!Hello world!!!)(0:!!!Hello world!!!)(6:!!!Hello world!!!)(4:!!!Hello world!!!)(10:!!!Hello world!!!)(5:!!!Hello world!!!)(11:!!!Hello world!!!)(3:!!!Hello world!!!
)(7:!!!Hello world!!!)(9:!!!Hello world!!!)[ÁlvaroVega b2estudiante24@atcgrid:~/bp0/ejer3] 2021-02-27 Saturday
```

(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

```
$srun -pac -Aac -n1 -c24 HelloOMP
(15:!!!Hello world!!!)(19:!!!Hello world!!!)(17:!!!Hello world!!!)(23:!!!Hello world!!!)(11:!!!Hello world!!!)(14:!!!Hel
lo world!!!)(8:!!!Hello world!!!)(13:!!!Hello world!!!)(3:!!!Hello world!!!)(5:!!!Hello world!!!)(21:!!!Hello world!!!)(
2:!!!Hello world!!!)(7:!!!Hello world!!!)(18:!!!Hello world!!!)(4:!!!Hello world!!!)(1:!!!Hello world!!!)(0:!!!Hello wor
ld!!!)(9:!!!Hello world!!!)(20:!!!Hello world!!!)(16:!!!Hello world!!!)(6:!!!Hello world!!!)(22:!!!Hello world!!!)(10:!!
!Hello world!!!)(12:!!!Hello world!!!)[ÁlvaroVega b2estudiante24@atcgrid:~/bp0/ejer3] 2021-03-05 Friday
```

(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:

```
[ÁlvaroVega b2estudiante24@atcgrid:~/bp0/ejer3] 2021-02-27 Saturday
$srun -n1 HelloOMP
(0:!!!Hello world!!!)(1:!!!Hello world!!!)[ÁlvaroVega b2estudiante24@atcgrid:~/bp0/ejer3] 2021-02-27 Saturday
```

Usa la partición por defecto y(que es ac) por tanto, utiliza un único core que tiene 2 hebras y por tanto la salida son 2 hello world

(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

-n1 → para crear solo un proceso

-p ac4 → escoger la cola ac4

-cpus-per-task=32 → srun use 12 cores físicos

--hint=nomultithread → Especificamos que no use cores lógicos

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.

Compilacion y prueba de HelloOMP2 en mi pc:

```
[AlvaroVega zuki@LAPTOP-9IDMGVGU:~/bp0/ejer4] 2021-02-27 Saturday
$gcc -O2 -fopenmp -o HelloOMP2 HelloOMP2.c
[AlvaroVega zuki@LAPTOP-9IDMGVGU:~/bp0/ejer4] 2021-02-27 Saturday
$cat HelloOMP2.c
/* Compilar con:
gcc -O2 -fopenmp -o HelloOMP2 HelloOMP2.c
*/
#include <stdio.h>
#include <omp.h>

int main(void)
{
        #pragma omp parallel
        {
        printf("(%d:!!!Hello)",omp_get_thread_num());
        printf("(%d:world!!!)",omp_get_thread_num());
        }
return(0);
}

[AlvaroVega zuki@LAPTOP-9IDMGVGU:~/bp0/ejer4] 2021-02-27 Saturday
$./HelloOMP2
(3:!!!Hello)(3:world!!!)(2:!!!Hello)(2:world!!!)(4:!!!Hello)(4:world!!!)(5:!!!Hello)(5:world!!!)(7:!!!Hello)(7:world!!!)(1:!!!Hello)(1:world!!!)(0:!!!Hello)(0:world!!!)(6:!!!Hello)(6:world!!!)[AlvaroVega zuki@LA
PTOP-9IDMGVGU:~/bp0/ejer4] 2021-02-27 Saturday
```

RESPUESTA:

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

Hemos usado el nodo atcgrid1 ya que lo podemos ver en el propio script con la variable $SLURM_JOB_NODELIST: "*Nodos asignados al trabajo: atcgrid1*"

RESPUESTA:

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:

```
[AlvaroVega zuki@LAPTOP-9IDMGVGU:~/bp0/ejer5] 2021-03-05 Friday
$gcc -O2 SumaVectoresC.c -o SumaVectoresC -lrt
SumaVectoresC.c: In function ‘main’:
SumaVectoresC.c:88:54: warning: format ‘%lu’ expects argument of type ‘long unsigned int’, but argument 3 has type ‘unsi
gned int’ [-Wformat=]
   printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%lu\n",ncgt,N);
                                                      ~~~^
                                                      %u
```

```
[AlvaroVega zuki@LAPTOP-9IDMGVGU:~/bp0/ejer5] 2021-03-05 Friday
$./SumaVectoresC 50
Tiempo(seg.):0.000000800        / Tamaño Vectores:50    / V1[0]+V2[0]=V3[0](4.670025+39.782077=44.452103) / / V1[49]+V2[49]=V3[49](3.618668+0.473688=4.092356) /
```

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 que tarda en sumar los vectores el cual es igual a la diferencia de tiempo desde que se llama a clock_gettime() hasta que se llama por segunda vez. La diferencia entre ambas variables se calcula como un double que contiene la diferencia en nanosegundos.

(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:
Devuelve la información en la estructura timespec.
La estructura se especifica en <time.h> y es:

```
struct timespec {
        time_t ty_sec; /*seconds*/
        long ty_nsec; /*nanoseconds*/
};
```

Los tipos de datos son:
- long: para indicar el número de nano segundos
- time_t para el número de segundos.

(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 función devuelve un struct con una hora en específico.

Por ejemplo, clock_gettime(CLOCK_REALTIME,&cgt1); lo que haría sería meter en el objeto cgt1 la hora actual.

Los valores devueltos por la función representan el número de segundos y nanosegundos que han transcurrido en función un determinado reloj.

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 (cola ac) 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 . Tabla en mi ordenador

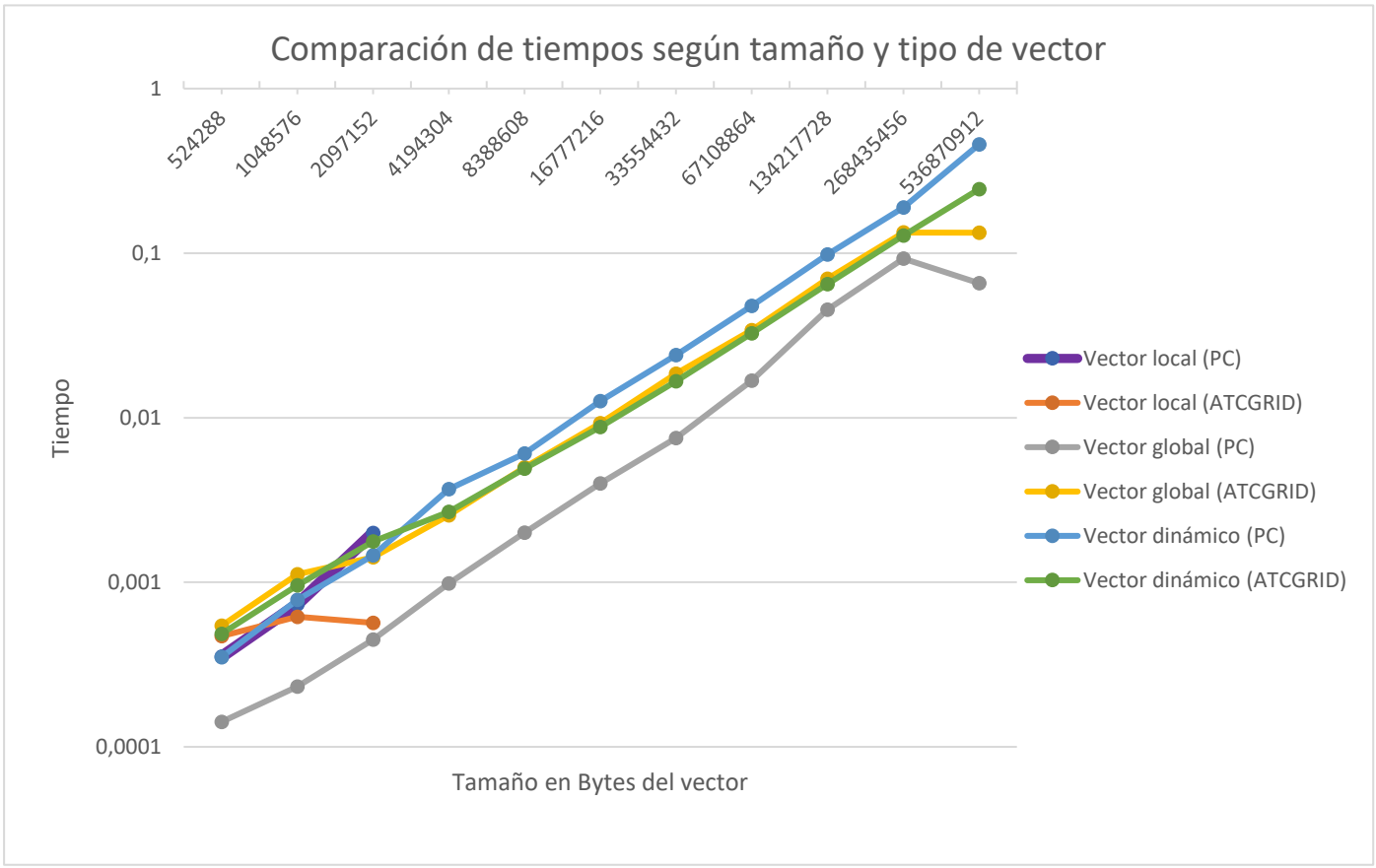
Nº de Componentes	Bytes de un vector(KB)	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	512KB	0,0003529	0,0001418	0,0003511
131072	2^10KB	0,0007361	0,0002328	0,0007839
262144	2^11	0,0019955	0,0004476	0,0014552
524288	2^12	-	0,0009824	0,0036752
1048576	2^13	-	0,0020021	0,0060699
2097152	2^14	-	0,003985	0,0126038
4194304	2^15	-	0,0075116	0,0240087
8388608	2^16	-	0,0168364	0,478199
16777216	2^17	-	0,045291	0,09838922
33554432	2^18	-	0,0926547	0,1897986
67108864	2^19	-	0,0656835	0,457683

Nº de Componentes	Bytes de un vector(KB)	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	512KB	0,00047084	0,000544397	0,00048655
131072	2^10KB	0,00061765	0,001119208	0,000958032
262144	2^11	0,000565887	0,00141769	0,001767766
524288	2^12		0,00254872	0,002674013
1048576	2^13		0,004988796	0,004901309
2097152	2^14		0,009262535	0,008782904
4194304	2^15		0,01852513	0,1662841
8388608	2^16		0,34053606	0,032644009
16777216	2^17		0,069962867	0,064722496
33554432	2^18		0,133435746	0,12786419
67108864	2^19		0,133282589	0,24505253

Tamaño = num_componentes * tam de cada componente (double = 8B)

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:



(He intentado que el eje horizontal no me lo ponga arriba de todas las maneras y no he conseguido volver a dejarlo abajo tras aplicar la escala logarítmica)
Obviamente los tiempos suelen ser un poco mejores en ATCGRID pero sin tampoco ser una diferencia abismal con respecto a mi ordenador

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:

Se obtienen errores para cuando sobrepasamos el tamaño de la pila que podemos ver con ulimit -a y que en mi caso es de 8MB (cuando superamos las 2^19 componentes)

```
[AlvaroVega zuki@LAPTOP-9IDMGVGU:~] 2021-03-05 Friday
$ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 7823
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 7823
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
[AlvaroVega zuki@LAPTOP-9IDMGVGU:~] 2021-03-05 Friday
```

(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 he obtenido ningún error, aunque a partir de un cierto tamaño, (en este caso el tamaño de componente sea mayor o igual a 2^25) deja de aumentar el tamaño ya que para vectores globales tenemos un máximo que es el del segmento de datos (2GB)

Así, e

(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:

No hay error ya que los datos se almacenan en el heap y no hay problemas de espacio ya que se reserva dinámicamente y por tanto, no es fijo como en los anteriores casos.

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

RESPUESTA:

Ya que N es de tipo unsigned y este ocupa 4B, el mayor valor que podemos almacenar es 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:

```
#ifdef VECTOR_GLOBAL
#define MAX 4294967295          //=2^32-1
double v1[MAX], v2[MAX], v3[MAX];
#endif
```

El error lo da el enlazador, que es el que se encarga de las dependencias de las variables globales.

El puntero RX86_64_PC32 sirve para direccionar 32B (funciones globales a algunas direcciones de memoria) y si tuviésemos que direccionar mas de 32B no podríamos

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;
}
```