2º curso / 2º cuatr.

Grado Ingeniería Informática

Arquitectura de Computadores (AC)

Cuaderno de prácticas.
Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos): Jose Luis Pedraza Román
Grupo de prácticas y profesor de prácticas: A2, Christian Morillas
Fecha de entrega: 4 marzo
Fecha evaluación en clase: 5 marzo

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 local.

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 más de uno se debe usar con `sbatch/srun` la opción `--cpus-per-task`.
- En slurm, por defecto, `cpu` se refiere a cores lógicos (ej. en la opción `--cpus-per-task`), 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 `-n1` en `sbatch/srun`.
- Para que no se ejecute más de un proceso en un nodo 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 slurm.

1. Ejecutar `lscpu` en el PC y en un nodo de cómputo de atcgrid. (Crear directorio `ejer1`)

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

RESPUESTA:

PC:

```
[JoseLuisPedrazaRoman jos@jos-GE62VR-7RF:~/Escritorio/practAC/BP0] 2020-02-20 jueves
$lscpu
Arquitectura:                    x86_64
modo(s) de operación de las CPUs:    32-bit, 64-bit
Orden de los bytes:              Little Endian
CPU(s):                          8
Lista de la(s) CPU(s) en línea:      0-7
Hilo(s) de procesamiento por núcleo: 2
Núcleo(s) por «socket»:          4
«Socket(s)»                      1
Modo(s) NUMA:                    1
ID de fabricante:                GenuineIntel
Familia de CPU:                  6
Modelo:                          158
Nombre del modelo:               Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz
Revisión:                        9
CPU MHz:                         2103.118
CPU MHz máx.:                    3800,0000
CPU MHz mín.:                    800,0000
BogoMIPS:                        5616.00
Virtualización:                  VT-x
Caché L1d:                       32K
Caché L1i:                       32K
Caché L2:                        256K
Caché L3:                        6144K
CPU(s) del nodo NUMA 0:          0-7
Indicadores:                     fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pg
 xtopology nonstop_tsc cpuid aperfmperf tsc_known_freq pni pclmulqdq dtes64 monitor ds_
etch cpuid_fault epb invpcid_single pti ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriorit
 pln pts hwp hwp_notify hwp_act_window hwp_epp md_clear flush_l1d
[JoseLuisPedrazaRoman jos@jos-GE62VR-7RF:~/Escritorio/practAC/BP0] 2020-02-20 jueves
$
```

ATCGRID:

```
[JoseLuisPedrazaRoman a2estudiante18@atcgrid:~/bp0] 2020-02-20 jueves
$cd ejer1
[JoseLuisPedrazaRoman a2estudiante18@atcgrid:~/bp0/ejer1] 2020-02-20 jueves
$ls
[JoseLuisPedrazaRoman a2estudiante18@atcgrid:~/bp0/ejer1] 2020-02-20 jueves
$sbatch -p ac --wrap "lscpu"
Submitted batch job 6403
[JoseLuisPedrazaRoman a2estudiante18@atcgrid:~/bp0/ejer1] 2020-02-20 jueves
$ls
slurm-6403.out
[JoseLuisPedrazaRoman a2estudiante18@atcgrid:~/bp0/ejer1] 2020-02-20 jueves
$cat slurm-6403.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:              4800.38
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 m
 tsc aperfmperf eagerfpu pni dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx1
[JoseLuisPedrazaRoman a2estudiante18@atcgrid:~/bp0/ejer1] 2020-02-20 jueves
$
```

(b) ¿Cuántos cores físicos y cuántos cores lógicos tienen los nodos de cómputo de atcgrid y el PC? Razonar las respuestas

RESPUESTA:

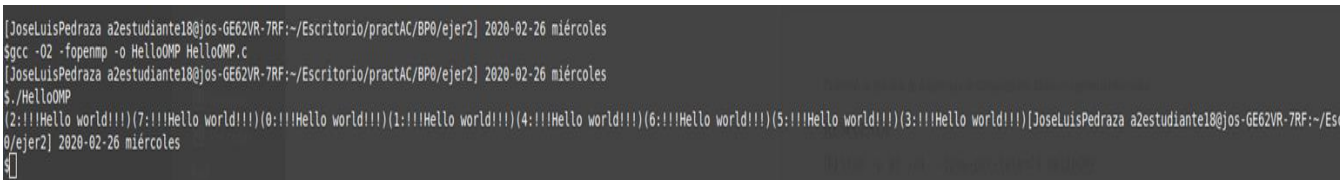
ATCGRID: Cada placa tiene 2 sockets (2 chips de cpu físicos). Cada chip físico tiene 6 núcleos físicos. Cada núcleo físico tiene 2 hebras (núcleos lógicos). 12 núcleos físicos y 24 núcleos lógicos.

PC: Un chip, 4 núcleos físicos y 8 núcleos lógicos.

2. Compilar y ejecutar en el PC el código `HelloOMP.c` del seminario (recordar que 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`, como se indica en las normas de prácticas).

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

RESPUESTA:



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

RESPUESTA:

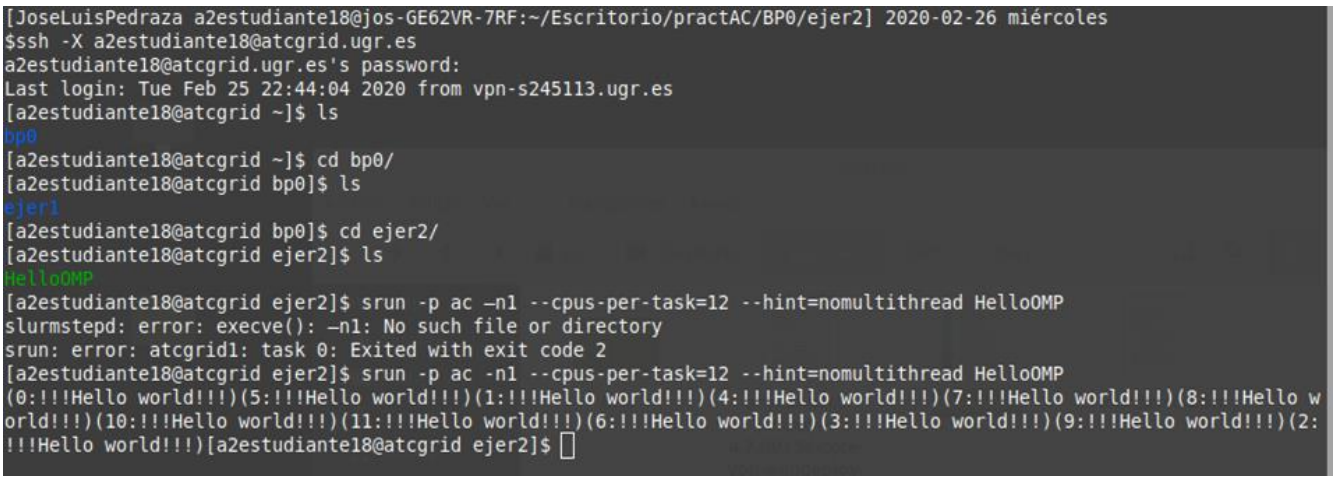
Como tengo 8 cores lógicos en mi PC, se imprime el mensaje 8 veces (de 0-7).

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 a través de cola ac del gestor de colas (no use ningún *script*) utilizando directamente en línea de comandos:

(a) `srun -p ac -n1 --cpus-per-task=12 --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:



(b) `srun -p ac -n1 --cpus-per-task=24 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:

```
[a2estudiante18@atcgrid ejer2]$ srun -p ac -n1 --cpus-per-task=24 HelloOMP
(21:!!!Hello world!!!)(13:!!!Hello world!!!)(15:!!!Hello world!!!)(20:!!!Hello world!!!)(1:!!!Hello world!!!)(0:!!!Hel
lo world!!!)(3:!!!Hello world!!!)(17:!!!Hello world!!!)(10:!!!Hello world!!!)(12:!!!Hello world!!!)(19:!!!Hello world!
!!)(22:!!!Hello world!!!)(8:!!!Hello world!!!)(23:!!!Hello world!!!)(9:!!!Hello world!!!)(18:!!!Hello world!!!)(5:!!!H
ello world!!!)(2:!!!Hello world!!!)(11:!!!Hello world!!!)(6:!!!Hello world!!!)(16:!!!Hello world!!!)(7:!!!Hello world!
!!)(4:!!!Hello world!!!)(14:!!!Hello world!!!)[a2estudiante18@atcgrid ejer2]$
```

(c) `srun -p ac -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.

RESPUESTA:

```
[a2estudiante18@atcgrid ejer2]$ srun -p ac -n1 HelloOMP
(0:!!!Hello world!!!)(1:!!!Hello world!!!)[a2estudiante18@atcgrid ejer2]$
```

(d) ¿Qué orden `srun` usaría para que HelloOMP utilice los 12 cores físicos de un nodo de cómputo de atcgrid (se debe imprimir un único mensaje desde cada uno de ellos, en total, 12)?

RESPUESTA:

La de la opción a: `srun -p ac -n1 --cpus-per-task=12 --hint=nomultithread HelloOMP`

4. Modificar en su PC `HelloOMP.c` para que se imprima "world" en un `printf` distinto al usado para "Hello", en ambos `printf` se debe imprimir el identificador del thread que escribe en pantalla. Nombrar al código resultante `HelloOMP2.c`. Compilar este nuevo código en el PC y ejecutarlo. Copiar el fichero ejecutable resultante al front-end de atcgrid (directorio `ejer4`). Ejecutar el código en un nodo de cómputo de atcgrid usando el script `script_helloomp.sh` del seminario (el nombre del ejecutable en el script debe ser `HelloOMP2`).

(a) Utilizar: `sbatch -p ac -n1 --cpus-per-task=12 --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:

```
1  /* Compilar con:
2  gcc -O2 -fopenmp -o HelloOMP2 HelloOMP2.c
3  */
4  #include <stdio.h>
5  #include <omp.h>
6
7  int main(void) {
8  #pragma omp parallel
9  {
10     printf("(%d:!!!Hello",
11         omp_get_thread_num());
12     printf("(%d:world!!!",
13         omp_get_thread_num());
14 }
15 return(0);
16 }
```

```
[JoseLuisPedraza a2estudiante18@jos-GE62VR-7RF:~/Escritorio/practAC/BP0/ejer4/bi
en] 2020-03-04 miércoles
$gcc -O2 -fopenmp -o HelloOMP2 HelloOMP2.c
[JoseLuisPedraza a2estudiante18@jos-GE62VR-7RF:~/Escritorio/practAC/BP0/ejer4/bi
en] 2020-03-04 miércoles
$
```

```
[a2estudiante18@atcgrid ejer4]$ ls
HelloOMP2
[a2estudiante18@atcgrid ejer4]$ ls
HelloOMP2  script_helloomp.sh
[a2estudiante18@atcgrid ejer4]$ sbatch -p ac -n1 --cpus-per-task=12 --hint=nomultithread script_helloomp.sh
Submitted batch job 17727
[a2estudiante18@atcgrid ejer4]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
[a2estudiante18@atcgrid ejer4]$ ls
HelloOMP2  script_helloomp.sh  slurm-17727.out
[a2estudiante18@atcgrid ejer4]$ cat slurm-17727.out
Id. usuario del trabajo: a2estudiante18
Id. del trabajo: 17727
Nombre del trabajo especificado por usuario: helloOMP
Directorio de trabajo (en el que se ejecuta el script): /home/a2estudiante18/bp0/ejer4
Cola: ac
Nodo que ejecuta este trabajo:atcgrid
Nº de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgrid1
CPUs por nodo: 24

 1. Ejecución helloOMP2 una vez sin cambiar nº de threads (valor por defecto):

(10:!!!Hello(10:world!!!(8:!!!Hello(0:!!!Hello(8:world!!!(0:world!!!(11:!!!Hello(9:!!!Hello(9:world!!!(11:world!!!(6:!
!!Hello(4:!!!Hello(7:!!!Hello(5:!!!Hello(1:!!!Hello(5:world!!!(7:world!!!(1:world!!!(3:!!!Hello(3:world!!!(4:world!!!(
6:world!!!(2:!!!Hello(2:world!!!
 2. Ejecución helloOMP2 varias veces con distinto nº de threads:


  - Para 12 threads:
(0:!!!Hello(0:world!!!(8:!!!Hello(8:world!!!(1:!!!Hello(1:world!!!(4:!!!Hello(4:world!!!(10:!!!Hello(10:world!!!(2:!!!
Hello(2:world!!!(7:!!!Hello(7:world!!!(6:!!!Hello(6:world!!!(3:!!!Hello(3:world!!!(11:!!!Hello(11:world!!!(9:!!!Hello(
9:world!!!(5:!!!Hello(5:world!!!
  - Para 6 threads:
(2:!!!Hello(2:world!!!(0:!!!Hello(0:world!!!(4:!!!Hello(4:world!!!(1:!!!Hello(1:world!!!(3:!!!Hello(3:world!!!(5:!!!He
llo(5:world!!!
  - Para 3 threads:
(0:!!!Hello(0:world!!!(1:!!!Hello(1:world!!!(2:!!!Hello(2:world!!!
  - Para 1 threads:
(0:!!!Hello(0:world!!![a2estudiante18@atcgrid ejer4]$
```

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

RESPUESTA:

El nodo atcgrid1, rescatado de la información de cabecera del script “Nodos asignados al trabajo: atcgrid1.”

NOTA: Utilizar siempre con `sbatch` las opciones `-n1` y `--cpus-per-task`, `--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 `--cpus-per-task` y, para usar cores físicos y no lógicos, no olvide incluir `--hint=nomultithread`. Recordar que los `srun` dentro de un script heredan las opciones utilizadas 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:

```
[JoseLuisPedraza a2estudiante18@jos-GE62VR-7RF:~/Escritorio/practAC/BP0/ejer5] 2020-02-26 miércoles
$gcc  -O2  SumaVectoresC.c -o SumaVectores -lrt
SumaVectoresC.c: In function 'main':
SumaVectoresC.c:45:32: warning: format '%u' expects argument of type 'unsigned int', but argument 3 has type 'long uns
igned int' [-Wformat=]
   printf("Tama□o Vectores:%u (%u B)\n",N, sizeof(unsigned int));
                               ~^
                               %lu
[JoseLuisPedraza a2estudiante18@jos-GE62VR-7RF:~/Escritorio/practAC/BP0/ejer5] 2020-02-26 miércoles
$./SumaVectores 100
Tama□o Vectores:100 (4 B)
Tiempo:0.000000500      / Tama□o Vectores:100  / V1[0]+V2[0]=V3[0](10.000000+10.000000=20.000000) / / V1[99]+V2[99]=V
3[99](19.900000+0.100000=20.000000) /
[JoseLuisPedraza a2estudiante18@jos-GE62VR-7RF:~/Escritorio/practAC/BP0/ejer5] 2020-02-26 miércoles
$
```

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:
Contiene el tiempo de ejecución con precisión 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:

La información se devuelve en una estructura time:

Struct timespec {

time_t tv_sec; //segundos

long tv_nsec; //nanosegundos

};

El tipo de dato time_t es un tipo de dato de la biblioteca ISO-C definido para el almacenamiento de valores de tiempo del sistema. Estos valores se devuelven desde la función time() de la biblioteca estándar. Este tipo es un typedef definido en la cabecera <time.h>

(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 "clock_gettime()" devuelve el tiempo en el momento actual, por tanto, nos permite calcular el tiempo total transcurrido desde un instante inicial hasta un instante final, para lo cual restaremos los resultados Final-Inicial.

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. Obtener estos resultados usando scripts (partir del script que hay en el seminario). Debe haber una tabla para atcgrid 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 PC:

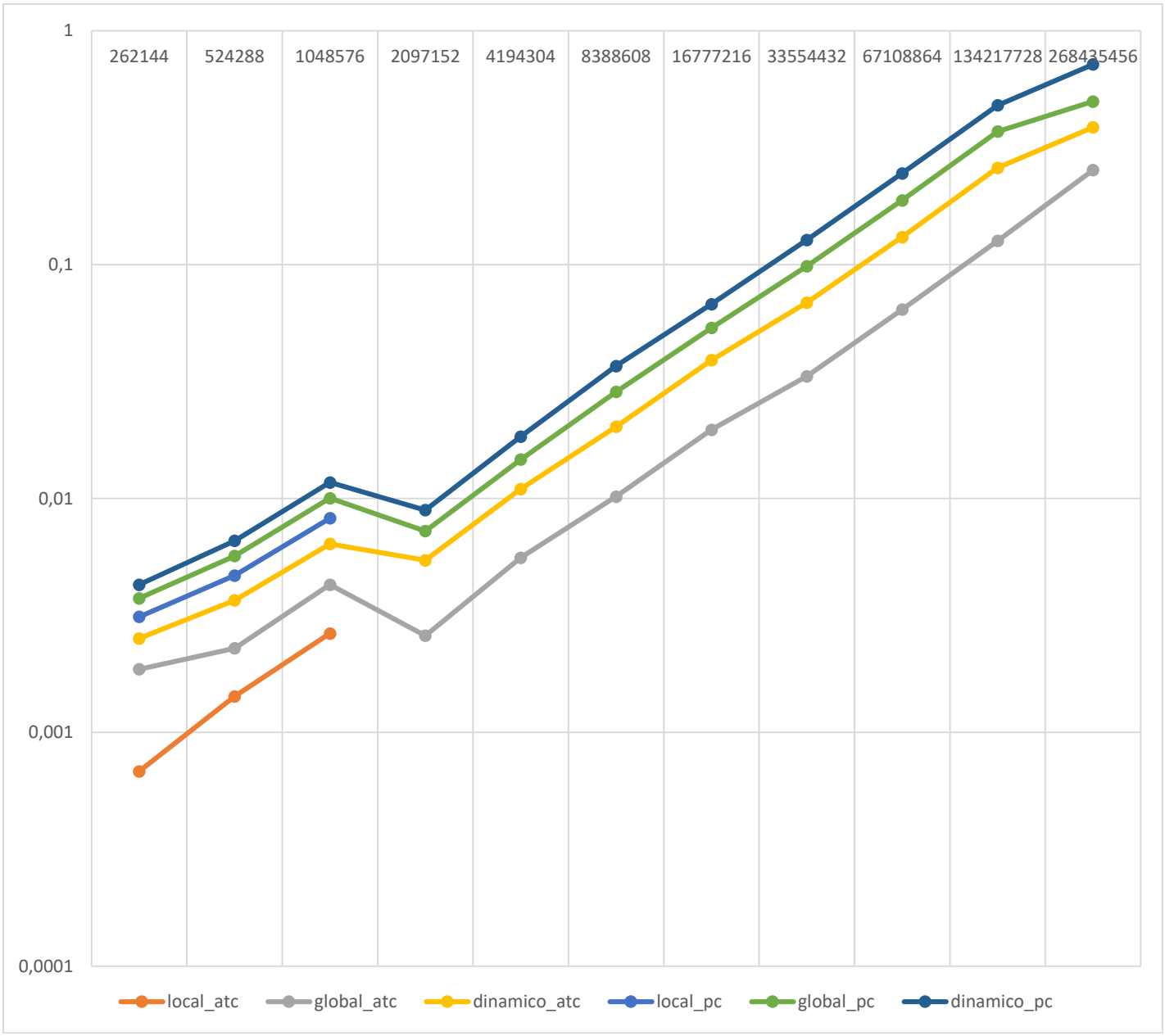
Tamano del vector	bytes	tiempo_local	tiempo_global	tiempo_dinamico
65536	262144	0.000596538	0.000626384	0.000537568
131072	524288	0.001012060	0.000993618	0.000917375
262144	1048576	0.001836437	0.001795008	0.001689816
524288	2097152		0.001817474	0.001664590
1048576	4194304		0.003699278	0.003716485
2097152	8388608		0.008314235	0.008267833
4194304	16777216		0.014575980	0.014130959
8388608	33554432		0.029748998	0.029011607
16777216	67108864		0.057166181	0.057237673
33554432	134217728		0.111418051	0.109179003
67108864	268435456		0.111650520	0.219743272

RESPUESTA ATCGRID:

Tamano del vector	bytes	tiempo_local	tiempo_global	tiempo_dinamico
65536	262144	0.000659816	0.001182027	0.000681923
131072	524288	0.001384444	0.000863874	0.001426812
262144	1048576	0.002117599	0.001634322	0.002650116
524288	2097152		0.002848739	0.002593980
1048576	4194304		0.005407102	0.005586908
2097152	8388608		0.010090349	0.010197140
4194304	16777216		0.019390890	0.019679993
8388608	33554432		0.035494579	0.033332012
16777216	67108864		0.066979665	0.064280167
33554432	134217728		0.132984371	0.126433645
67108864	268435456		0.133012327	0.253701666

1. Con ayuda de la hoja de cálculo representar **en una misma gráfica** los tiempos de ejecución obtenidos en atcgrid y en su PC para vectores locales, globales y dinámicos (eje y) en función del tamaño en bytes de un vector (por tanto, los valores de la segunda columna de la tabla, que están en escala logarítmica, deben estar en el eje x). Utilizar escala logarítmica en el eje de ordenadas (eje y). ¿Hay diferencias en los tiempos de ejecución?

RESPUESTA:



2. **(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:

```
[JoseLuisPedraza a2estudiante18@jos-GE62VR-7RF:~/Escritorio/practAC/BP0/ejer7/PC] 2020-03-04 miércoles
$./SumaVectores_local 262144
Tiempo:0.000932101       / Vectores:262144
[JoseLuisPedraza a2estudiante18@jos-GE62VR-7RF:~/Escritorio/practAC/BP0/ejer7/PC] 2020-03-04 miércoles
$./SumaVectores_local 524288
Violación de segmento (`core' generado)
[JoseLuisPedraza a2estudiante18@jos-GE62VR-7RF:~/Escritorio/practAC/BP0/ejer7/PC] 2020-03-04 miércoles
$
```

Dado que las variables locales se almacenan en la pila, esta solo puede almacenar vectores como muy grandes de 262144. Error en tiempo de ejecución, segmentation fault.

(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:

```
[JoseLuisPedraza a2estudiante18@jos-GE62VR-7RF:~/Escritorio/practAC/BP0/ejer7/PC] 2020-03-04 miércoles
$./SumaVectores_global 33554432
Tiempo:0.107752291       / Vectores:33554432
[JoseLuisPedraza a2estudiante18@jos-GE62VR-7RF:~/Escritorio/practAC/BP0/ejer7/PC] 2020-03-04 miércoles
$./SumaVectores_global 67108864
Tiempo:0.107038197       / Vectores:33554432
[JoseLuisPedraza a2estudiante18@jos-GE62VR-7RF:~/Escritorio/practAC/BP0/ejer7/PC] 2020-03-04 miércoles
$
```

No se obtiene error como tal, pero las dos últimas ejecuciones dan el mismo resultado, ya que nunca se llega a calcular la suma para tamaños de 67108864 ya que en el código fuente del programa está limitado. Si cambiásemos esta limitación para permitírselo, el programa no compilaría (las variables globales se almacenan en el heap) y daría error en tiempo de compilación.

(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:

Para vectores dinámicos no se obtiene ningún error.

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

RESPUESTA:

Máximo valor de N = 2^{32}-1 = 4294967295. (sizeof(unsigned int)) = 4B

(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:

Obtenemos el siguiente error debido a que el tamaño de la sección de datos se encuentra limitado para tamaños tan grandes (por lo que deberemos usar vectores dinámicos normalmente).
Los vectores globales se almacenan en el segmento de datos del programa, y están creando vectores de 4GB-1 número. Cada uno de esos números ocupa 8 bytes dado que son de tipo double, por lo que cada vector necesita (2^32-1)*8.
El error no se produce porque en memoria no “quepa”, sino porque los compiladores de GNU usan por defecto un modelo de memoria Small, el cual solo permite que el segmento de datos ocupe 2GB.

```
[JoseLuisPedraza a2estudiante18@jos-GE62VR-7RF:~/Escritorio/practAC/BP0/ejer7/3b] 2020-03-04 miércoles
$gcc  -O2  SumaVectoresC.c -o SumaVectores -lrt
/tmp/ccrX84tX.o: En la función `main':
SumaVectoresC.c:(.text.startup+0x5a): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v2' definido
 en la sección COMMON en /tmp/ccrX84tX.o
SumaVectoresC.c:(.text.startup+0xb2): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v3' definido
 en la sección COMMON en /tmp/ccrX84tX.o
collect2: error: ld returned 1 exit status
[JoseLuisPedraza a2estudiante18@jos-GE62VR-7RF:~/Escritorio/practAC/BP0/ejer7/3b] 2020-03-04 miércoles
$
```

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

```
/* SumaVectoresC.c
 Suma de dos vectores: v3 = v1 + v2

 Para compilar usar (-lrt: real time library, no todas las versiones de gcc necesitan que se incluya -lrt):
         gcc  -O2 SumaVectores.c -o SumaVectores -lrt
         gcc  -O2 -S SumaVectores.c -lrt   //para generar el código ensamblador

 Para ejecutar use: SumaVectoresC longitud
*/

#include <stdlib.h> // biblioteca con funciones atoi(), malloc() y free()
#include <stdio.h> // biblioteca donde se encuentra la función printf()
#include <time.h>       // biblioteca donde se encuentra la función clock_gettime()

//Sólo puede estar definida una de las tres constantes VECTOR_ (sólo uno de los ...
//tres defines siguientes puede estar descomentado):
//#define VECTOR_LOCAL    // descomentar para que los vectores sean variables ...
                          // locales (si se supera el tamaño de la pila se ...
                          // generará el error "Violación de Segmento")
//#define VECTOR_GLOBAL// descomentar para que los vectores sean variables ...
                          // globales (su longitud no estará limitada por el ...
                          // tamaño de la pila del programa)
#define VECTOR_DYNAMIC    // descomentar para que los vectores sean variables ...
                          // dinámicas (memoria reutilizable durante la ejecución)
#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
  for(i=0; i<N; i++){
     v1[i] = N*0.1+i*0.1; v2[i] = N*0.1-i*0.1; //los valores dependen de N
  }

  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;
}
```