

Arquitectura de Computadores (AC)

Cuaderno de prácticas.

Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos): Mario Garcia Marquez

Grupo de prácticas y profesor de prácticas: Grupo 1 - Maribel

Fecha de entrega:

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:

```
[MarioGarciaMarquez elestudiente9@atcgrid:~] 2021-03-04 jueves
$srunc -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 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_t
                        sc 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
```

```
[MarioGarciaMarquez elestudiente9@atcgrid:~] 2021-03-04 jueves
$srunc -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:                 1191.503
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 nonst
                        op_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid dca ss
                        e4_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_ppin intel_pt ssbd mba ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi flexpriority ept vpid fsgsbas
                        e tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm cqm mpx rdt_a avx512f avx512dq rdseed adx smap clflushopt clwb avx
                        512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local dtherm ida arat pln pt
                        s pku ospke avx512_vnni md_clear spec_ctrl intel_stibp flush_l1d arch_capabilities
```

```

> ls
BP0_GarciaMarquezMario_Y_2020_21.odt  ejer3  ejer2
ac/bp0 on ↵ main [?]
> lscpu
Architecture:                x86_64
CPU op-mode(s):              32-bit, 64-bit
Byte Order:                   Little Endian
Address sizes:                43 bits physical, 48 bits virtual
CPU(s):                       16
On-line CPU(s) list:         0-15
Thread(s) per core:          2
Core(s) per socket:          8
Socket(s):                    1
NUMA node(s):                1
Vendor ID:                    AuthenticAMD
CPU family:                   23
Model:                        8
Model name:                   AMD Ryzen 7 2700 Eight-Core Processor
Stepping:                     2
Frequency boost:              enabled
CPU MHz:                      2800.000
CPU max MHz:                  3200.0000
CPU min MHz:                  1550.0000
BogoMIPS:                     6389.44
Virtualization:               AMD-V
L1d cache:                   256 KiB
L1i cache:                   512 KiB
L2 cache:                     4 MiB
L3 cache:                     16 MiB
NUMA node0 CPU(s):           0-15
Vulnerability Itlb multihit:  Not affected
Vulnerability L1tf:           Not affected
Vulnerability Mds:            Not affected
Vulnerability Meltdown:       Not affected

```

(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:

Atcgrid4 cuenta con un total de 32 cores físicos y gracias a la tecnología hyperthreading de intel con 64 lógicos. Atcgrid1, 2 y 3 cuentan en cambio con 12 físicos y 24 lógicos gracias a la misma tecnología. Mi PC cuenta con un ryzen 7 2700 por lo que suma 8 cores físicos y 16 cores lógicos gracias a la tecnología de multithreading de AMD. Esto se puede ver en las capturas de lscpu fijandonos en los campos Core(s) per socket, Socket(s) y threads per core. CPU(s) marca el total de cores lógicos.

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:

```

Alacritty
ac/bp0/ejer2 on main [?]
> ls ../
BP0_GarciaMarquezMario_Y_2020_21.odt  ejer1  ejer2
ac/bp0/ejer2 on main [?]
> gcc -fopenmp -O2 -o HelloOMP HelloOMP.c
ac/bp0/ejer2 on main [?]
> ./HelloOMP
(3:!!!Hello world!!!)(5:!!!Hello world!!!)(14:!!!Hello world!!!)(0:!!!Hello wor
ld!!!)(7:!!!Hello world!!!)(11:!!!Hello world!!!)(6:!!!Hello world!!!)(13:!!!He
llo world!!!)(9:!!!Hello world!!!)(10:!!!Hello world!!!)(15:!!!Hello world!!!)(
2:!!!Hello world!!!)(12:!!!Hello world!!!)(4:!!!Hello world!!!)(8:!!!Hello worl
d!!!)(1:!!!Hello world!!!)ac/bp0/ejer2 on main [?]
> 

```

(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 por pantalla tantos Hello World como hilos tiene nuestro procesador, en mi caso 16 veces.

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:

```

[MarioGarciaMarquez e1estudiante9@atcgrid:~/bp0/ejer2] 2021-03-04 jueves
$ srun -pac -Aac -n1 -c12 --hint=nomultithread HelloOMP
(1:!!!Hello world!!!)(11:!!!Hello world!!!)(2:!!!Hello world!!!)(10:!!!Hello world!!!)(7:!!!Hello world!!!)(0:!!!Hello w
orld!!!)(8:!!!Hello world!!!)(6:!!!Hello world!!!)(3:!!!Hello world!!!)(9:!!!Hello world!!!)(4:!!!Hello world!!!)(5:!!!H
ello world!!!)[MarioGarciaMarquez e1estudiante9@atcgrid:~/bp0/ejer2] 2021-03-04 jueves
$ 

```

(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:

```

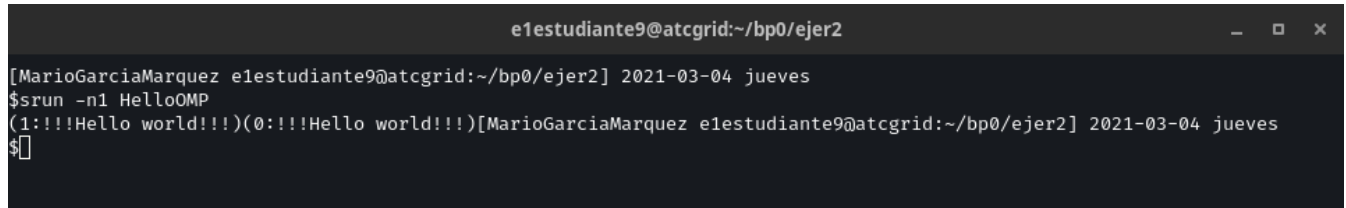
e1estudiante9@atcgrid:~/bp0/ejer2
[MarioGarciaMarquez e1estudiante9@atcgrid:~/bp0/ejer2] 2021-03-04 jueves
$ srun -pac -Aac -n1 -c24 HelloOMP
(8:!!!Hello world!!!)(0:!!!Hello world!!!)(6:!!!Hello world!!!)(14:!!!Hello world!!!)(20:!!!Hello world!!!)(11:!!!Hello
world!!!)(9:!!!Hello world!!!)(22:!!!Hello world!!!)(17:!!!Hello world!!!)(23:!!!Hello world!!!)(18:!!!Hello world!!!)(5
:!!!Hello world!!!)(19:!!!Hello world!!!)(15:!!!Hello world!!!)(7:!!!Hello world!!!)(16:!!!Hello world!!!)(12:!!!Hello w
orld!!!)(4:!!!Hello world!!!)(10:!!!Hello world!!!)(2:!!!Hello world!!!)(3:!!!Hello world!!!)(13:!!!Hello world!!!)(1: !
!Hello world!!!)(21:!!!Hello world!!!)[MarioGarciaMarquez e1estudiante9@atcgrid:~/bp0/ejer2] 2021-03-04 jueves
$ 

```

(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:



```
e1estudiante9@atcgrid:~/bp0/ejer2
[MarioGarciaMarquez e1estudiante9@atcgrid:~/bp0/ejer2] 2021-03-04 jueves
$ srun -n1 HelloOMP
(1:!!!Hello world!!!)(0:!!!Hello world!!!)[MarioGarciaMarquez e1estudiante9@atcgrid:~/bp0/ejer2] 2021-03-04 jueves
$
```

Se ejecuta en la cola AC que es la cola por defecto.

(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 -pac4 -Aac -n1 -c32 --hint=nomultithread HelloOMP`

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

(a) Utilizar: `sbatch -pac -n1 -c12 --hint=nomultithread script_helloomp.sh`. Adjuntar capturas de pantalla que muestren el nuevo código, la compilación, el envío a la cola de la ejecución y el resultado de esta ejecución tal y como la devuelve el gestor de colas.

RESPUESTA:

```

e1estudiante9@atcgrid:~/bp0/ejer4
[MarioGarciaMarquez e1estudiante9@atcgrid:~/bp0/ejer4] 2021-03-04 jueves
$cat HelloOMP2.c
#include <omp.h>
#include <stdio.h>
int main(void) {
#pragma omp parallel
    printf("(%d:!!!Hello )", omp_get_thread_num());
    printf("(%d: World!!!)", omp_get_thread_num());
    return (0);
}
[MarioGarciaMarquez e1estudiante9@atcgrid:~/bp0/ejer4] 2021-03-04 jueves
$gcc -O2 -fopenmp -o HelloOMP HelloOMP2.c
[MarioGarciaMarquez e1estudiante9@atcgrid:~/bp0/ejer4] 2021-03-04 jueves
$
[MarioGarciaMarquez e1estudiante9@atcgrid:~/bp0/ejer4] 2021-03-04 jueves
$sbatch -pac -n1 -c12 --hint=nomultithreading script_helloomp.sh
sbatch: error: unrecognized --hint argument "nomultithreading", see --hint=help
[MarioGarciaMarquez e1estudiante9@atcgrid:~/bp0/ejer4] 2021-03-04 jueves
$sbatch -pac -n1 -c12 --hint=nomultithread script_helloomp.sh
Submitted batch job 58942
[MarioGarciaMarquez e1estudiante9@atcgrid:~/bp0/ejer4] 2021-03-04 jueves
$ls
HelloOMP HelloOMP2.c script_helloomp.sh slurm-58942.out
[MarioGarciaMarquez e1estudiante9@atcgrid:~/bp0/ejer4] 2021-03-04 jueves
$cat slurm-58942.out
Id. usuario del trabajo: e1estudiante9
Id. del trabajo: 58942
Nombre del trabajo especificado por usuario: helloOMP
Directorio de trabajo (en el que se ejecuta el script): /home/e1estudiante9/bp0/ejer4
Cola: ac
Nodo que ejecuta este trabajo:atcgrid.ugr.es
Nº de nodos asignadosal trabajo: 1
Nodos asignados al trabajo: atcgrid1
CPUs por nodo: 24

1. Ejecución helloOMP una vez sin cambiar nº de threads(valor por defecto):
/var/spool/slurmd/job58942/slurm_script: línea 21: srun./HelloOMP: No existe el fichero o el directorio

2. Ejecución helloOMPvarias veces con distinto nº de threads:

- Para 12 threads:
(9:!!!Hello )(7:!!!Hello )(10:!!!Hello )(2:!!!Hello )(1:!!!Hello )(6:!!!Hello )(0:!!!Hello )(4:!!!Hello )(3:!!!Hello )(5:!!!Hello )(11:!!!Hello )(8:!!!Hello )(0: World!!!)
- Para 6 threads:
(3:!!!Hello )(1:!!!Hello )(2:!!!Hello )(4:!!!Hello )(0:!!!Hello )(5:!!!Hello )(0: World!!!)
- Para 3 threads:
(1:!!!Hello )(2:!!!Hello )(0:!!!Hello )(0: World!!!)
- Para 1 threads:
(0:!!!Hello )(0: World!!!)[MarioGarciaMarquez e1estudiante9@atcgrid:~/bp0/ejer4] 2021-03-04 jueves
$

```

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

RESPUESTA:

Ha sido el atcgrid1 como nos muestra la script, que imprime por pantalla una variable de entorno de slurm: el sistema que gestiona las colas del atcgrid.

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:

```

ac/bp0/ejer5 on p main [?] took 7s
> ls ..
BP0_GarciaMarquezMario_Y_2020_21.odt
ac/bp0/ejer5 on p main [?]
> gcc -o listado1 -fopenmp -O2 listado1.c
ac/bp0/ejer5 on p main [?]
> ./listado1 1000
Tiempo(seg.):0.000003146 / Tamaño Vectores:1000 / V1[0]+V2[0]=V3[0](1.914548+0.691679=2.606227) / V1[999]+V2[999]=V3[999](4.297912+3.434320=7.732232) /
ac/bp0/ejer5 on p main [?]
> scro
  
```

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 diferencia de tiempo entre el tiempo en el que se inicia la suma y el tiempo en el que esta acaba.

(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:

Guarda esto en una estructura llamada `timespec`, esta estructura cuenta con un campo para los segundos y otro para los nanosegundos. Esta estructura es similar a la conocida estructura `timeval` pero usando nanosegundos en lugar de microsegundos.

(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:

Guarda la información del reloj pasado como primer parámetro en la estructura pasada como segundo parámetro. En el caso del programa usando `CLOCK_REALTIME` estamos pasando el reloj del sistema, que mide los segundos desde 1970, según el estándar POSIX 1003.4.

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:

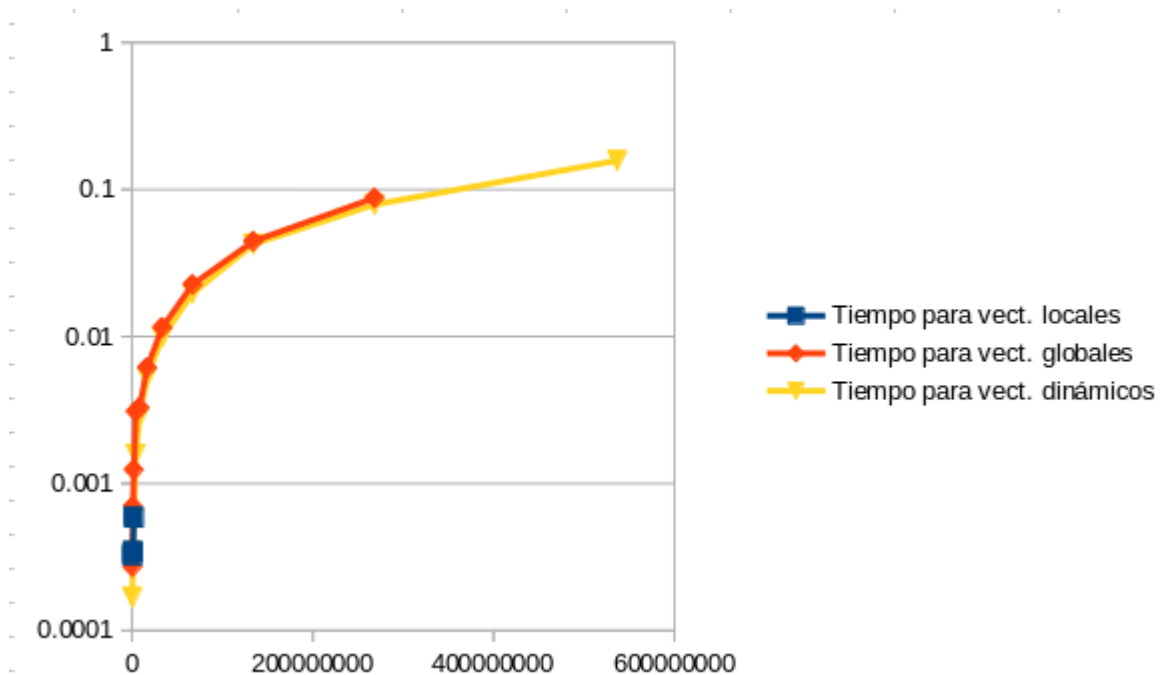
Tabla 1. Copiar la tabla de la hoja de cálculo utilizada (He copiado la del Xeon)

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000474358	0.000542867	0.000482667
131072	1048576	0.000781288	0.000506526	0.000960877
262144	2097152	0.000623241	0.001467835	0.001722354
524288	4194304	seg fault	0.002446578	0.002681677
1048576	8388608	seg fault	0.004968636	0.005051195
2097152	16777216	seg fault	0.008872473	0.008759515
4194304	33554432	seg fault	0.016841284	0.01660189
8388608	67108864	seg fault	0.033114103	0.032637403
16777216	134217728	seg fault	0.067617241	0.064645554
33554432	268435456	seg fault	0.132110155	0.127708037
67108864	536870912	seg fault		0.244564375

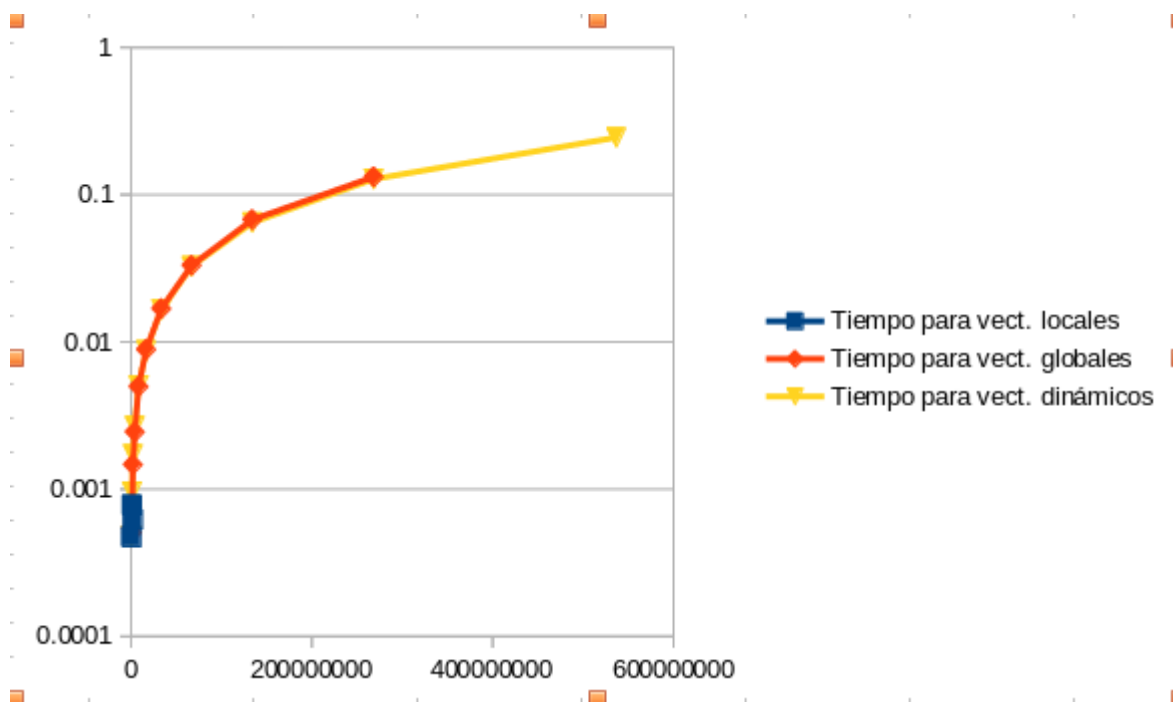
1. Con ayuda de la hoja de cálculo representar **en una misma gráfica** los tiempos de ejecución obtenidos en atcgrid y en su PC para vectores locales, globales y dinámicos (eje y) en función del tamaño en bytes de un vector (por tanto, los valores de la segunda columna de la tabla, que están en escala 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:

PC:



ATC:



Vemos que hay diferencia en tiempo de ejecución siendo mi PC más rápido el ATCGRID, sin embargo los tiempos de ambos evolucionan igualmente.

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:

Efectivamente hay error en los vectores locales, esto se debe a que estos se encuentran en el stack, de espacio reducido, por lo que al reservar tamaños grandes estos no caben en el stack.

```

Alacrity
Mar 6 12:26

ac/bp0/ejer7 on p main [?]
> ls ..
BP0_GarciaMarquezMario_Y_2020_21.odt
ac/bp0/ejer7 on p main [?]
> bash ejecutar.sh
Tiempo(seg.):0.000269151 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](1.008376+0.596351=1.604727) // V1[65535]+V2[65535]=V3[65535](2.828119+2.056210=4.884329) /
Tiempo(seg.):0.000624983 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](1.008376+0.596351=1.604727) // V1[131071]+V2[131071]=V3[131071](1.546003+1.296882=2.842885) /
Tiempo(seg.):0.001158104 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](1.008376+0.596351=1.604727) // V1[262143]+V2[262143]=V3[262143](0.325091+1.038238=1.363329) /
ejecutar.sh: line 4: 19876 Segmentation fault (core dumped) ./listado1 $i
ejecutar.sh: line 4: 19879 Segmentation fault (core dumped) ./listado1 $i
ejecutar.sh: line 4: 19889 Segmentation fault (core dumped) ./listado1 $i
ejecutar.sh: line 4: 19897 Segmentation fault (core dumped) ./listado1 $i
ejecutar.sh: line 4: 19905 Segmentation fault (core dumped) ./listado1 $i
ejecutar.sh: line 4: 19913 Segmentation fault (core dumped) ./listado1 $i
ejecutar.sh: line 4: 19922 Segmentation fault (core dumped) ./listado1 $i
ejecutar.sh: line 4: 19931 Segmentation fault (core dumped) ./listado1 $i
ac/bp0/ejer7 on p main [?]
> scrot

```

(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, sin embargo el tamaño de estos vectores esta capado por una variable global. Si se supera este tamaño entonces se escogera ese maximo como tamaño.

```

Alacrity
Mar 6 12:26

ac/bp0/ejer7 on p main [?]
> ls ..
BP0_GarciaMarquezMario_Y_2020_21.odt
ac/bp0/ejer7 on p main [?]
> bash ejecutar.sh
Tiempo(seg.):0.000191797 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](1.418429+1.317325=2.735754) // V1[65535]+V2[65535]=V3[65535](0.164018+0.635036=0.799054) /
Tiempo(seg.):0.000604035 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](1.418429+1.317325=2.735754) // V1[131071]+V2[131071]=V3[131071](0.068548+0.709343=0.777891) /
Tiempo(seg.):0.001130344 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](1.418429+1.317325=2.735754) // V1[262143]+V2[262143]=V3[262143](0.028397+0.891075=0.919472) /
Tiempo(seg.):0.002721104 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](1.418429+1.317325=2.735754) // V1[524287]+V2[524287]=V3[524287](0.151830+1.464664=1.616494) /
Tiempo(seg.):0.003083489 / Tamaño Vectores:1048576 / V1[0]+V2[0]=V3[0](1.418429+1.317325=2.735754) // V1[1048575]+V2[1048575]=V3[1048575](0.823315+0.694095=1.517410) /
Tiempo(seg.):0.006307872 / Tamaño Vectores:2097152 / V1[0]+V2[0]=V3[0](1.418429+1.317325=2.735754) // V1[2097151]+V2[2097151]=V3[2097151](0.828276+0.382402=1.210677) /
Tiempo(seg.):0.012226569 / Tamaño Vectores:4194304 / V1[0]+V2[0]=V3[0](6.565395+0.357695=6.923090) // V1[4194303]+V2[4194303]=V3[4194303](1.021318+0.047617=1.068936) /
Tiempo(seg.):0.024794473 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](6.565395+0.357695=6.923090) // V1[8388607]+V2[8388607]=V3[8388607](25.359608+1.648357=27.007965) /
Tiempo(seg.):0.047295194 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](6.565395+0.357695=6.923090) // V1[16777215]+V2[16777215]=V3[16777215](1.086596+0.369156=1.455752) /
Tiempo(seg.):0.092953074 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](6.565395+0.357695=6.923090) // V1[33554431]+V2[33554431]=V3[33554431](0.436357+0.122373=0.558730) /
Tiempo(seg.):0.092915955 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](44.024233+0.071796=44.096028) // V1[33554431]+V2[33554431]=V3[33554431](0.142770+1.734834=1.877604) /
ac/bp0/ejer7 on p main [?] took 2s
> scrot

```

(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, no hay ningún problema ya que los vectores dinámicos se ubican en el heap donde hay suficiente espacio para estos para existir.

```

ac/bp0/ejer7 on 12 main [?]
> ls ..
BP0_GarciaMarquezMario_Y_2020_21.odt
ac/bp0/ejer7 on 12 main [?]
> bash ejecutar.sh
Tiempo(seg.):0.000180256 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](0.926331+0.854177=1.780508) / V1[65535]+V2[65535]=V3[65535](0.231554+0.840560=1.072114) /
Tiempo(seg.):0.000351424 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](0.926331+0.854177=1.780508) / V1[131071]+V2[131071]=V3[131071](0.731088+0.203877=0.934966) /
Tiempo(seg.):0.000685076 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](0.926331+0.854177=1.780508) / V1[262143]+V2[262143]=V3[262143](0.115172+1.469528=1.584700) /
Tiempo(seg.):0.002198775 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](0.926331+0.854177=1.780508) / V1[524287]+V2[524287]=V3[524287](0.525544+0.560689=1.086233) /
Tiempo(seg.):0.002502191 / Tamaño Vectores:1048576 / V1[0]+V2[0]=V3[0](0.926331+0.854177=1.780508) / V1[1048575]+V2[1048575]=V3[1048575](6.370097+0.168569=6.538666) /
Tiempo(seg.):0.004976189 / Tamaño Vectores:2097152 / V1[0]+V2[0]=V3[0](0.926331+0.854177=1.780508) / V1[2097151]+V2[2097151]=V3[2097151](0.441767+0.561991=1.003758) /
Tiempo(seg.):0.012605420 / Tamaño Vectores:4194304 / V1[0]+V2[0]=V3[0](0.926331+0.854177=1.780508) / V1[4194303]+V2[4194303]=V3[4194303](0.095549+0.687164=0.782713) /
Tiempo(seg.):0.020977663 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](0.888777+84.903772=85.792549) / V1[8388607]+V2[8388607]=V3[8388607](0.532660+1.646522=2.179182) /
Tiempo(seg.):0.044401694 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](0.888777+84.903772=85.792549) / V1[16777215]+V2[16777215]=V3[16777215](1.080079+0.129790=1.209869) /
Tiempo(seg.):0.081490937 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](0.888777+84.903772=85.792549) / V1[33554431]+V2[33554431]=V3[33554431](0.106713+3.425619=3.532632) /
Tiempo(seg.):0.162272385 / Tamaño Vectores:67108864 / V1[0]+V2[0]=V3[0](1.633740+2.106174=3.739914) / V1[67108863]+V2[67108863]=V3[67108863](10.103569+2.718406=12.821975) /
ac/bp0/ejer7 on 12 main [?] took 3s
> scrot

```

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

RESPUESTA:

N se guarda como int(no unsigned) por lo que su maximo valor es 2147483647, ya que usa 4 bytes para representar un numero.

(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 codigo no compila, ya que da error por ser un tamaño demasiado grande para poder ubicar esos vectores en la memoria.

```

ac/bp0/ejer7 on P main [?]
> ls ..
BP0_GarciaMarquezMario_Y_2020_21.odt
ac/bp0/ejer7 on P main [?]
> gcc ../ejers/listadol.c -o listadol -O2 -fopenmp
/tmp/ccZ8hP0Z.o: in function 'main':
listadol.c:(.text.startup+0x61): relocation truncated to fit: R_X86_64_PC32 against symbol `v1' defined in .bss section in /tmp/ccZ8hP0Z.o
listadol.c:(.text.startup+0x68): relocation truncated to fit: R_X86_64_PC32 against symbol `v2' defined in .bss section in /tmp/ccZ8hP0Z.o
listadol.c:(.text.startup+0x1a3): relocation truncated to fit: R_X86_64_PC32 against symbol `v1' defined in .bss section in /tmp/ccZ8hP0Z.o
listadol.c:(.text.startup+0x1aa): relocation truncated to fit: R_X86_64_PC32 against symbol `v2' defined in .bss section in /tmp/ccZ8hP0Z.o
collect2: error: ld returned 1 exit status
ac/bp0/ejer7 on P main [?]
> scrot -s
Key was pressed, aborting shot
glib error: no image grabbed
ac/bp0/ejer7 on P main [?]
> scrot

```

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));v

//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;
}

```