

BP0.pdf



PruebaAlien



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.





**KEEP
CALM
AND
ESTUDIA
UN POQUITO**

2º curso / 2º cuatr.

Grado Ingeniería
Informática

Arquitectura de Computadores (AC)

Cuaderno de prácticas.

Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos):

Grupo de prácticas y profesor de prácticas:

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 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

```

ruben:~$ 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) i5-8300H CPU @ 2.30GHz
Revisión:                   10
CPU MHz:                    1300.543
CPU MHz máx.:               2300,0000
CPU MHz mín.:               800,0000
BogoMIPS:                   4599.93
Virtualización:             VT-x
Caché L1d:                  32K
Caché L1i:                  32K
Caché L2:                   256K
Caché L3:                   8192K
CPU(s) del nodo NUMA 0:      0-7
Indicadores:                fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat ps
e36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_pe
rfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmpf perf_pni pclmulqdq dtes64 monitor ds_cpl v
mx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xs
ave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb invpcid_single ptl ssbd ibrs ibpb stibp tpr
_shadow vnmi flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid mpx rdseed
adx smap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify hwp_a
ct_window hwp_epp md_clear flush_l1d

```

ATCGRID

```

[b4estudiante10@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:                1599.822
CPU max MHz:            3900,0000
CPU min MHz:            800,0000
BogoMIPS:                7008.00
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 fx
sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology non
stop_tsc aperfmpf eagerfpn pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse
4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch epb intel_pt ssbd
ibrs ibpb stibp tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm mpx
rdseed adx smap clflushopt xsaveopt xsavec xgetbv1 dtherm ida arat pln pts hwp hwp_notify hwp_act_window hwp_epp md_cl
ear sspec ctrl intel stibn flush_l1d

```

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

RESPUESTA: son 4 cores físicos y 8 cores lógicos en el PC y en el ATCGRID son 4 cores físicos y 8 cores lógicos, por que según la consola tengo 4 nucleos por socket y como solo tengo una CPU tengo 4 físicos y 8 cores lógicos por que según dice la consola tengo 2 hilos de procesamiento por nucleo $2 * 4$ (nucleos) = 8 cores 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:

```
ruben:~/Escritorio$ gcc -O2 -fopenmp -o hello_omp HelloOMP.c
ruben:~/Escritorio$ ./hello_omp
(1):;;;HOLA MUNDO!!!
(0):;;;HOLA MUNDO!!!
(6):;;;HOLA MUNDO!!!
(4):;;;HOLA MUNDO!!!
(2):;;;HOLA MUNDO!!!
(5):;;;HOLA MUNDO!!!
(3):;;;HOLA MUNDO!!!
(7):;;;HOLA MUNDO!!!
```

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

RESPUESTA: Lanza del 0 al 7 hebras, es decir 8 hebras que ejecutan en paralelo que es el n.^o máximo que puede en el PC, puesto que he visto anteriormente cuantos son con lscpu.

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:

```
[b4estudiante10@atcgrid ~]$ srun -p ac -n1 --cpus-per-task=12 --hint=nomultithread HelloOMP
(8):;;;HOLA MUNDO!!!
(4):;;;HOLA MUNDO!!!
(1):;;;HOLA MUNDO!!!
(0):;;;HOLA MUNDO!!!
(2):;;;HOLA MUNDO!!!
(3):;;;HOLA MUNDO!!!
(7):;;;HOLA MUNDO!!!
(10):;;;HOLA MUNDO!!!
(9):;;;HOLA MUNDO!!!
(11):;;;HOLA MUNDO!!!
(6):;;;HOLA MUNDO!!!
(5):;;;HOLA MUNDO!!!
```

Nombre de archivo	Tamaño de archivo	Última modificación	P
.cache	20/02/19 12:41:42	dr	
.config	20/02/19 12:41:42	dr	
.mozilla	09/11/18 09:00:48	dr	
.ssh	21/01/20 08:48:21	dr	
Xauthority	53 B	21/02/20 09:18:07	rv

(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:

```
b4estudiante10@atcgrid:~$ srun -p ac -n1 --cpus-per-task=24 HelloOMP
(21):!!!HOLA MUNDO!!!
(19):!!!HOLA MUNDO!!!
(3):!!!HOLA MUNDO!!!
(1):!!!HOLA MUNDO!!!
(11):!!!HOLA MUNDO!!!
(12):!!!HOLA MUNDO!!!
(18):!!!HOLA MUNDO!!!
(0):!!!HOLA MUNDO!!!
(5):!!!HOLA MUNDO!!!
(7):!!!HOLA MUNDO!!!
(14):!!!HOLA MUNDO!!!
(8):!!!HOLA MUNDO!!!
(10):!!!HOLA MUNDO!!!
(20):!!!HOLA MUNDO!!!
(6):!!!HOLA MUNDO!!!
(2):!!!HOLA MUNDO!!!
(9):!!!HOLA MUNDO!!!
(17):!!!HOLA MUNDO!!!
(23):!!!HOLA MUNDO!!!
(4):!!!HOLA MUNDO!!!
(13):!!!HOLA MUNDO!!!
(15):!!!HOLA MUNDO!!!
(22):!!!HOLA MUNDO!!!
(16):!!!HOLA MUNDO!!!
[b4estudiante10@atcgrid ~]$
```

(c) `srun -p ac -n1 HelloOMP`

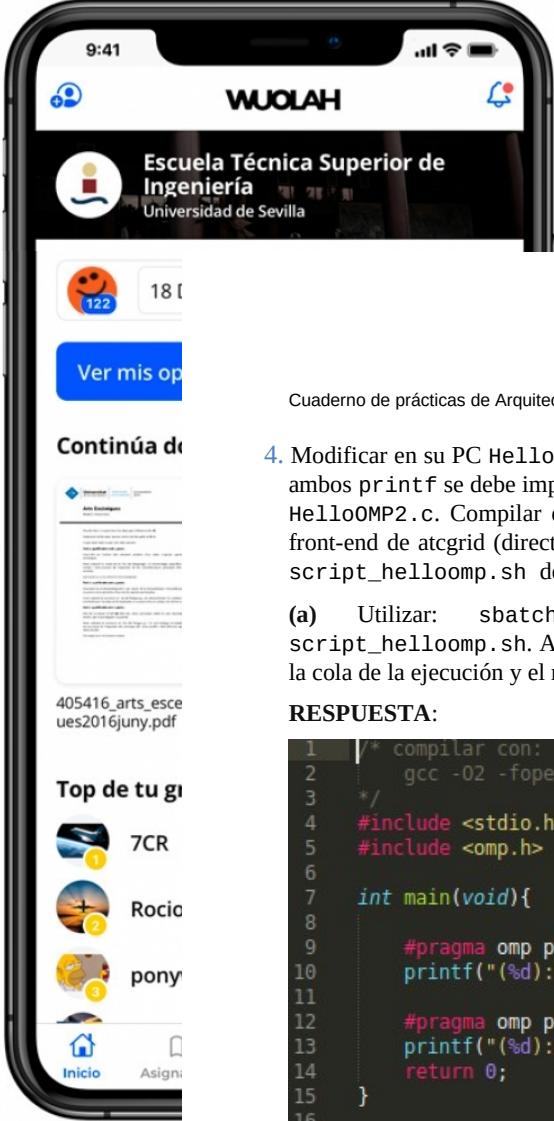
Adjuntar capturas de pantalla que muestren el envío a la cola de la ejecución y el resultado de esta ejecución tal y como la devuelve el gestor de colas.

RESPUESTA:

```
b4estudiante10@atcgrid:~$ srun -p ac -n1 HelloOMP
(0):!!!HOLA MUNDO!!!
(1):!!!HOLA MUNDO!!!
```

(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)?

`srun -p ac -n1 --cpus-per-task=12 --hint=nomultithread HelloOMP`



Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

Available on the
App Store

GET IT ON
Google Play

Cuaderno de prácticas de Arquitectura de Computadores, Grado en Ingeniería Informática

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 HelloOMP HelloOMP.c  
3   */  
4   #include <stdio.h>  
5   #include <omp.h>  
6  
7   int main(void){  
8  
9     #pragma omp parallel  
10    printf("(%d):!!!HOLA MUNDO!!!\n",omp_get_thread_num());  
11  
12    #pragma omp parallel  
13    printf("(%d):!!!MUNDO!!!\n",omp_get_thread_num());  
14    return 0;  
15  }  
16
```

```
ruben:~/Escritorio/AC/BP0/ejer4$ gcc -O2 -fopenmp -o HelloOMP2 HelloOMP2.c  
ruben:~/Escritorio/AC/BP0/ejer4$ ./HelloOMP2  
(7):!!!HOLA MUNDO!!!  
(1):!!!HOLA MUNDO!!!  
(5):!!!HOLA MUNDO!!!  
(4):!!!HOLA MUNDO!!!  
(3):!!!HOLA MUNDO!!!  
(2):!!!HOLA MUNDO!!!  
(0):!!!HOLA MUNDO!!!  
(6):!!!HOLA MUNDO!!!  
(0):!!!MUNDO!!! (MP-varias veces con distintos nº de threads)  
(2):!!!MUNDO!!!  
(7):!!!MUNDO!!!  
(4):!!!MUNDO!!!  
(1):!!!MUNDO!!!  
(5):!!!MUNDO!!!  
(6):!!!MUNDO!!!  
(3):!!!MUNDO!!!
```

```
[b4estudiante10@atcgrid ejer4]$ sbatch -p ac -n1 --cpus-per-task=12 --hint=nomultithread script_helloomp.sh
Submitted batch job 8542
[b4estudiante10@atcgrid ejer4]$ ls
HelloOMP2 script_helloomp.sh slurm-8535.out slurm-8542.out
[b4estudiante10@atcgrid ejer4]$ cat s
script_helloomp.sh slurm-8535.out slurm-8542.out
[b4estudiante10@atcgrid ejer4]$ cat slurm-8542.out
Id. usuario del trabajo: b4estudiante10
Id. del trabajo: 8542
Nombre del trabajo especificado por usuario: helloOMP
Directorio de trabajo (en el que se ejecuta el script): /home/b4estudiante10/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 helloOMP una vez sin cambiar nº de threads (valor por defecto):
(2):;;;HOLA MUNDO!!!
(0):;;;HOLA MUNDO!!!
(6):;;;HOLA MUNDO!!!
(4):;;;HOLA MUNDO!!!
(1):;;;HOLA MUNDO!!!
(11):;;;HOLA MUNDO!!!
(8):;;;HOLA MUNDO!!!
(7):;;;HOLA MUNDO!!!
(9):;;;HOLA MUNDO!!!
(3):;;;HOLA MUNDO!!!
(10):;;;HOLA MUNDO!!!
(5):;;;HOLA MUNDO!!!
(2):;;;MUNDO!!!
(4):;;;MUNDO!!!
(3):;;;MUNDO!!!
(7):;;;MUNDO!!!
(5):;;;MUNDO!!!
(9):;;;MUNDO!!!
(8):;;;MUNDO!!!
(0):;;;MUNDO!!!
(10):;;;MUNDO!!!
(6):;;;MUNDO!!!
(1):;;;MUNDO!!!
(11):;;;MUNDO!!!
```

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

RESPUESTA: Es el nodo 1.

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

- 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: (Tuve que ponerle `lu` por que no era compatible `%u`)

```
ruben:~/Escritorio$ gcc -O2 SumaVectoresC.c -o SumaVectoresC -lrt VECTOR_GLOBAL
SumaVectoresC.c: In function ‘main’:
SumaVectoresC.c:45:32: warning: format ‘%u’ expects argument of type ‘unsigned int’, but argument 3 has type ‘long unsigned int’ [-Wformat=]
    printf("Tamaño Vectores:%u (%u B)\n",N, sizeof(unsigned int));
                           ~^
                           %lu
```

```

ruben:~/Escritorio$ gcc -O2 SumaVectoresC.c -o SumaVectoresC -lrt
ruben:~/Escritorio$ ./SumaVectoresC 2000
Tamaño Vectores:2000 ( 4 B )
Tiempo:0.000015074      / Tamaño Vectores:2000 / V1[0]+V2[0]=V3[0](200.000000+2
00.000000=400.000000) / / V1[1999]+V2[1999]=V3[1999](399.900000+0.100000=400.000
00) /
ruben:~/Escritorio$ █

```

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 ha tardado desde `clock_gettime(CLOCK_REALTIME,&cgt1);`, hasta el `clock_gettime(CLOCK_REALTIME,&cgt2);`, es decir el tiempo que tarda en hacer la suma de vectores.

- (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 normalmente 2 tipos, uno de tipo `time_t` que almacena en segundos y el otro de tipo `long` que almacena en nanosegundos.

- (c) ¿qué información devuelve exactamente la función `clock_gettime()` en la estructura de datos descrita en el apartado (b)? ¿qué representan los valores numéricos que devuelve?

RESPUESTA: La función guarda una estructura de datos de tipo `TIME`, que recibe por parámetro el tipo de instante de tiempo, en este caso `CLOCK_REALTIME` y la variable donde se guardara el tiempo, esta función devuelve un entero 0 o 1 que nos indica de si se ha ejecutado correctamente esa función o no. De tal forma que nos sirve el tiempo que ha tardado en ejecutarse una secuencia de código.

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:

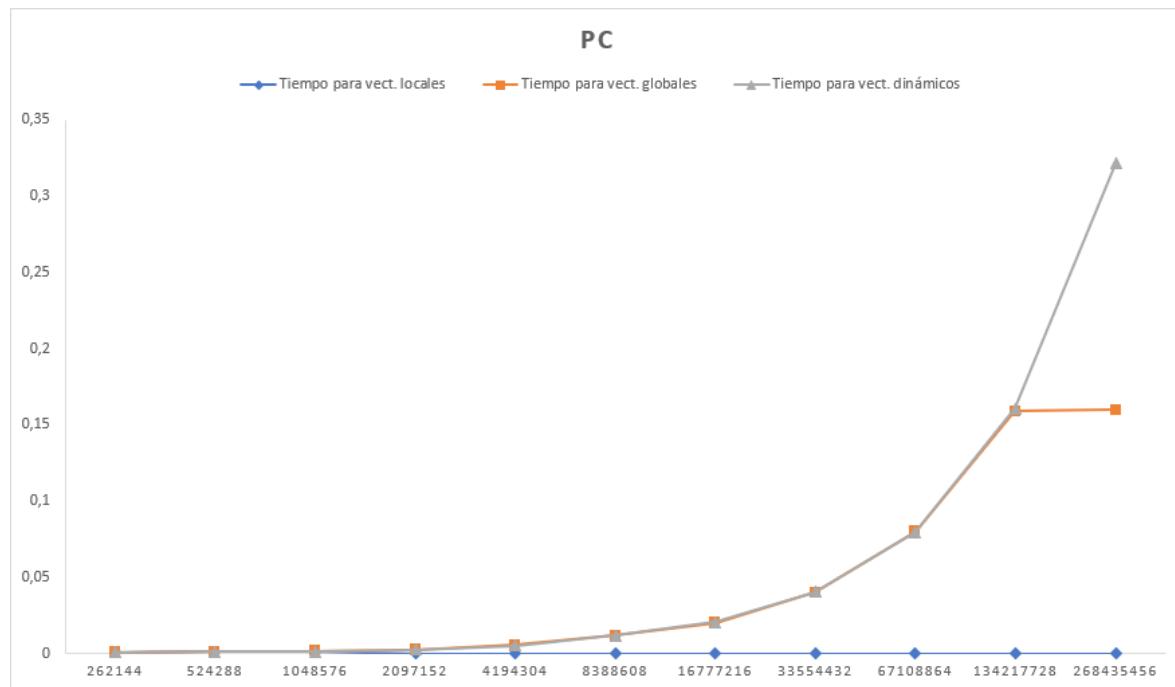
ATCGRID

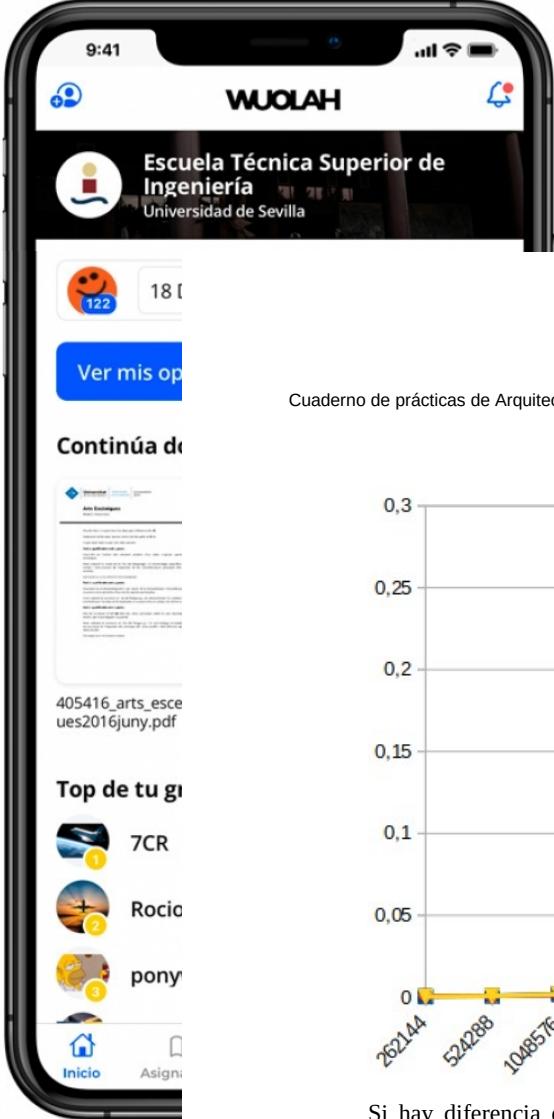
Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	262144	0,000460513	0,00094936	0,000361672
131072	524288	0,00093244	0,000675081	0,000783573
262144	1048576	0,001855519	0,001458693	0,001795226
524288	2097152	0	0,002813933	0,002624136
1048576	4194304	0	0,005704482	0,004839413
2097152	8388608	0	0,010449208	0,010311014
4194304	16777216	0	0,019079666	0,016980853
8388608	33554432	0	0,034205324	0,032710115
16777216	67108864	0	0,065158701	0,064199014
33554432	134217728	0	0,126698665	0,126350253
67108864	268435456	0	0,126845582	0,251929613

PC

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	262144	0.000505166	0.000532816	0.000525972
131072	524288	0.001076237	0.001120566	0.001080390
262144	1048576	0.001275231	0.001365620	0.001269394
524288	2097152	0	0.002562139	0.002519791
1048576	4194304	0	0.005714833	0.005090382
2097152	8388608	0	0.011560024	0.011611638
4194304	16777216	0	0.020097376	0.020691417
8388608	33554432	0	0.039791053	0.040187399
16777216	67108864	0	0.080041795	0.079376552
33554432	134217728	0	0.158696527	0.160649096
67108864	268435456	0	0.159842621	0.321234654

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:



Descarga la APP de Wuolah.

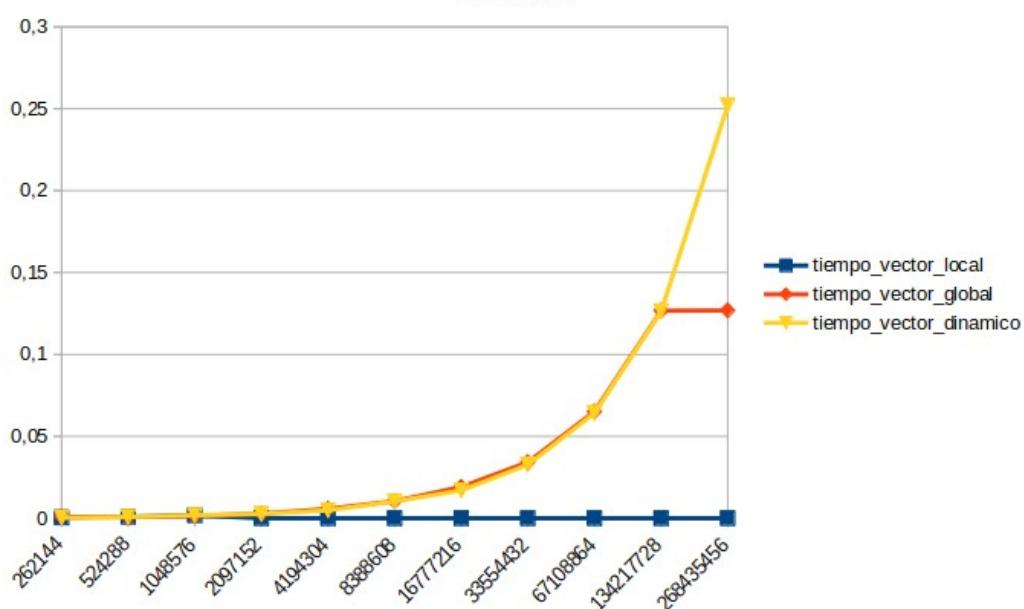
Ya disponible para el móvil y la tablet.

Available on the
App Store

GET IT ON
Google Play

Cuaderno de prácticas de Arquitectura de Computadores, Grado en Ingeniería Informática

Continúa de



Si hay diferencia entre los distintos tiempos de ejecución, de tal forma que influye la velocidad de procesamiento.

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: 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 cuando metemos un tamaño de 524288 para arriba violación de segmento. De tal forma se utiliza para valores 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 tal que si se supera se interpreta que es $33554432 * 4$ Bytes 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 está limitado, la diferencia a la local es que el tamaño en el que está 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 de tipo **unsigned int** de tal forma que su tamaño es de 4 Bytes = 32 bits, 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:

```
ruben:~/Escritorio/AC/BP0/ejer3$ gcc -O2 SumaVectoresC2.c -o SumaVectoresC2 -l
rt
/tmp/ccnT0vr0.o: En la función `main':
SumaVectoresC2.c:(.text.startup+0x76): reubicación truncada para ajustar: R_X86_
64_PC32 contra el símbolo `v2' definido en la sección COMMON en /tmp/ccnT0vr0.o
SumaVectoresC2.c:(.text.startup+0xc9): reubicación truncada para ajustar: R_X86_
64_PC32 contra el símbolo `v3' definido en la sección COMMON en /tmp/ccnT0vr0.o
collect2: error: ld returned 1 exit status
ruben:~/Escritorio/AC/BP0/ejer3$
```

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

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

```
/*
 * SumaVectores.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)
#ifndef 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;  
}
```