2º curso / 2º cuatr.

Grado Ing. Inform.

Doble Grado Ing. Inform. y Mat.

Arquitectura de Computadores (AC)

Cuaderno de prácticas.
Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos): Rafael Nogales Vaquero

Grupo de prácticas:

Fecha de entrega:

Fecha evaluación en clase:

[-RECORDATORIO, quitar todo este texto en rojo del cuaderno definitivo–

1. COMENTARIO

Este cuaderno de prácticas se utilizará para asignarle una puntuación durante la evaluación continua de prácticas y también lo utilizará como material de estudio y repaso para preparar el examen de prácticas escrito. Luego redáctelo con cuidado, y sea ordenado y claro.

2. NORMAS SOBRE EL USO DE LA PLANTILLA

1) Usar **interlineado SENCILLO**.

2) Respetar los tipos de letra y tamaños indicados:

- Calibri 11 para el texto
- **Courier New-10 para nombres de fichero, comandos, variables de entorno, etc., cuando se usan en el texto.**
- **Courier New-8 o Courier New-9 para el código fuente en los listados de código fuente.**
- Formatee el código fuente de los listados para que sea legible, limpio y claro. Consulte, como ejemplo, los Listados 1 y 2 del guion (tabule, comente, …)

3) Insertar las capturas de pantalla donde se pidan y donde se considere oportuno

Recuerde que debe **adjuntar al zip de entrega, el pdf de este fichero, todos los ficheros con código fuente implementados/utilizados y el resto de ficheros que haya implementado/utilizado (scripts, hojas de cálculo, etc.), lea la Sección 1.4 del guion]**

---

1. En el primer ejemplo de ejecución en atcgrid usando TORQUE se ejecuta el ejemplo `HelloOMP.c` de la página 12 del seminario usando la siguiente orden: `echo 'hello/HelloOMP' | qsub -q ac`. El resultado de la ejecución de este código en atcgrid se puede ver en la página 19 del seminario. Conteste a las siguientes preguntas:

   a. ¿Para qué se usa en `qsub` la opción `-q`?

      RESPUESTA: Para elegir la cola (en este caso la cola ac)

   b. ¿Cómo sabe el usuario que ha terminado la ejecución en atcgrid?

      RESPUESTA: Cuando en qstat no aparece tu trabajo. O al hacer ls que aparecen los archivos STDIN.oXXXX y STDIN.eXXXX

   c. ¿Cómo puede saber el usuario si ha habido algún error en la ejecución?

      RESPUESTA: Al abrir el archivo STDIN.eXXXX ves los errores (si está vacío es porque no ha habido errores)

   d. ¿Cómo ve el usuario el resultado de la ejecución?

      RESPUESTA: Abriendo el archivo `STDIN.oXXXX`

e. ¿Por qué en el resultado de la ejecución aparecen 24 saludos "¡¡¡Hello World!!!"?

RESPUESTA: Porque tiene 24 núcleos.

Se puede ver mandando la orden `echo 'cat /proc/cpuinfo' | qstat -q ac`

Y nos dice que hay 24 núcleos de está forma:

```
processor : 23
vendor_id : GenuineIntel
cpu family: 6
model: 44
model name: Intel(R) Xeon(R) CPU           E5645  @ 2.40GHz
stepping  : 2
cpu MHz   : 1600.000
cache size: 12288 KB
```

2. En el segundo ejemplo de ejecución en atcgrid usando TORQUE el script `script_helloomp.sh` de la página 22 del seminario usando la siguiente orden: `qsub script_helloomp.sh`. El script ejecuta varias veces el ejecutable del código `HelloOMP.c`. El resultado de la ejecución de este código en atcgrid se puede ver en la página 26 del seminario. Conteste a las siguientes preguntas:

   a. ¿Por qué no acompaña a al orden `qsub` la opción `-q` en este caso?

   RESPUESTA: Porque ya viene en el script al principio

   b. ¿Cuántas veces ejecuta el script el ejecutable `HelloOMP` en atcgrid?

   RESPUESTA: Lo ejecuta 4 veces

   c. ¿Cuántos saludos "¡¡¡Hello World!!!" se imprimen en cada ejecución? ¿Por qué se imprime ese número?

   RESPUESTA: Lo escribe tantas veces como hebras intervengan en esa ejecucion del programa `HelloOMP`.

3. Realizar las siguientes modificaciones en el script "¡¡¡Hello World!!!":

   - Eliminar la variable de entorno `$PBS_O_WORKDIR` en el punto en el que aparece.
   - Añadir lo necesario para que, cuando se ejecute el script, se imprima la variable de entorno `$PBS_O_WORKDIR`.

   Ejecutar el script con estas modificaciones. ¿Qué resultados de ejecución se obtienen en este caso? Incorporar en el cuaderno de trabajo volcados de pantalla que muestren estos resultados.

RESPUESTA:

```
[E1estudiante16@atcgrid ~]$ cat helloomp.o29458
Id. usuario del trabajo: E1estudiante16
Id. del trabajo: 29458.atcgrid
Nombre del trabajo especificado por usuario: helloomp
Nodo que ejecuta qsub: atcgrid
Cola: ac
Nodos asignados al trabajo:
Workdir: /home/E1estudiante16
atcgrid2
Nº de threads inicial: 12

Para 12 threads:

Para 6 threads:

Para 3 threads:

Para 1 threads:
[E1estudiante16@atcgrid ~]$
```

4. Incorporar en el fichero .zip que se entregará al profesor el fichero `/proc/cpuinfo` de alguno de los nodos de atcgrid (atcgrid1, atcgrid2, atcgrid3), del PC del aula de prácticas y de su PC (si tiene Linux instalado). Indique qué ha hecho para obtener el contenido de `/proc/cpuinfo` en atcgrid.

   Mi ordenador local: `/usr/sbin/system_profiler > ~/Desktop/hardware_info`

   (*) En linux sería con `cat /proc/cpuinfo` pero en OSX no existe dicho archivo, aunque si existe un equivalente: `system_profiler` que contiene la información de todo el hardware del ordenador incluyendo puertos físicos, tarjetas de video y audio, estado de los drivers...
   Solo tenemos que buscar en medio de todo eso para encontrar la parte relativa a la CPU:

```
Hardware Overview:
      Model Name: MacBook Pro
      Model Identifier: MacBookPro9,2
      Processor Name: Intel Core i5
      Processor Speed: 2,5 GHz
      Number of Processors: 1
      Total Number of Cores: 2
      L2 Cache (per Core): 256 KB
      L3 Cache: 3 MB
      Memory: 16 GB
      Boot ROM Version: MBP91.00D3.B08
      SMC Version (system): 2.2f38
      Serial Number (system): C02HQ49ZDTY3
      Hardware UUID: F077752A-7F70-5619-A213-FAF0801F78BB
      Sudden Motion Sensor:
          State: Enabled
```

   En el front-end `cat /proc/cpuinfo > frontend-CPUinfo`

   En atcgrid `echo "cat /proc/cpuinfo" | qsub -q ac`

5. En el Listado 1 se puede ver un código fuente C que calcula la suma de dos vectores y en el Listado 2 una versión con C++:

   v3 = v1 + v2; v3(i) = v1(i) + v2(i), i=0,…N-1

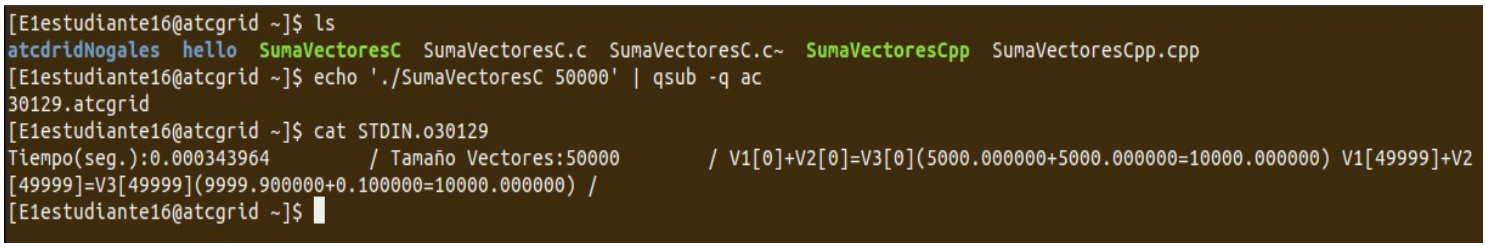
   Los códigos utilizan directivas del compilador para fijar el tipo de variable de los vectores (v1, v2 y v3). En los comentarios que hay al principio de los códigos se indica cómo hay que compilarlos. Los vectores pueden ser:

   - Variables locales: descomentando en el código `#define VECTOR_LOCAL` y comentando `#define VECTOR_GLOBAL` y `#define VECTOR_DYNAMIC`
   - Variables globales: descomentando `#define VECTOR_GLOBAL` y comentando `#define VECTOR_LOCAL` y `#define VECTOR_DYNAMIC`
   - Variables dinámicas: descomentando `#define VECTOR_DYNAMIC` y comentando `#define VECTOR_LOCAL` y `#define VECTOR_GLOBAL`. Si se usan los códigos tal y como están en Listado 1 y Listado 2, sin hacer ningún cambio, los vectores (v1, v2 y v3) serán variables dinámicas.

   Por tanto, se debe definir sólo una de las siguientes constantes: `VECTOR_LOCAL`, `VECTOR_GLOBAL` o `VECTOR_DYNAMIC`.
   En los dos códigos (Listado 1 y Listado 2) se utiliza la función `clock_gettime()` para obtener el tiempo de ejecución del trozo de código que calcula la suma de vectores. En el código se imprime la variable `ncgt`, ¿qué contiene esta variable? ¿qué devuelve la función `clock_gettime()`?
   RESPUESTA:
   `ncgt` contiene el tiempo en segundos que ha tardado en ejecutarse el programa.
   `clock_gettime()` devuelve el tiempo actual.

   Escribir en el cuaderno de prácticas las diferencias que hay entre el código fuente C y el código fuente C++ para la suma de vectores.
   RESPUESTA:
   El algoritmo de suma es el mismo, la diferencia es que en C la forma de gestionar la memoria dinámica con `malloc` y `free` en C y `new` y `delete` en C++.
   (Tambien se diferencian en el `printf` y `cout`)

6. Generar 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`). Ejecutar el código ejecutable resultante en atcgrid usando el la cola TORQUE. Incorporar volcados de pantalla que demuestren la ejecución correcta en atcgrid.
   RESPUESTA:



7. Ejecutar en atcgrid el código generado en el apartado anterior usando el `script` del Listado 3. Generar el ejecutable usando la opción de optimización –O2 tal y como se indica en el comentario que hay al principio del programa. Ejecutar el código también en su PC local para los mismos tamaños. ¿Se obtiene error para alguno de los tamaños? En caso afirmativo, ¿a qué se debe este error?
   RESPUESTA:

```
[E1estudiante16@atcgrid ~]$ qsub SumaVectores.sh
30130.atcgrid
[E1estudiante16@atcgrid ~]$ ls
atcdridNogales  SumaVectoresC    SumaVectoresC.c~  SumaVectoresCpp.cpp           SumaVectoresC_vlocales.o30130
hello           SumaVectoresC.c  SumaVectoresCpp   SumaVectoresC_vlocales.e30130  SumaVectores.sh
[E1estudiante16@atcgrid ~]$ cat SumaVectoresC_vlocales.o30130
Id. usuario del trabajo: E1estudiante16
Id. del trabajo: 30130.atcgrid
Nombre  del trabajo especificado por usuario: SumaVectoresC_vlocales
Nodo que ejecuta qsub: atcgrid
Directorio en el que se ha ejecutado qsub: /home/E1estudiante16
Cola: ac
Nodos asignados al trabajo:
atcgrid1
Tiempo(seg.):0.000503271        / Tamaño Vectores:65536        / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) V1[65535]+V2
[65535]=V3[65535](13107.100000+0.100000=13107.200000) /
Tiempo(seg.):0.000887142        / Tamaño Vectores:131072       / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.400000) V1[131071]
+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tiempo(seg.):0.001814257        / Tamaño Vectores:262144       / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.800000) V1[262143]
+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
[E1estudiante16@atcgrid ~]$ cat SumaVectoresC_vlocales.e30130
/var/spool/torque/mom_priv/jobs/30130.atcgrid.SC: línea 20: 31931 Violación de segmento  (`core' generado) $PBS_O_WORKDIR/SumaVectores
C $N
/var/spool/torque/mom_priv/jobs/30130.atcgrid.SC: línea 20: 31934 Violación de segmento  (`core' generado) $PBS_O_WORKDIR/SumaVectores
C $N
/var/spool/torque/mom_priv/jobs/30130.atcgrid.SC: línea 20: 31937 Violación de segmento  (`core' generado) $PBS_O_WORKDIR/SumaVectores
C $N
/var/spool/torque/mom_priv/jobs/30130.atcgrid.SC: línea 20: 31939 Violación de segmento  (`core' generado) $PBS_O_WORKDIR/SumaVectores
C $N
/var/spool/torque/mom_priv/jobs/30130.atcgrid.SC: línea 20: 31941 Violación de segmento  (`core' generado) $PBS_O_WORKDIR/SumaVectores
C $N
/var/spool/torque/mom_priv/jobs/30130.atcgrid.SC: línea 20: 31943 Violación de segmento  (`core' generado) $PBS_O_WORKDIR/SumaVectores
C $N
/var/spool/torque/mom_priv/jobs/30130.atcgrid.SC: línea 20: 31945 Violación de segmento  (`core' generado) $PBS_O_WORKDIR/SumaVectores
C $N
/var/spool/torque/mom_priv/jobs/30130.atcgrid.SC: línea 20: 31947 Violación de segmento  (`core' generado) $PBS_O_WORKDIR/SumaVectores
C $N
[E1estudiante16@atcgrid ~]$
```

   Se obtiene error debido a que se sobrepasa el tamaño maximo de la pila.

8. Generar los ejecutables del código fuente C para vectores globales y para dinámicos. Genere el ejecutable usando –O2. Ejecutar los dos códigos en atcgrid usando un `script` como el del Listado 3 (hay que poner en el `script` el nombre de los ficheros ejecutables generados en este ejercicio) para el mismo rango de tamaños utilizado en el ejercicio anterior. Ejecutar también los códigos en su PC local. ¿Se obtiene error usando vectores globales o dinámicos? ¿A qué cree que es debido?
   RESPUESTA:
   No hay error debido a que los vectores globales se almacenan directamente en el ejecutable y el de vectores dinámicos almacena el vector en la memoria principal y no en la pila como el que usa vectores locales.

9. Rellenar una tabla como la Tabla 1 .para atcgrid y otra para el PC local con los tiempos de ejecución obtenidos en los ejercicios anteriores para el trozo de código que realiza la suma de vectores. En la columna "Bytes de un vector" hay que poner el total de bytes reservado para un vector. Ayudándose de una hoja de cálculo represente en una misma gráfica los tiempos de ejecución obtenidos en atcgrid para vectores locales, globales y dinámicos (eje y) en función del tamaño en bytes de un vector (eje x). Realice otra gráfica con los tiempos obtenidos en el PC local.
RESPUESTA:

Tabla 1 . Tiempo de ejecución de la suma de vectores para vectores locales, globales y dinámicos

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288 B	0.000471311	0.000546472	0.000428862
131072	1048576 B	0.000983899	0.000486419	0.000885637
262144	2097152 B	0.001509065	0.001444043	0.001812072
524288	4194304 B	Stack overflow	0.002656301	0.002736547
1048576	8388608 B	Stack overflow	0.005286185	0.004820643
2097152	16777216 B	Stack overflow	0.008720559	0.008935546
4194304	33554432 B	Stack overflow	0.017671053	0.016605114
8388608	67108864 B	Stack overflow	0.034306461	0.032816315
16777216	134217728 B	Stack overflow	0.068669328	0.065074436
33554432	268435456 B	Stack overflow	0.133166682	0.128591741
67108864	536870912 B	Stack overflow	(*)0.133720562(*)	0.253786113

Atcgrid

Tabla 2 . Tiempo de ejecución de la suma de vectores para vectores locales, globales y dinámicos

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288 B	0.000439885	0.001119930	0.000552952
131072	1048576 B	0.000858632	0.000938974	0.000895647
262144	2097152 B	0.001791168	0.002263957	0.002583060
524288	4194304 B	Stack overflow	0.003985906	0.003607027
1048576	8388608 B	Stack overflow	0.008259308	0.009580643
2097152	16777216 B	Stack overflow	0.012739410	0.014207407
4194304	33554432 B	Stack overflow	0.024119238	0.026142184
8388608	67108864 B	Stack overflow	0.048427849	0.048889406
16777216	134217728 B	Stack overflow	0.105067384	0.107017866
33554432	268435456 B	Stack overflow	0.268460655	0.196285935
67108864	536870912 B	Stack overflow	(*) 0.201799026 (*)	(**)0.915523797(**)

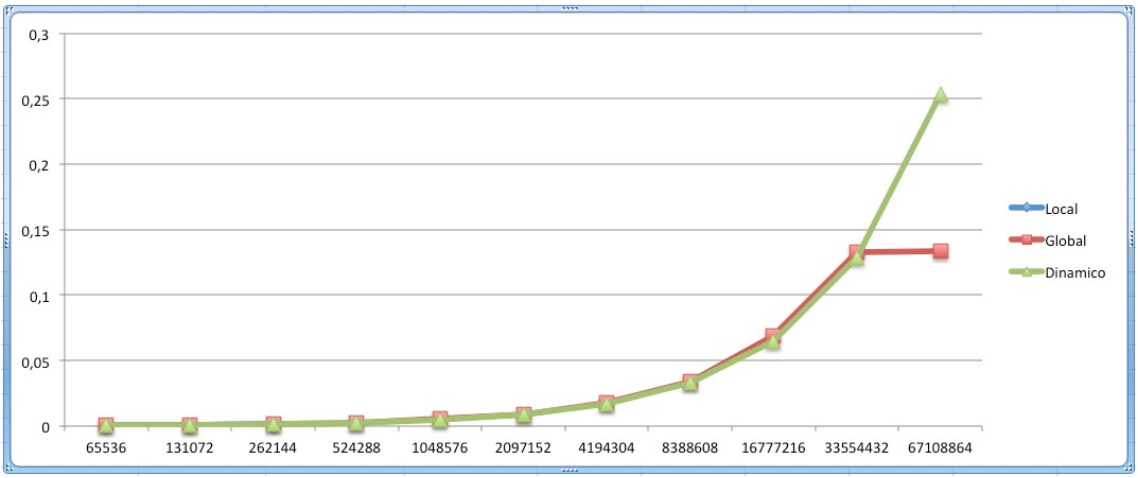
Ordenador Local (intel core i5)

Comentarios:

(*)Como 67108864 se pasa del maximo que es: 33554432 declara tan solo un vector de 33554432 componentes:

```
//Aqui está el motivo
#ifdef VECTOR_GLOBAL
#define MAX 33554432    //=2^25
double v1[MAX], v2[MAX], v3[MAX];
#endif
```

(**) Cuando el vector no cabe en caché se realizan accesos a memoria RAM que son mucho mas lentos, eso puede explica que tarde se 5 veces más en ejecutar el doble de sumas.
(En algunas ejecuciones tarda solo 0.4 segundos y en otras llega a 1.1 segundos)



10. Modificar el código fuente C 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 (MAX=2^32-1). Generar el ejecutable usando variables globales. ¿Qué ocurre? Razone además por qué el máximo número que se puede almacenar en N es 2^{32}-1.
RESPUESTA:

```
//#define VECTOR_LOCAL

#define VECTOR_GLOBAL

//#define VECTOR_DYNAMIC

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

No puede compilar porque el archivo es demasiado pesado.
De hecho v1 pesa 32GB
(2^32 -1)* 8 /(1024*1024*1024)= 32GB
De hecho el problema está en la fase de enlazado,
la fase de compilación funciona correctamente:
v1 sobrepasa el tamaño maximo y no queda espacio para v2 ni para v3

```
[E1estudiante16@atcgrid ~]$ ls
local.sh  SumaVectoresC.c  SumaVectoresC.c~  SumaVectoresCpp  SumaVectoresCpp.cpp  SumaVectoresLocal
[E1estudiante16@atcgrid ~]$ gcc -c SumaVectoresC.c
[E1estudiante16@atcgrid ~]$ ls
local.sh  SumaVectoresC.c  SumaVectoresC.c~  SumaVectoresC.o  SumaVectoresCpp  SumaVectoresCpp.cpp  SumaVectoresLocal
[E1estudiante16@atcgrid ~]$ ld SumaVectoresC.o -o SumaVectoresGlobal-Ejecutable
ld: warning: cannot find entry symbol _start; defaulting to 0000000004000e8
SumaVectoresC.o: In function `main':
SumaVectoresC.c:(.text+0x1b): undefined reference to `puts'
SumaVectoresC.c:(.text+0x25): undefined reference to `exit'
SumaVectoresC.c:(.text+0x38): undefined reference to `atoi'
SumaVectoresC.c:(.text+0xed): relocation truncated to fit: R_X86_64_32S against symbol `v2' defined in COMMON section in SumaVectoresC
.o
SumaVectoresC.c:(.text+0x10e): undefined reference to `clock_gettime'
SumaVectoresC.c:(.text+0x133): relocation truncated to fit: R_X86_64_32S against symbol `v2' defined in COMMON section in SumaVectores
C.o
SumaVectoresC.c:(.text+0x145): relocation truncated to fit: R_X86_64_32S against symbol `v3' defined in COMMON section in SumaVectores
C.o
SumaVectoresC.c:(.text+0x162): undefined reference to `clock_gettime'
SumaVectoresC.c:(.text+0x1b4): relocation truncated to fit: R_X86_64_32S against symbol `v3' defined in COMMON section in SumaVectores
C.o
SumaVectoresC.c:(.text+0x1c5): relocation truncated to fit: R_X86_64_32S against symbol `v2' defined in COMMON section in SumaVectores
C.o
SumaVectoresC.c:(.text+0x1f0): relocation truncated to fit: R_X86_64_PC32 against symbol `v3' defined in COMMON section in SumaVectore
sC.o
SumaVectoresC.c:(.text+0x1f8): relocation truncated to fit: R_X86_64_PC32 against symbol `v2' defined in COMMON section in SumaVectore
sC.o
SumaVectoresC.c:(.text+0x21d): undefined reference to `printf'
[E1estudiante16@atcgrid ~]$
```