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): Adrián Portillo Sánchez
Grupo de prácticas: A2
Fecha de entrega: 11/03/2014 (hasta las 12 pm)
Fecha evaluación en clase: 19/03/2014

[-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 o Liberation Serif-11 para el texto
- Courier New-10 o Liberation Mono-10 para nombres de fichero, comandos, variables de entorno, etc., cuando se usan en el texto.
- Courier New o Liberation Mono de tamaño 8 o 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 10 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 17 del seminario. Conteste a las siguientes preguntas:

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

RESPUESTA: Para enviar el trabajo a la cola ac, -q establece un destino al trabajo que puede ser una cola o un servidor; en este caso el trabajo se envía a la cola ac

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

RESPUESTA: Hay dos posibles maneras, la primera sería utilizar el comando qstat para ver si hay algún trabajo pendiente, la segunda sería utilizar el comando ls hasta que aparezcan los archivos de salida.

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

RESPUESTA: Leyendo el archivo de salida de errores, que tiene el formato STDIN.e*, en él qsub deja la salida de errores de la ejecución del programa.

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

RESPUESTA: Leyendo el archivo de salida del programa, que tiene el formato STDIN.o*, en él qsub deja la salida del programa cuando este se ejecuta

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

RESPUESTA: Porque en el programa se ejecutan las hebras a través de OMP parallel, que indica el conjunto de instrucciones del programa que se pueden ejecutar de forma paralela, como el pc local posee un procesador de 4 núcleos, se ejecutan 4 hebras; en el cluster se ejecutan 24 hebras, ya que el cluster posee 2 procesadores, cada uno con 6 núcleos con hyperthreading(12 núcleos con 2 hebras cada), por tanto el programa se ejecuta en 24 hebras.

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:

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

RESPUESTA: porque en este caso no se especifica la cola de destino de la ejecución del programa, así que por defecto este se manda a la cola predeterminada, que igualmente es la cola ac.

c. ¿Cuántas veces ejecuta el script el ejecutable HelloOMP en atcgrid? ¿Por qué ejecuta ese número?

RESPUESTA: Lo ejecuta 4 veces, cada una para hacer una prueba distinta asignándole al programa la mitad de hebras que la anterior ejecución, hasta llegar a 1 hebra solamente; así nos muestra como se ejecutan los programas en OMP.

d. ¿Cuántos saludos "¡¡¡Hello World!!!" se imprimen en cada ejecución? (indique el número exacto) ¿Por qué se imprime ese número?

RESPUESTA: En la primera se imprimen 12, luego 6, luego 3 y luego 1, cada vez la mitad, ya que el script por cada ejecución establece el número máximo de hebras que pueden ejecutar el programa a la mitad, comenzando por 12 hebras, hasta llegar a 0.

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: $PBS_O_WORKDIR contiene el directorio de trabajo actual, si borramos esta variable el programa no se ejecuta, ya que no puede leer el programa si no posee la dirección de trabajo actual, al mostrarlo este contiene, en mi caso: /home/A2estudiante21/hello

```
Nombre del trabajo especificado por usuario: helloomp
Nodo que ejecuta qsub: atcgrid
Cola: ac
Nodos asignados al trabajo:
atcgrid1
Nº de threads inicial: 12

Para 12 threads:
variable PBS_O_WORKDIR: /home/A2estudiante21/hello

Para 6 threads:
variable PBS_O_WORKDIR: /home/A2estudiante21/hello

Para 3 threads:
variable PBS_O_WORKDIR: /home/A2estudiante21/hello

Para 1 threads:
variable PBS_O_WORKDIR: /home/A2estudiante21/hello
```

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.

RESPUESTA: En la terminal de ssh ejecuté la línea de comandos echo 'cat /proc/cpuinfo' | qsub -q ac ; luego mediante el comando qstat pude ver que se correspondia con el trabajo 3847, por lo que al acabar la ejecución hice cat STDIN.o3847, el resultado mostraba las especificciones de cada uno de los núcleos.

Teniendo en cuenta el contenido de cpuinfo conteste a las siguientes preguntas (justifique las respuestas):

a. ¿Cuántos cores físicos y cuántos cores lógicos tiene el PC del aula de prácticas?

RESPUESTA: 4 cores físicos y 4 lógicos.

b. ¿Cuántos cores físicos y cuántos cores lógicos tiene su PC?

RESPUESTA: 4 cores físicos y 8 lógicos (posee hyperthreading).

c. ¿Cuántos cores físicos y cuántos cores lógicos tiene un nodo de atcgrid?

RESPUESTA: 12 cores físicos y 24 lógicos (también por el hyperthreading).

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`.

b. 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 información devuelve exactamente la función clock_gettime()? ¿en qué estructura de datos devuelve clock_gettime() la información?

RESPUESTA: La variable ngct contiene el tiempo total que ha tardado el programa en ejecutarse.

La función clock_gettime() devuelve el instante de tiempo en el que se encuentra el sistema cuando se ejecuta.

En una estructura timespec.

b. 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: En lugar de cout se utiliza printf en c.

Los vectores se utilizan dinámicamente sin necesidad de usar malloc y realloc en c++.

En lugar de utilizar free utiliza delete [] en c++.

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:

```
[A2estudiante21@atcgrid ~]$ echo './SumaVectores 100000' | qsub -q ac
7642.atcgrid
[A2estudiante21@atcgrid ~]$ qstat
Job id                    Name             User            Time Use S Que
------------------------- ---------------- --------------- -------- - ---
7642.atcgrid               STDIN            A2estudiante21  00:00:00 C ac
[A2estudiante21@atcgrid ~]$ ls
hello  STDIN.e7642  STDIN.o7642  SumaVectores
[A2estudiante21@atcgrid ~]$ cat STDIN.o7642
Tiempo(seg.):0.000614355         / Tamaño Vectores:100000       / V1[0]+V
[A2estudiante21@atcgrid ~]$ cat STDIN.o7642
Tiempo(seg.):0.000614355         / Tamaño Vectores:100000       / V1[0]+V
2[0]=V3[0](10000.000000+10000.000000=20000.000000) / /V1[99999]+V2[99999]
=V3[99999](19999.900000+0.100000=20000.000000) /
[A2estudiante21@atcgrid ~]$
```

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:

```
[A2estudiante21@atcgrid ~]$ echo './listado3' | qsub -q ac
7651.atcgrid
[A2estudiante21@atcgrid ~]$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
7651.atcgrid               STDIN            A2estudiante21  00:00:00 C ac
[A2estudiante21@atcgrid ~]$ ls
hello     STDIN.e7642  STDIN.e7651  STDIN.o7649  SumaVectores
listado3  STDIN.e7649  STDIN.o7642  STDIN.o7651  SumaVectoresC
[A2estudiante21@atcgrid ~]$ cat STDIN.o7651
Id. usuario del trabajo: A2estudiante21
Id. del trabajo: 7651.atcgrid
Nombre del trabajo especificado por usuario: STDIN
Nodo que ejecuta qsub: atcgrid
Directorio en el que se ha ejecutado qsub: /home/A2estudiante21
Cola: ac
Nodos asignados al trabajo:
atcgrid1
Tiempo(seg.):0.000356986         / Tamaño Vectores:65536        / V1[0]+V2[0]=V3[0](65
53.600000+6553.600000=13107.200000) / /V1[65535]+V2[65535]=V3[65535](13107.100000+0.10
0000=13107.200000) /
Tiempo(seg.):0.000736830         / Tamaño Vectores:131072       / V1[0]+V2[0]=V3[0](13
107.200000+13107.200000=26214.400000) / /V1[131071]+V2[131071]=V3[131071](26214.300000
+0.100000=26214.400000) /
Tiempo(seg.):0.001470120         / Tamaño Vectores:262144       / V1[0]+V2[0]=V3[0](26
214.400000+26214.400000=52428.800000) / /V1[262143]+V2[262143]=V3[262143](52428.700000
+0.100000=52428.800000) /
```

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:

```
[A2estudiante21@atcgrid ~]$ echo './listado4' | qsub -q ac
7654.atcgrid
[A2estudiante21@atcgrid ~]$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
7654.atcgrid               STDIN            A2estudiante21         0 R ac
[A2estudiante21@atcgrid ~]$ ls
hello     STDIN.e7642  STDIN.o7642  SumaVectores   SumaVectoresG
listado3  STDIN.e7649  STDIN.o7649  SumaVectoresC
listado4  STDIN.e7651  STDIN.o7651  SumaVectoresD
[A2estudiante21@atcgrid ~]$ cat STDIN.o7654
Id. usuario del trabajo: A2estudiante21
Id. del trabajo: 7654.atcgrid
Nombre del trabajo especificado por usuario: STDIN
Nodo que ejecuta qsub: atcgrid
Directorio en el que se ha ejecutado qsub: /home/A2estudiante21
Cola: ac
Nodos asignados al trabajo:
atcgrid1
Tiempo(seg.):0.000363315         / Tamaño Vectores:65536         / V1[0]+V2[0]=V3[0](65
53.600000+6553.600000=13107.200000) / /V1[65535]+V2[65535]=V3[65535](13107.100000+0.10
0000=13107.200000) /
Tiempo(seg.):0.000384668         / Tamaño Vectores:65536         / V1[0]+V2[0]=V3[0](65
53.600000+6553.600000=13107.200000) / /V1[65535]+V2[65535]=V3[65535](13107.100000+0.10
0000=13107.200000) /
Tiempo(seg.):0.000690413         / Tamaño Vectores:131072        / V1[0]+V2[0]=V3[0](13
107.200000+13107.200000=26214.400000) / /V1[131071]+V2[131071]=V3[131071](26214.300000
+0.100000=26214.400000) /
```

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). Utilice escala logarítmica en el eje ordenadas (eje y) en todas las gráficas. ¿Hay diferencias en los tiempos de ejecución con vectores locales, globales y dinámicos?

RESPUESTA:

Tabla 1 . Tiempos 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	2097152	0.000401419	0.000384668	0.000363315
131072	4194304	0.000766951	0.000736667	0.000690413
262144	8388608	0.001213772	0.001161636	0.001478022
524288	16777216	0.263896232	0.002392828	0.002394889
1048576	33554432	0.005344576	0.005162357	0.005245336
2097152	67108864	0.013934624	0.010279227	0.010367873
4194304	134217728	0.025747289	0.020305005	0.020343156
8388608	268435456	0.050854378	0.040185059	0.040364337
16777216	536870912	0.864154654	0.079929387	0.080732197
33554432	1073741824	0.179983202	0.159609643	0.160772871
67108864	2147483648	0.379845129	0.349083471	0.322735870

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? ¿A qué es debido? Razone además por qué el máximo número que se puede almacenar en N es 2^{32}-1.

RESPUESTA:

```
SumaVectores.c:(.text.startup+0x102): reubicación truncada para ajustar: R_X86_6
4_32S contra el símbolo `v2' definido en la sección COMMON en /tmp/ccUbErYv.o
collect2: error: ld returned 1 exit status
adri@adri-virtual-machine ~/Descargas $ gcc -O2 SumaVectores.c -o SumaVectores4
-lrt
adri@adri-virtual-machine ~/Descargas $ gcc -O2 SumaVectores.c -o SumaVectores4
-lrt
/tmp/cc4aeUxR.o: En la función `main':
SumaVectores.c:(.text.startup+0x65): reubicación truncada para ajustar: R_X86_64
_32S contra el símbolo `v2' definido en la sección COMMON en /tmp/cc4aeUxR.o
SumaVectores.c:(.text.startup+0x9e): reubicación truncada para ajustar: R_X86_64
_32S contra el símbolo `v2' definido en la sección COMMON en /tmp/cc4aeUxR.o
SumaVectores.c:(.text.startup+0xa7): reubicación truncada para ajustar: R_X86_64
_32S contra el símbolo `v3' definido en la sección COMMON en /tmp/cc4aeUxR.o
SumaVectores.c:(.text.startup+0xcf): reubicación truncada para ajustar: R_X86_64
_PC32 contra el símbolo `v3' definido en la sección COMMON en /tmp/cc4aeUxR.o
SumaVectores.c:(.text.startup+0xde): reubicación truncada para ajustar: R_X86_64
_PC32 contra el símbolo `v2' definido en la sección COMMON en /tmp/cc4aeUxR.o
SumaVectores.c:(.text.startup+0xe7): reubicación truncada para ajustar: R_X86_64
_32S contra el símbolo `v3' definido en la sección COMMON en /tmp/cc4aeUxR.o
SumaVectores.c:(.text.startup+0x102): reubicación truncada para ajustar: R_X86_6
4_32S contra el símbolo `v2' definido en la sección COMMON en /tmp/cc4aeUxR.o
collect2: error: ld returned 1 exit status
adri@adri-virtual-machine ~/Descargas $
```

No me permite compilarlo ya que se pasa del tamaño máximo que el programa puede soportar, ya que un double tiene un tamaño máximo de 32 bits, y ese elemento es double.