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): Cristóbal López Peñalver
Grupo de prácticas:
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 indicar a qué cola queremos mandarlo señalando a continuación el nombre de ésta.

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

RESPUESTA: Si se ha terminado la ejecución del trabajo en atcgrid, dejará de aparecer en "qstat".

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

RESPUESTA: Si ha habido algún error se guardará en un archivo con extensión .e[número de proceso].

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

RESPUESTA: En un archivo con extensión .o[número de proceso].

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

RESPUESTA: Porque "#pragma omp parallel" hace que lo ejecuten todas las hebras; tenemos 2 procesadores, de 6 cores con posibilidad de ejecutar 2 hebras al mismo tiempo (2*6*2 = 24).

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 ya ha sido asignada la cola dentro del script gracias a la instrucción "#PBS -q ac".

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

RESPUESTA: 4 veces, ya que el lanzador del ejecutable se encuentra dentro de un bucle for cuyo contador parte de 12 y desciende la mitad cada vez mientras es menor que 0.

Debería ejecutarse 5 veces, pero cada iteración reduce a la mitad las hebras que se van a utilizar, y no se puede usar media hebra.

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

RESPUESTA: 12, 6, 3 y 1 respectivamente porque usamos ese número de hebras en cada ejecución; el ejecutable imprime un "¡¡¡Hello World!!!"por cada hebra disponible.

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:

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:

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

a. 1 físico y 4 lógicos ¿Cuántos cores físicos y cuántos cores lógicos tiene el PC del aula de prácticas?

RESPUESTA:

b. 1 físico y 2 lógicos ¿Cuántos cores físicos y cuántos cores lógicos tiene su PC?

RESPUESTA:

c. 2 físicos y 12 lógicos ¿Cuántos cores físicos y cuántos cores lógicos tiene un nodo de atcgrid?

RESPUESTA:

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 ncgt contiene el tiempo que tardan en sumarse los vectores en segundos (9 decimales)

La función clock_gettime devuelve 0, si no ha habido ningún error en su ejecución, o -1, en caso contrario. Se utiliza para almacenar en una estructura llamada timespec los segundos transcurridos desde las 0 horas del 1 de enero de 1970 (junto a los nanosegundos transcurridos desde que se completó el último segundo).

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: Además de que mismas variables se declaran en sitios distintos y ni C tiene cout ni C++ printf, a la hora de declarar el vector dinámico reservan memoria de distinta forma; en C usamos la orden malloc para solicitar un bloque de memoria del tamaño pasado por argumento (los bytes de cada uno de los double que contendrá el vector), devolviendo un puntero a la zona concedida, mientras que en C++ adjudicamos ese espacio con "new double[número de doubles]". En C usamos free(vector) para liberar su zona de memoria y en C++ delete[] vector.

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.

```
[B1estudiante11@atcgrid vectores]$ echo /home/B1estudiante11/vectores/SumaVector
esC 10000 | qsub -q ac
5264.atcgrid
[B1estudiante11@atcgrid vectores]$ cat STDIN.o5264
Tiempo(seg.):0.000039324        / Tamaño Vectores:10000        / V1[0]+V2[0]=V3
[0 (1000.000000+1000.000000=2000.000000) / / V1[9999]+V2[9999]=V3[9999](1999.900
000+0.100000=2000.000000) /
```

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: Suponiendo que el script funciona correctamente se debería ejecutar 11 veces, pero solo lo hace 3. El error surge cuando el script manda operar con un tamaño de 524.288 en adelante, devolviendo un error de violación de segmento debido a que estamos intentando acceder a una memoria a la que no tenemos acceso.

A partir de cierto número menor que 524.288 no caben en el stack.

En el PC Local me da el mismo error para dichas cantidades.

349281

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: Con vectores globales se ejecuta dos veces con 33554432 ya que la última ejecución supera su máximo, fijando el tamaño en éste.

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:

ATCGRID

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000286859	0.000298166	0.000279630
131072	1048576	0.000576648	0.000612035	0.000860523
262144	2097152	0.001228653	0.001174383	0.001163252
524288	4194304	X	0.002641023	0.003449472
1048576	8388608	X	0.005870047	0.006321350
2097152	16777216	X	0.011412814	0.011574259
4194304	33554432	X	0.022441788	0.021719402
8388608	67108864	X	0.042889291	0.045014921
16777216	134217728	X	0.086319579	0.090016490
33554432	268435456	X	0.176005432	0.178090129
67108864	536870912	X	0.175949952	0.348772189

MI PC

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000167997	0.000486081	0.000441357
131072	1048576	0.000388215	0.000709023	0.000874345
262144	2097152	0.001691561	0.001308249	0.001922914
524288	4194304	X	0.001856599	0.003214093
1048576	8388608	X	0.004353945	0.004165672
2097152	16777216	X	0.007140514	0.007094639
4194304	33554432	X	0.015410614	0.012970351
8388608	67108864	X	0.027783264	0.026065351
16777216	134217728	X	0.060670580	0.052372856
33554432	268435456	X	0.115317333	0.103518699
67108864	536870912	X	0.107346401	0.395937246

No soy capaz de hacer el gráfico; me da muchísimos problemas.

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:

Da error de compilación ya que no es capaz de reservar tal inmensa cantidad de datos.

Unsigned int tiene 32 bits, con lo que codificar 2^32 números. Si contamos desde 0, el máximo número será 2^32 – 1.