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): Marta Gómez Macías
Grupo de prácticas: C1
Fecha de entrega: 11/03/2014 (hasta las 12 pm)
Fecha evaluación en clase: 19/03/2014

---

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 asignar al trabajo que vamos a lanzar una determinada cola, en este caso, la cola **ac**

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

RESPUESTA: ejecutando el comando qstat podemos ver el estado del proceso en ejecución, tras la ejecución nos encontramos dos ficheros, uno acabado en .e y otro acabado en .o que se generan al finalizar la ejecución.

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

RESPUESTA: viendo que el tamaño del fichero .e generado es distinto de cero, se puede ver ejecutando en terminal ls -lag

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

RESPUESTA: viendo el contenido del fichero .o, esto en terminal puede hacerse con la orden cat.

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

RESPUESTA: Porque los procesadores del atcgrid tienen hyperthreading y entre todos dan un total de 24 cores virtuales y nuestro programa se ejecutaba en paralelo en cada uno de ellos.

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 la cola ya se especifica dentro del script con la orden **#PBS -q ac**

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

RESPUESTA: cuatro veces, porque es el número de ciclos que tiene el bucle for que hay al final del script, teniendo en cuenta que P inicialmente vale 12 y que en cada ciclo se reduce a la mitad.

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

RESPUESTA: Se ejecutan tantos saludos como sea el valor de P. Porque P especifica el máximo de hebras que están ejecutando en paralelo el script.

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: Se obtienen varios errores, en concreto 4, porque no ha podido encontrar la orden HelloOMP, y en la salida de la ejecución no se ha mostrado ningún saludo.

```
[marta@marta-PC ~]$ cd Documentos/AC/Bloque0/Ejemplo/
[marta@marta-PC Ejemplo]$ cat ejercicio3.o1057
Id. usuario del trabajo:  PBS_O_LOGNAME
Id. del trabajo: 1057.atcgrid
Nombre del trabajo especificado por usuario: ejercicio3
Nodo que ejecuta qsub: atcgrid
Cola: ac
Nodos asignados al trabajo:
atcgrid1
Nº de threads inicial: 12
/home/C1estudiante9/hello

Para 12 threads:

Para 6 threads:

Para 3 threads:

Para 1 threads:
```

```
[marta@marta-PC Ejemplo]$ cat ejercicio3.e1057
/var/lib/torque/mom_priv/jobs/1057.atcgrid.SC: line 24: HelloOMP: command not found
/var/lib/torque/mom_priv/jobs/1057.atcgrid.SC: line 24: HelloOMP: command not found
/var/lib/torque/mom_priv/jobs/1057.atcgrid.SC: line 24: HelloOMP: command not found
/var/lib/torque/mom_priv/jobs/1057.atcgrid.SC: line 24: HelloOMP: command not found
[marta@marta-PC Ejemplo]$
```

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 ventana ssh he ejecutado:

[C1estudiante9@atcgrid hello]$ echo 'cat /proc/cpuinfo' | qsub -q ac

Y después he copiado el fichero .o obtenido a mi ordenador mediante sftp.

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: hay un procesador físico (el Intel ® Core™ i5-3350P) que tiene cuatro cores lógicos. Porque hay cuatro procesadores pero todos tieien el mismo physical id.

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

RESPUESTA: hay un procesador físico (Intel(R) Core(TM) i5-3337U) que tiene dos cores físicos. Cada core físico a su vez, tiene dos cores lógicos. Porque hay un sólo physical id, hay sólo dos valores para core id.

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

RESPUESTA: en atcgrid hay dos procesadores físicos (Intel(R) Xeon(R) CPU E5645) con seis cores físicos cada uno. Cada core físico a su vez, tiene dos cores lógicos. Porque hay dos valores distintos para physical id (0 y 1), seis valores distintos core id (0,1,2,8,9,10) y cada valor de core id se repite cuatro veces, dos por cada procesador físico.

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:

- **ncgt** contiene el tiempo de ejecución de la suma de vectores.

- La función **clock_gettime()** devuelve el tiempo del reloj del sistema en ese momento, por eso se ejecuta antes y después del momento en el que queremos medir el tiempo de ejecución y luego restamos.

- Los datos de la función clock_gettime() se devuelven en el **struct 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: Las principales diferencias observables en el código de ambas son:

- La forma de imprimir resultados (en C se usa la función printf y en C++ la función cout).
- La forma de tratar los vectores dinámicos: en C se usa la función malloc para reservar memoria y free para liberarla y en C++, se usa el operador new para reservar memoria y el delete [] para liberarla.

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: Ejecutamos el programa en C de la misma forma en la que ejecutamos el ejemplo de HelloOMP. Al finalizar comprobamos que el archivo de errores esté vacío:

```
[Clestudiante9@atcgrid hello]$ echo 'hello/SumaVectoresC 500' | qsub -q ac
3678.atcgrid
[Clestudiante9@atcgrid hello]$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
3678.atcgrid               STDIN            Clestudiante9   00:00:00 C ac

[Clestudiante9@atcgrid hello]$ ls -lag
total 72
drwxrwxr-x 2 Clestudiante9  4096 mar  5 09:23 .
drwx------ 6 Clestudiante9  4096 mar  4 15:43 ..
-r--r--r-- 1 Clestudiante9   703 mar  4 15:43 cpuinfo
-rw------- 1 Clestudiante9   336 feb 25 17:19 ejercicio3.e1057
-rw------- 1 Clestudiante9   325 feb 25 17:19 ejercicio3.o1057
-rwxr-xr-x 1 Clestudiante9  7392 feb 25 17:18 HelloOMP
-rw-rw-r-- 1 Clestudiante9   836 feb 25 17:18 script_helloomp.sh
-rw------- 1 Clestudiante9     0 mar  5 09:23 STDIN.e3678
-rw------- 1 Clestudiante9 29620 mar  5 09:23 STDIN.o3678
-rwxr-xr-x 1 Clestudiante9  8064 mar  5 09:13 SumaVectoresC
[Clestudiante9@atcgrid hello]$
```

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: Se obtiene un error de violación de segmento en ambos equipos (atcgrid y mi propio PC). El error se debe a que el tamaño del vector supera la capacidad del stack, en ambos equipos llega hasta el tamaño 262144 por limitaciones del lenguaje.

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 se obtiene error al ejecutar ninguno de los dos porque, en vectores globales cuando el tamaño de longitud del vector supera al máximo, se establece como valor dicho máximo. Por eso las últimas ejecuciones del programa con vectores globales dan los mismos resultados. Por otro lado, con vectores dinámicos no da error porque reserva la memoria que necesita, si el sistema operativo no dispone de la memoria que pedimos sí nos daría un error a la hora de asignar memoria.

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?

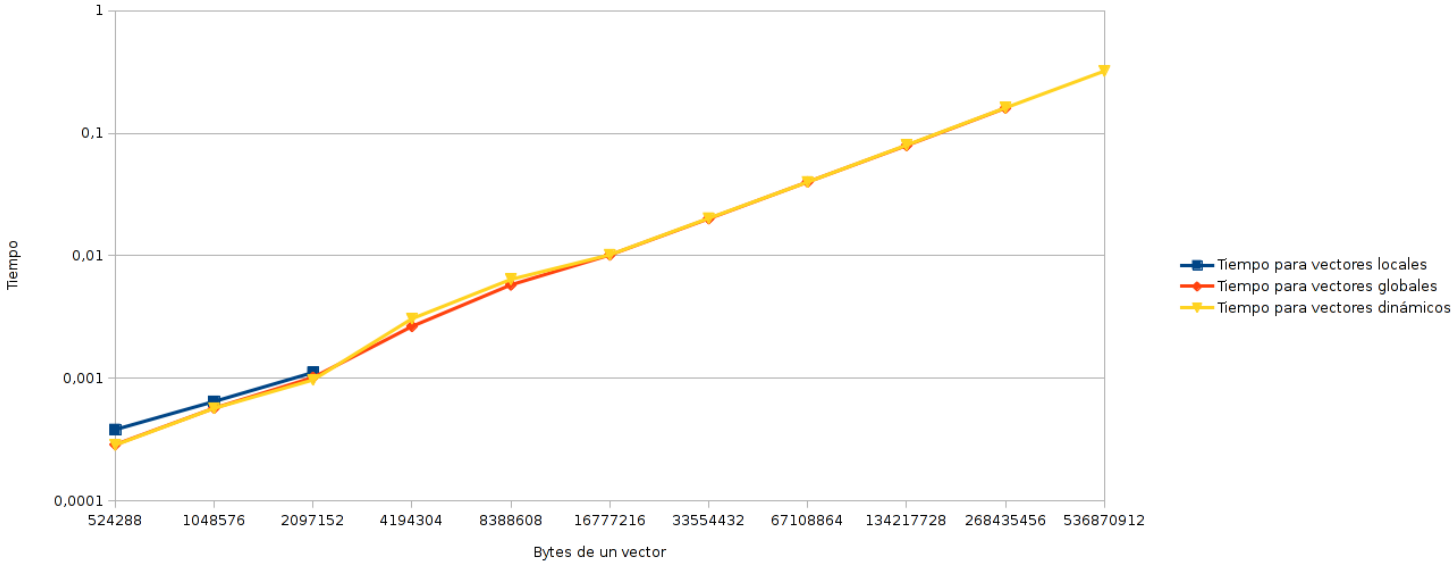
RESPUESTA:

Tabla 1 . Tiempos de ejecución de la suma de vectores para vectores locales, globales y dinámicos para mi PC

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000202943	0.000401601	0.000210011
131072	1048576	0.000570058	0.000616890	0.000755419
262144	2097152	0.002393286	0.001527466	0.001193045
524288	4194304		0.001974425	0.002453608
1048576	8388608		0.005565675	0.004369545
2097152	16777216		0.015999572	0.011322151
4194304	33554432		0.028680028	0.015496210
8388608	67108864		0.035551962	0.040378133
16777216	134217728		0.067111045	0.075788804
33554432	268435456		0.143934075	0.133089440
67108864	536870912			0.304047674

Tabla 2. Tiempos de ejecución de la suma de vectores para vectores locales, globales y dinámicos para el atcgrid

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000381452	0.000287485	0.000285507
131072	1048576	0.000646631	0.000570881	0.000569309
262144	2097152	0.001115528	0.001015508	0.000970508
524288	4194304		0.002652562	0.003065757
1048576	8388608		0.005796258	0.006422912
2097152	16777216		0.010168135	0.010196820
4194304	33554432		0.020127207	0.020270352
8388608	67108864		0.040101783	0.040045641
16777216	134217728		0.079602547	0.080178626
33554432	268435456		0.160743747	0.161486891
67108864	536870912			0.322627262



Como se puede apreciar, en tamaños pequeños del vector, los vectores locales son menos eficientes y los globales y dinámicos están muy igualados. Conforme va creciendo el tamaño del vector, los locales se quedan fuera de juego y los globales son un poco más rápidos que los dinámicos. Por último, tanto globales como dinámicos obtienen tiempos de ejecución muy parecidos para tamaños muy grandes.

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: El código proporcionado ya hace eso, así que modificarlo yo

misma no tendría mucho sentido. Lo que ocurre es que no genera fallos ya que cuando el valor introducido como longitud del vector sobrepasa el máximo, asigna al valor de dicha variable (N) el máximo tamaño que puede tener el vector que es 2^{32} - 1