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): Jose Miguel Hernández García
Grupo de prácticas: 2º C – C3
Fecha de entrega: 08-03-2017 (23:59:59)
Fecha evaluación en clase: 09-03-2017

Ejercicios basados en los ejemplos del seminario práctico

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

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

      RESPUESTA: Sirve para añadir a la cola un proceso a ejecutar, en concreto, la opción -q se utiliza para seleccionar el destino de trabajo.

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

      RESPUESTA: Mediante el comando `qstat` podemos comprobar si el proceso sigue aún en la cola o ha pasado a "Completado". Por otro lado, también es posible saberlo comprobando si se han creado los ficheros con extensión .o, que se trata del fichero de salida y el fichero de extensión .e que es el fichero de errores.

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

      RESPUESTA: Es posible saberlo porque el tamaño del fichero de errores .e es mayor a 0

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

      RESPUESTA: Mostrando lo que hay dentro del fichero de salida .o

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

      RESPUESTA: Porque disponemos de 24 procesadores lógicos, ya que existen dos procesadores físicos, con 6 núcleos cada uno y cada núcleo dispone de 2 hilos, mostrando así un mensaje por cada hilo.

2. En el segundo ejemplo de ejecución en atcgrid usando TORQUE el script `script_helloomp.sh` 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 el seminario. Conteste a las siguientes preguntas:

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

      RESPUESTA: No lo acompaña ya que en el script que hemos utilizado ya viene incluida la opción -q.

   b. ¿Cuántas veces ejecuta el `script` el ejecutable `HelloOMP` en atcgrid? ¿Por qué lo ejecuta ese número de veces?

      RESPUESTA: Se ejecuta 4 veces dado que el bucle for que se utiliza en este ejercicio realiza la iteración 4 veces. En la primera iteración se usan 12 threads, en la segunda 6, en la tercera 3 y en la cuarta 1.

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

      RESPUESTA: Se imprimen en total 22 mensajes, ya que en cada ejecución del script se llama al ejecutable 4 veces, pintando primero 12, luego 6, luego 3 y luego 1.

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: Para imprimir la variable de entorno `$PBS_O_WORKDIR` he añadido al script la siguiente linea:

echo "Directorio de trabajo: `$PBS_O_WORKDIR`"

Tras ejecutar el script habiendo modificado esta línea, este da error puesto que no se puede ejecutar `HelloOMP`, dado que no se encuentra.

Resto de ejercicios

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), y del PC del aula de prácticas o de su PC. Indique qué ha hecho para obtener el contenido de `/proc/cpuinfo` en atcgrid.

RESPUESTA: Para obtener el contenido de `/proc/cpuinfo` en el atcgrid, he utilizado el siguiente comando: `echo 'cat /proc/cpuinfo' | qsub -q ac.`

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 o su PC?

RESPUESTA: Si observamos el campo `cpu cores` nos indica que dispone de 4 cores físicos y dado que disponemos de hyperthreading, tenemos 8 cores lógicos (4x2).

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

RESPUESTA: Al igual que en el apartado anterior, observando el campo `cpu cores` vemos que disponemos de 12 cores físicos pues tiene 2 procesadores y en cada procesador hay 6 cores y dado que se dispone de hyperthreading, cada core dispone de 2 hilos, por lo qué disponemos de un total de 24 cores lógicos (2x6x2).

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`.

a. 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 (indicar el tipo de estructura de datos y describir la estructura de datos)?

RESPUESTA: En la variable `ncgt` almacenamos la diferencia entre la llamada cgt1 al empezar el bucle y la variable cgt2 al terminar para así poder conocer el tiempo que ha tardado.

La función `clock_gettime()` devuelve el valor de un instante de tiempo y lo almacena en un struct que contiene los segundos y nanosegundos en los que se llamó a dicha función.

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:

Descripción diferencia	En C	En C++
Salida de datos	Función `printf()`	`cout <<`
Definir tam. vectores	Función `malloc()`	`new double ()`
Liberar espacio vectores	Función `free()`	Función `delete[]`
Incluir cabeceras	`nombre_cabecera.h`	`nombre_cabecera`
namespaces	no	sí

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: Para ejecutar el programa en actgrid he recurrido a la siguiente orden:

`echo './SumaVectoresC 5' | qsub -q ac` .Obteniendo el siguiente resultado:

```
[C3estudiante9@atcgrid ~]$ echo './SumaVectoresC 5' | qsub -q ac
45514.atcgrid
[C3estudiante9@atcgrid ~]$ qstat
Job ID                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
45514.atcgrid             STDIN            C3estudiante9   00:00:00 C ac
[C3estudiante9@atcgrid ~]$ ll
total 16
-rw------- 1 C3estudiante9 C3estudiante9    0 mar  7 18:57 STDIN.e45514
-rw------- 1 C3estudiante9 C3estudiante9  147 mar  7 18:57 STDIN.o45514
-rwxrwxr-x 1 C3estudiante9 C3estudiante9 8888 mar  7 18:53 SumaVectoresC
[C3estudiante9@atcgrid ~]$ cat STDIN.o45514
Tiempo(seg.):0.000000160        / Tamaño Vectores:5    / V1[0]+V2[0]=V3[0](0.500000+0.500000=1.000000) / / V1[4]+V2[4]=V3[4](0.900000+0.100000=1.000000) /
[C3estudiante9@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: Esta vez ejecutamos el script en lugar del programa utilizando la orden:

`echo './SumaVectores.sh' | qsub -q ac`. Obteniendo el siguiente resultado:

```
428.700000+0.100000=52428.800000) /
[C3estudiante9@atcgrid ~]$ cat STDIN.e45518
./SumaVectores.sh: line 22: 20524 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
./SumaVectores.sh: line 22: 20527 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
./SumaVectores.sh: line 22: 20533 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
./SumaVectores.sh: line 22: 20538 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
./SumaVectores.sh: line 22: 20543 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
./SumaVectores.sh: line 22: 20549 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
./SumaVectores.sh: line 22: 20555 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
./SumaVectores.sh: line 22: 20560 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
[C3estudiante9@atcgrid ~]$
```

Vemos que se produce una violación de segmento dado que se ha desbordado la pila del programa, en concreto solo se ejecuta para los siguientes valores:

```
Tiempo(seg.):0.000381090        / Tamaño Vectores:65536        / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) / / V1[65535]+V2[65535]=V3[65535](13107.1
00000+0.100000=13107.200000) /
Tiempo(seg.):0.000765560        / Tamaño Vectores:131072       / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.400000) / / V1[131071]+V2[131071]=V3[131071](26
214.300000+0.100000=26214.400000) /
Tiempo(seg.):0.001378727        / Tamaño Vectores:262144       / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.800000) / / V1[262143]+V2[262143]=V3[262143](52
428.700000+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: No se obtiene error dado que en esta ocasión, las variables globales se almacenan en la zona de datos del programa y las variables dinámicas en el heap, y no en la pila como las variables locales.

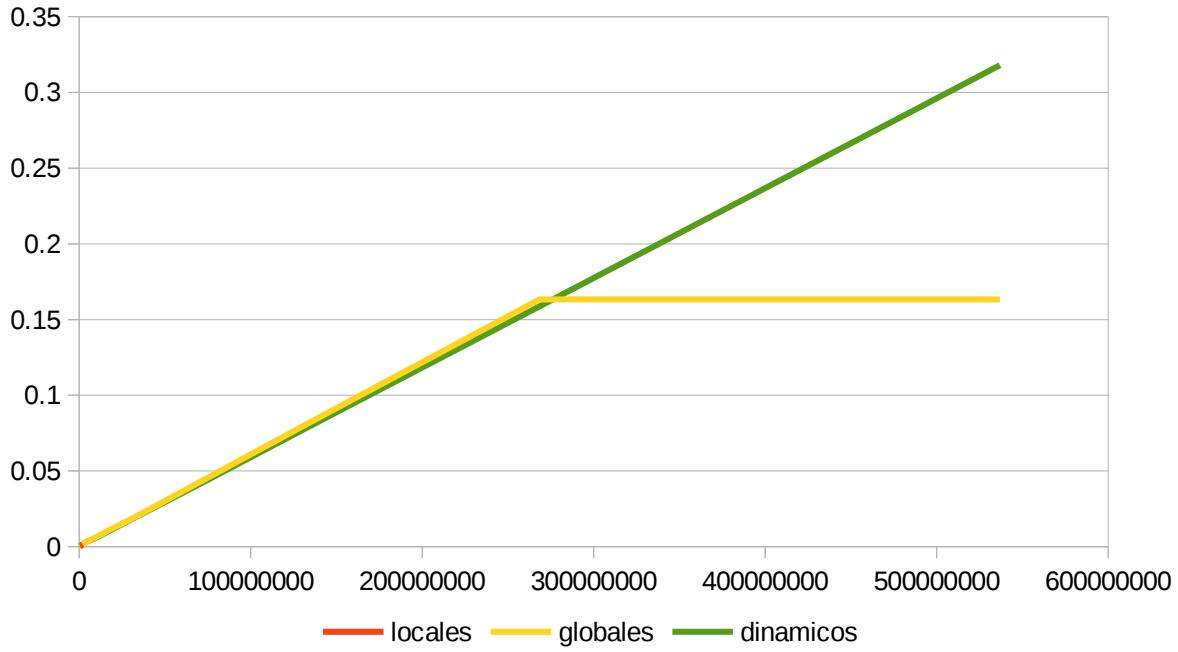
9. Rellenar una tabla como la Tabla 1 para atcgrid y otra para el PC local con los tiempos de ejecución obtenidos en los ejercicios anteriores para el trozo de código que realiza la suma de vectores. En la columna “Bytes de un vector” hay que poner el total de bytes reservado para un vector. Ayudándose 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). Utilice escala logarítmica en el eje de ordenadas (eje y) en todas las gráficas. ¿Hay diferencias en los tiempos de ejecución con vectores locales, globales y dinámicos?

RESPUESTA: Si que existe diferencia, pero no es mucha, casi inapreciable.

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

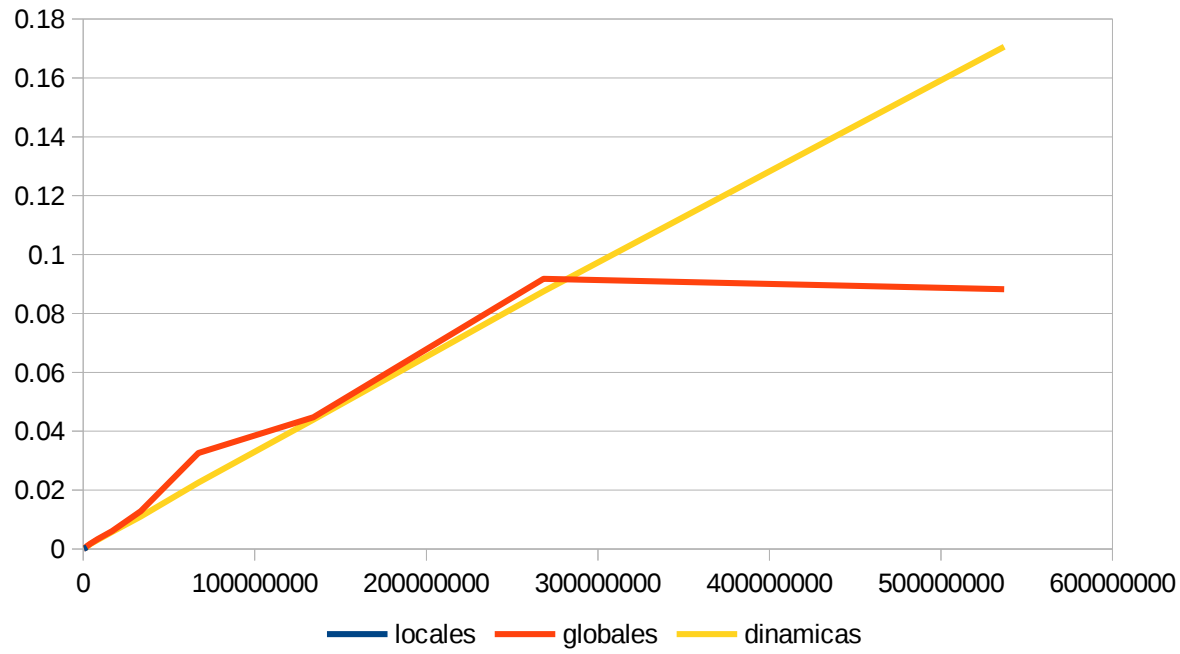
Para atcgrid:

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000411069	0.000386087	0.000341375
131072	1048576	0.000740459	0.000732678	0.000765106
262144	2097152	0.001491743	0.000970516	0.001481198
524288	4194304		0.003178448	0.003158195
1048576	8388608		0.005282087	0.005023932
2097152	16777216		0.010510013	0.010056216
4194304	33554432		0.020030447	0.019883984
8388608	67108864		0.040693417	0.039517264
16777216	134217728		0.081828097	0.079397642
33554432	268435456		0.163252464	0.158678541
67108864	536870912		0.163228040	0.317901719



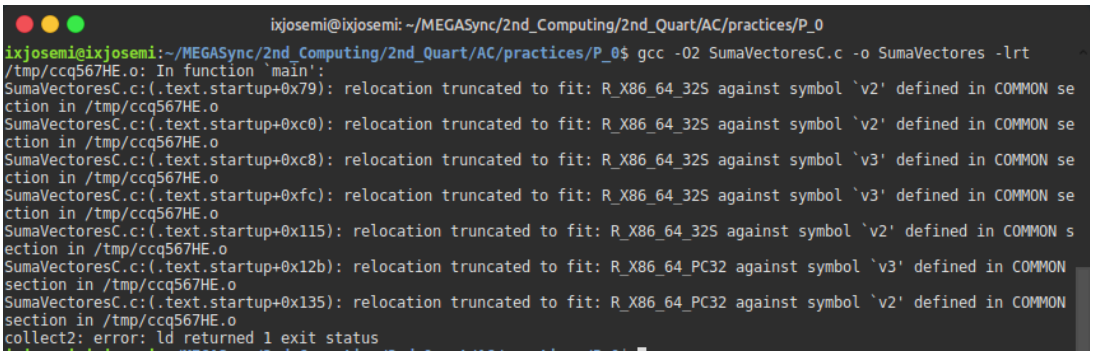
Para PC local:

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000164535	0.000164370	0.000151977
131072	1048576	0.000304390	0.000227306	0.000319473
262144	2097152	0.000635716	0.000955098	0.000724557
524288	4194304		0.001763144	0.001714927
1048576	8388608		0.003317063	0.003031913
2097152	16777216		0.006020259	0.005758548
4194304	33554432		0.012844033	0.011022017
8388608	67108864		0.032645370	0.022532635
16777216	134217728		0.044778539	0.043884124
33554432	268435456		0.091781229	0.087515813
67108864	536870912		0.088275471	0.170556654



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: tras modificar la linea `#define MAX 4294967295 //=2^32 - 1` y compilar el programa utilizando variables globales como se indica, al ejecutar el programa obtenemos el siguiente error:




```
relocation truncated to fit: R_X86_64_32S against symbol `v2' defined
in COMMON section in /tmp/ccq567HE.o
```

Que nos indica que hemos sobrepasado el tamaño permitido y el compilador detecta el desbordamiento. El número máximo que se puede almacenar en N es $2^{32} - 1$ porque es el máximo número que se puede representar en un dato de tipo entero de 32 bits (Es necesario restarle 1 porque hay que tener en cuenta el 0).