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): Emilio Chica Jiménez
Grupo de prácticas: B2
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: Establece el destino del trabajo. El destino nombra una cola, un servidor, o una cola en un servidor. La hemos enviado a la cola AC

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

RESPUESTA: Con qstat te permite saber el estado de tu trabajo

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

RESPUESTA: Con el fichero de salida que comienza por extension .e

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

RESPUESTA: La salida del programa aparece en un archivo que comienza por la extensión .o

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

RESPUESTA: Por que al mandar el trabajo a la cola lo ejecuta uno de los nodos atgrid en sus correspondientes 6 nucleos fisicos del xenon, 6 núcleos virtuales de la tecnolgía hyperthreading y como el nodo tiene dos procesadores pues hay tenemos los 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: Por que no se manda a la cola AC, sino que se se manda a la cola por defecto.

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

RESPUESTA: Se ejecuta 12 la primera vez, 6 la segunda vez, 3 la segunda vez y una vez la ultima vez. Se ejecuta ese numero de veces porque establece en la variable export OMP_THREAD_LIMIT=12.

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

RESPUESTA: Se ejecuta 12 la primera vez, 6 la segunda vez, 3 la segunda vez y una vez la ultima vez. Se ejecuta ese numero de veces porque establece en la variable export OMP_THREAD_LIMIT=12.

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: No se ejecuta HelloOMP y te muestra el directorio actual.

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: 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?

RESPUESTA: 4 y 4

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

RESPUESTA: 2 y 2

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

RESPUESTA: 6 físicos en un procesador otros 6 físicos en otro procesador y cada uno de estos utiliza hyperthreading para tener otros 6 logicos cada uno. En total 24, 12 y 12.

RESPUESTA: El fichero de captura se llama EJERCICIO 3 2015-03-05 12:52:32

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:

- El tiempo de ejecución de:

```
for(i=0; i<N; i++)
    v3[i] = v1[i] + v2[i];
```

Es decir la diferencia entre la toma de tiempo inicial que se guarda en la variable cgt1 y el tiempo final después de haber ejecutado la función en concreto guardado en la variable cgt2. Primero estimado en segundos y luego estimado en nanosegundos.

- El tiempo de ejecución de un trozo de codigo de un programa, en este caso el de la suma de vectores.
- Se devuelve en una estructura del tipo 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: Se usa malloc en lugar de new, en c++ se utilza cout en lugar de printf, c++ utiliza delete para liberar el espacio de memoria reservado previamente y c se utiliza free.

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:

```
echo "hello/SumaVectores 2" | qsub -q ac
```

El fichero de salida se llama SALIDA EJERCCIO 6.

El contenido es:

Tiempo(seg.):0.000000104 / Tamaño Vectores:2 / V1[0]+V2[0]=V3[0] (0.200000+0.200000=0.400000) // V1[1]+V2[1]=V3[1](0.300000+0.100000=0.400000) /

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:

- A partir del tamaño 262144 se obtiene error tanto en local como en el atcgrid1.
- Porque ha superado el tamaño de la pila del programa.

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, debido a que con los vectores globales no dependemos del tamaño de la pila del programa y con los vectores dinamicos se aumenta el tamaño automaticamente.

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: Para valores muy altos de bytes de un vector si hay diferencia con respecto a los vectores dinámicos y los globales, se ve que el rendimiento decrece en los dinámicos.

- PC-LOCAL

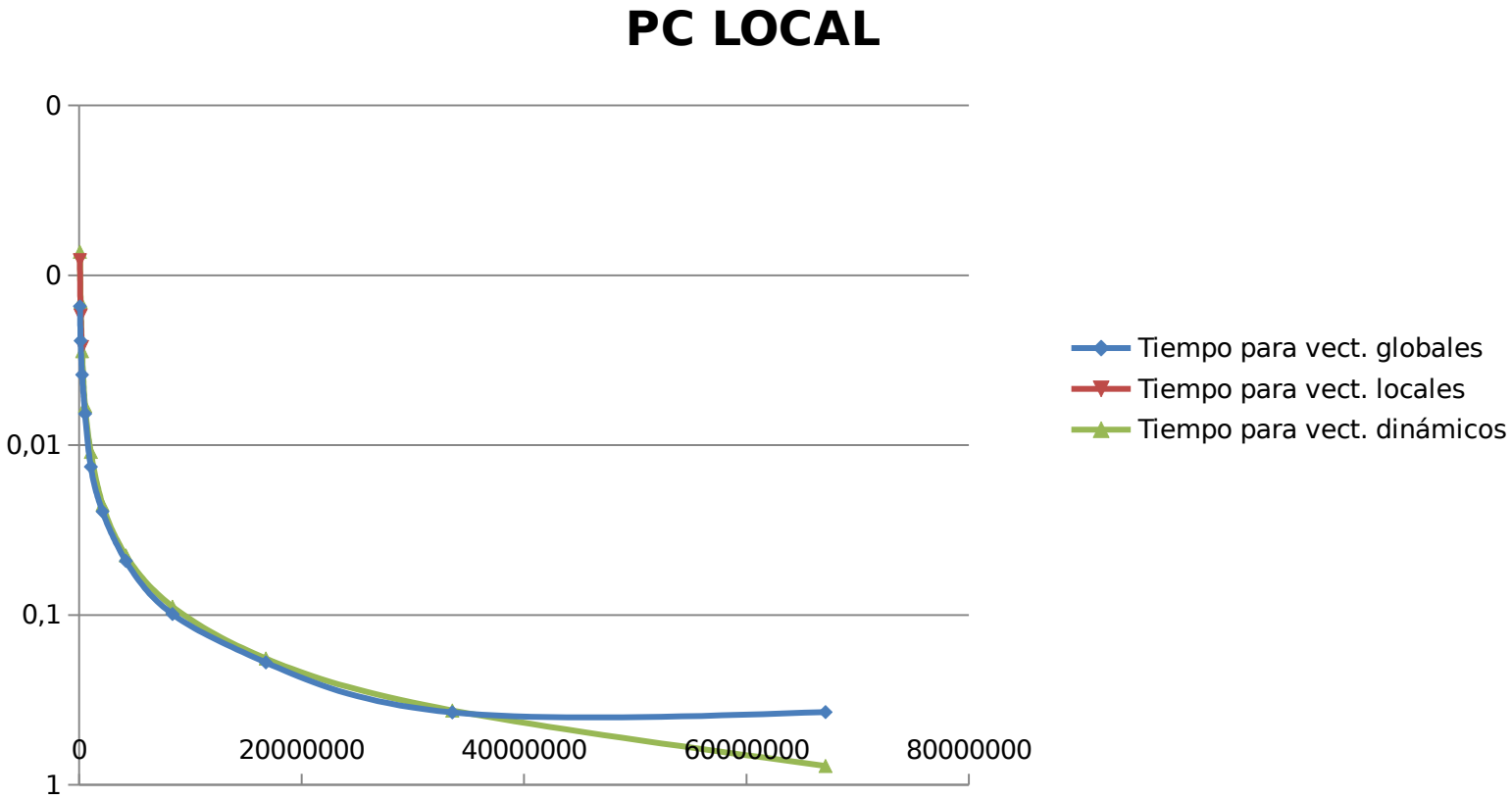
Tabla 1 . Tiempos de ejecución de la suma de vectores para vectores locales, globales y dinámicos

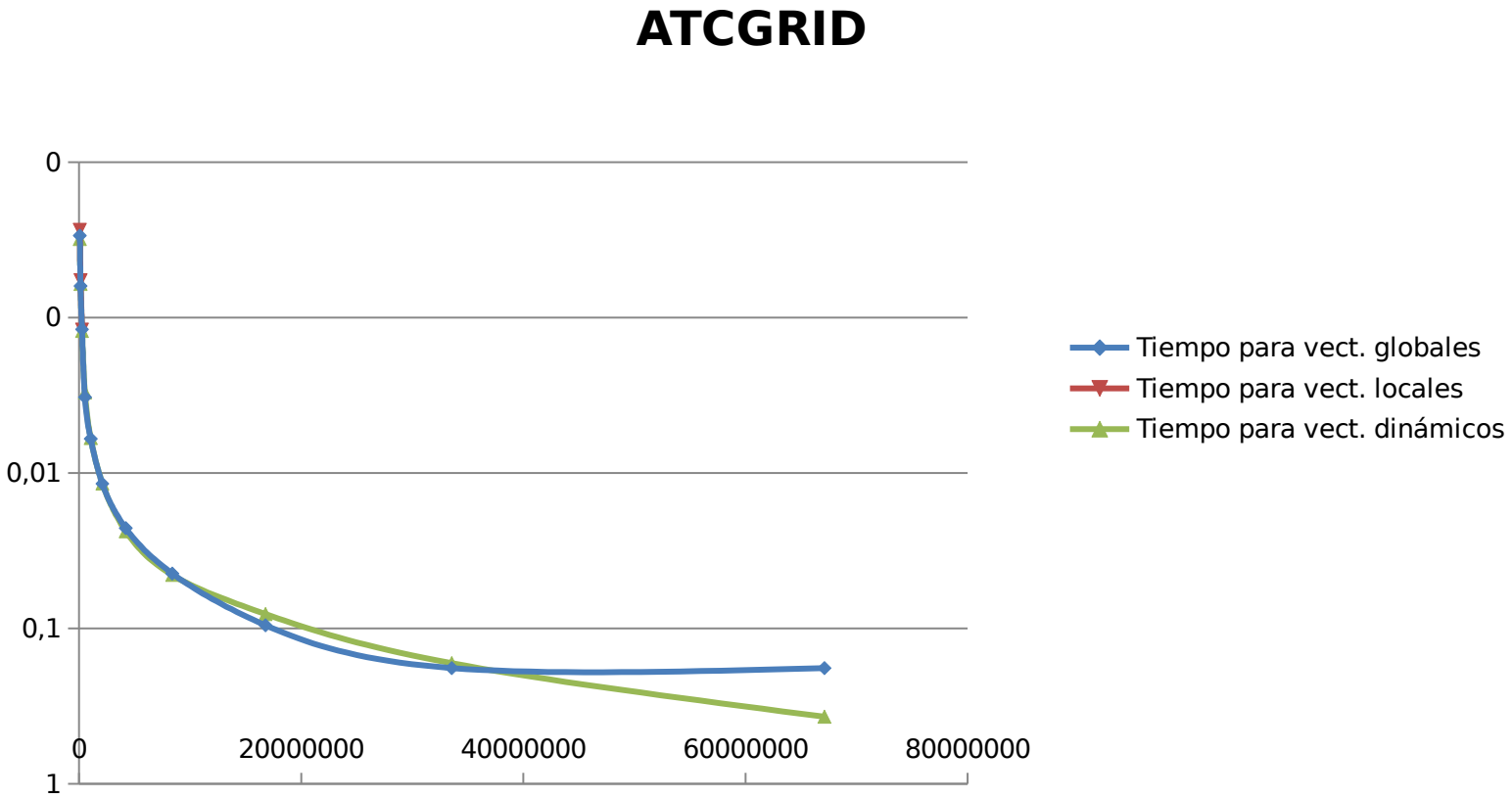
Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	65536	0.000815327	0.001522890	0.000726488
131072	131072	0.001727177	0.002430618	0.001432165
262144	262144	0.002643566	0.003854474	0.002795814
524288	524288	-	0.006529599	0.005871420
1048576	1048576	-	0.013414204	0.010971856
2097152	2097152	-	0.024573238	0.022309892
4194304	4194304	-	0.048042422	0.044575849
8388608	8388608	-	0.098822692	0.089224823
16777216	16777216	-	0.190151065	0.180305329
33554432	33554432	-	0.374161758	0.364552084
67108864	67108864	-	0.372603316	0.772840251

- ATCGRID

Tabla 2 . 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	65536	0.000272595	0.000296468	0.000311816
131072	131072	0.000576196	0.000625144	0.000606821
262144	262144	0.001196051	0.001190939	0.001217061
524288	524288	-	0.003253252	0.003026856
1048576	1048576	-	0.006017237	0.005955137
2097152	2097152	-	0.011700117	0.011655401
4194304	4194304	-	0.022612746	0.023654297
8388608	8388608	-	0.044290250	0.045311096
16777216	16777216	-	0.095294318	0.080768360
33554432	33554432	-	0.179725736	0.167370244
67108864	67108864	-	0.180132726	0.369762453


PC LOCAL
0
0
0,01
0,1
1
0
20000000
40000000
60000000
80000000
Tiempo para vect. globales
Tiempo para vect. locales
Tiempo para vect. dinámicos



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:

Muestra el error siguiente:

> reubicación truncada para ajustar: R_X86_64_32S contra el símbolo `v2' definido en la sección COMMON en /tmp/cc6yfqKY.o

Por que no es capaz de ubicar el maximo direccionable en 32 bits y alinear los datos. El máximo numero que se puede almacenar en 32 bits es 2^{32}-1 por que 2^{32} es el número -1, es decir ya es un numero negativo.