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): Miguel Ángel Robles Urquiza
Grupo de prácticas: A1
Fecha de entrega: 09/03/2017
Fecha evaluación en clase: 10/03/2017

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: Para indicar la cola en la que se va a trabajar

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

   RESPUESTA: Nos lo facilita la orden qstat ya que no hay esta en cola, está completado y se han creado los archivos de salida y errores

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

   RESPUESTA: Mirando el tamaño del archivo STDIN.e*

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

   RESPUESTA: Mirando el contenido del archivo STDIN.o*

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

   RESPUESTA: Depende de el número de procesadores, cada mensaje sale por un thread

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: Porque está indicado ya en el script

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

   RESPUESTA: 4 veces. Se debe a que lo ejecuta una vez con 12 threads, una segunda vez con 6, una tercera vez con 3 y por ultimo con 1.

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

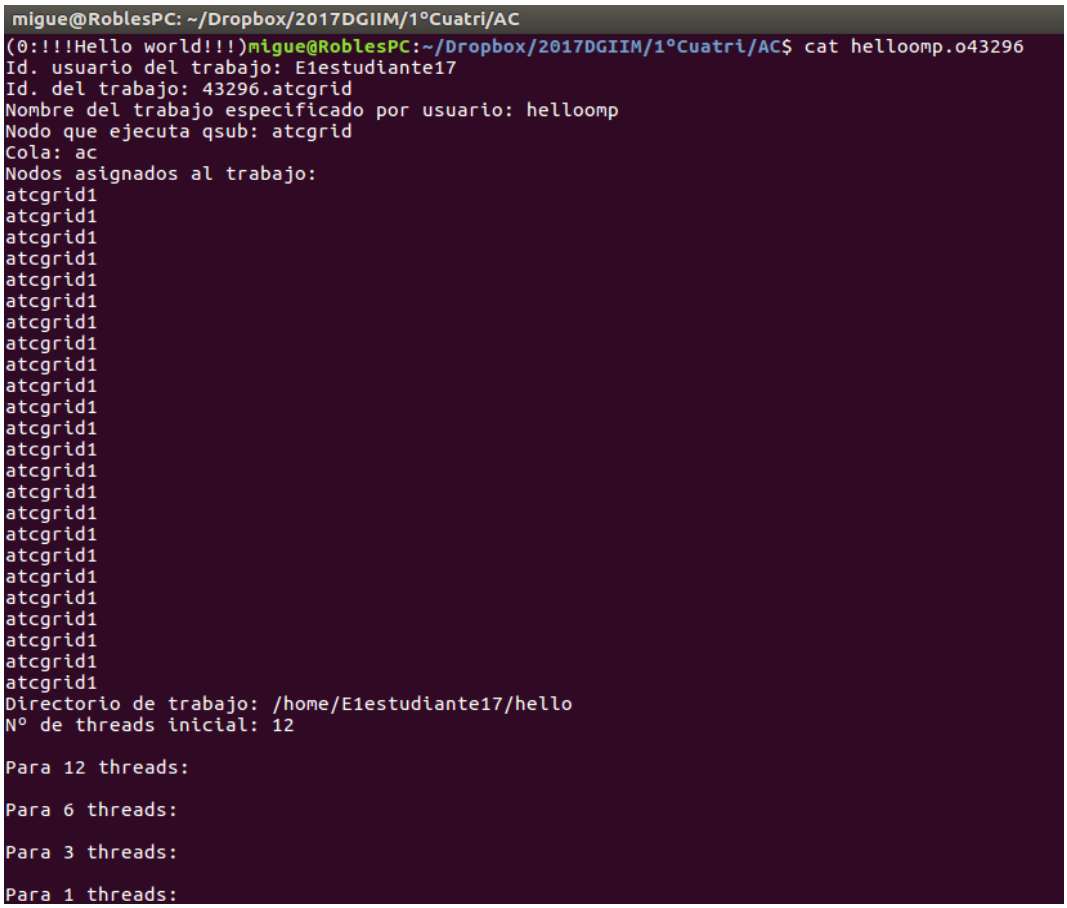
   RESPUESTA:

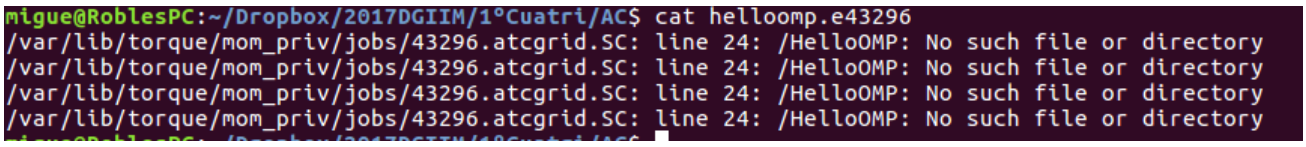
3. Realizar las siguientes modificaciones en el `script` "¡¡¡Hello World!!!":

- 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: He añadido un echo que muestra la variable, que es el directorio de trabajo. Se obtiene un error ya que no se puede encontrar el ejecutable al quitar la ruta en el guion





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: Ejecutando la orden 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: Vemos que tenos 4 procesadores lógicos y 2 físicos

Precessor: 0-3

Core ID: 0,0,1,1

Physical ID: 0

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

RESPUESTA: Vemos que tenemos 24 procesadores lógicos y 12 físicos

Precessor: 0-23

Core ID: 0,1,2,8,9,10

Physical ID: 0,1

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 ncgt guardamos la diferencia entre cgt1 antes del bucle y

la variable cgt2 al finalizarlo, es decir, el tiempo que ha tardado.

La funcion clock_gettime() obtiene el valor de un instante de tiempo, lo

almacena en una estructura que tiene los segundos y nanosegundos en el que se llamó a la

función. Se guarda en la estructura timespec:

```
struct timespec {
 time_t tv_sec; /* seconds */
 long tv_nsec; /* nanoseconds */
};
```

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++
Namespace	No hay	std
Cabeceras	.h	
Liberar espacio de vectores	free()	delete []
Reservar memoria para vectores	malloc()	new double[]
Output	printf()	cout <<

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:

```
E1estudiante17@atcgrid ~]$ echo './SumaVectores 1000'|qsub -q ac
5940.atcgrid
```

```
nigue@RoblesPC:~/Dropbox/2017DGIIM/2ºCuatri/AC$ cat STDIN.045940
Tiempo(seg.):0.000004094        Tamaño Vectores:1000            V1[0]+V2[0]=V3[0](100.000000+100.000000=200.000000)  V1[999]+V2[999]=V3[999](199.9
00000+0.100000=200.000000)
```

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?

1.

RESPUESTA: Si, a partir de 524288 componentes vemos que se agota la pila por estar declaradas de forma local las variables, puesto que en la ejecución del programa nos muestra en salida hasta ese numero y en errores vemos que el resto son segmentation fault

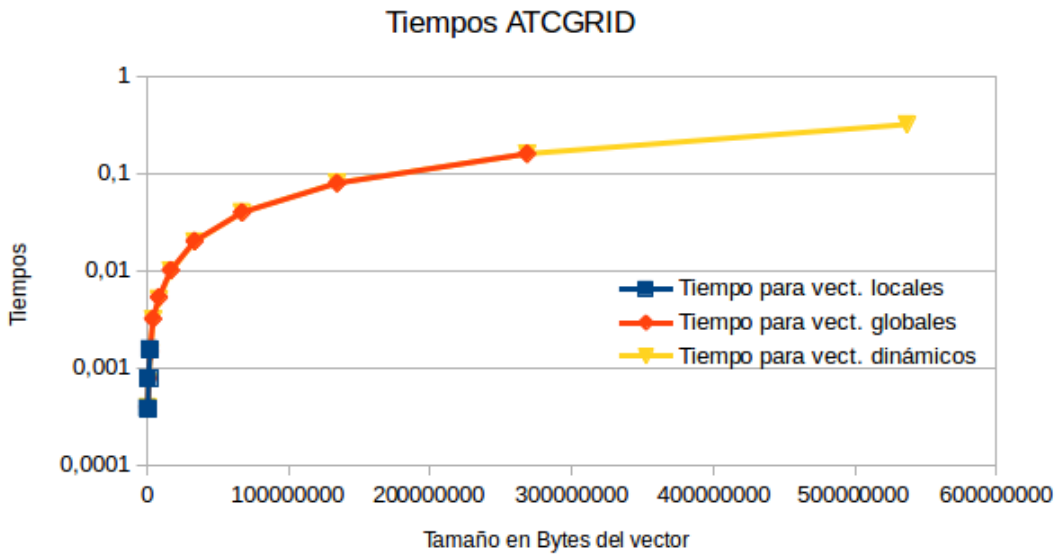
8. Generar los ejecutables del código fuente C para vectores globales y para dinámicos. Genere el ejecutable usando –O2. Ejecutar los dos códigos en atcgrid usando un `script` como el del Listado 3 (hay que poner en el `script` el nombre de los ficheros ejecutables generados en este ejercicio) para el 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: Vemos que hay un solo error, que es cuando lo hacemos con variables globales no podemos tomar el ultimo valor al reescribirse el MAX en el código e SumaVectores.c, es decir, no tenemos información sobre 67108864

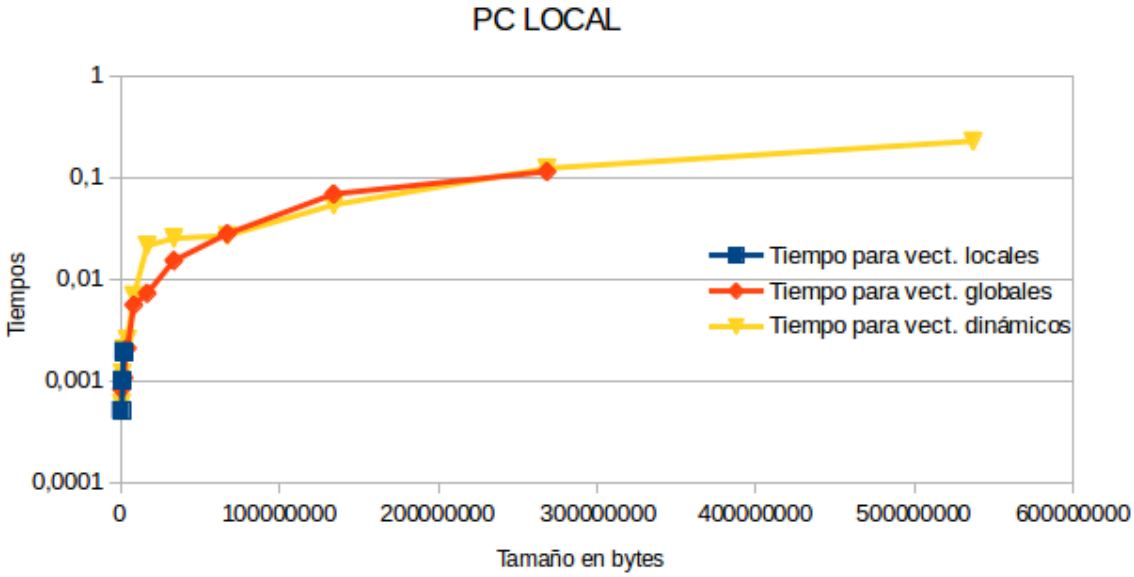
9. Rellenar una tabla como la Tabla 1 para atcgrid y otra para el PC local con los tiempos de ejecución obtenidos en los ejercicios anteriores para el trozo de código que realiza la suma de vectores. En la columna "Bytes de un vector" hay que poner el total de bytes reservado para un vector. Ayudándose 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 hay diferencia, pero mínima.

ATCGRID				
Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0,000386355	0,000400009	0,000361063
131072	1048576	0,000766145	0,000779107	0,000717760
262144	2097152	0,001536715	0,001448733	0,001263463
524288	4194304		0,003050449	0,003211862
1048576	8388608		0,005235915	0,005461788
2097152	16777216		0.010008117	0,010516348
4194304	33554432		0,019798417	0,021049031
8388608	67108864		0,039201302	0,041479740
16777216	134217728		0,078858601	0,084620752
33554432	268435456		0,157742114	0,171314071
67108864	536870912			0,318363207



PC-LOCAL				
Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0,000502542	0,000820995	0,000583564
131072	1048576	0,000998746	0,000882999	0,001180071
262144	2097152	0,001918289	0,001066609	0,002030931
524288	4194304		0,002073639	0,002534430
1048576	8388608		0,005514314	0,006855770
2097152	16777216		0,007199317	0,021185240
4194304	33554432		0,015109045	0,024969652
8388608	67108864		0,027699273	0,026590113
16777216	134217728		0,068241117	0,053454990
33554432	268435456		0,113448531	0,122197198
67108864	536870912			0,226140368



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: Le adjunto el archivo ya modificado. Me da error al compilar ya que el compilador detecta que hay un desbordamiento, el maximo numero a almacenar en N es porque es el máximo numero que se puede representar en un entero de 32bits.

```
migue@RoblesPC:~/Dropbox/2017DGIIM/2ºCuatri/AC$    gcc -O2 SumaVectoresCMax.c -o SumaVectores -lrt
/tmp/cchieGu2.o: En la función `main':
SumaVectoresCMax.c:(.text.startup+0x79): reubicación truncada para ajustar: R_X86_64_32S contra el símbolo `v2' definido en la sección COMMON en /tmp/cchieGu2.o
SumaVectoresCMax.c:(.text.startup+0xc0): reubicación truncada para ajustar: R_X86_64_32S contra el símbolo `v2' definido en la sección COMMON en /tmp/cchieGu2.o
SumaVectoresCMax.c:(.text.startup+0xc8): reubicación truncada para ajustar: R_X86_64_32S contra el símbolo `v3' definido en la sección COMMON en /tmp/cchieGu2.o
SumaVectoresCMax.c:(.text.startup+0xfc): reubicación truncada para ajustar: R_X86_64_32S contra el símbolo `v3' definido en la sección COMMON en /tmp/cchieGu2.o
SumaVectoresCMax.c:(.text.startup+0x115): reubicación truncada para ajustar: R_X86_64_32S contra el símbolo `v2' definido en la sección COMMON en /tmp/cchieGu2.o
SumaVectoresCMax.c:(.text.startup+0x12b): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v3' definido en la sección COMMON en /tmp/cchieGu2.o
SumaVectoresCMax.c:(.text.startup+0x135): reubicación truncada para ajustar: R_X86_64_PC32 contra el símbolo `v2' definido en la sección COMMON en /tmp/cchieGu2.o
collect2: error: ld returned 1 exit status
```