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): Ernesto Serrano Collado
Grupo de prácticas: A2
Fecha de entrega: 9/3/2016
Fecha evaluación en clase: 10/3/2016

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 indicarle el destino del trabajo, nosotros siempre usaremos el destino " ac"

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

   RESPUESTA: Porque al ejecutar "qstat" no aparece el trabajo en cola, y le ha puesto el estado "C" de "Completed", y porque se han generado el fichero de salida `.o*` y de errores `.e*`

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

   RESPUESTA: Porque el tamaño del fichero de errores `.e*` es mayor que 0

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

   RESPUESTA: Mostrando el contenido del fichero de salida `.o*`

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

   RESPUESTA: Porque tenemos 24 procesadores lógicos (2 Procesadores físicos con 6 nucleos y dos hilos de ejecución por núcleo 2x6x2), y se muestra 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: porque ya se indica en el script como orden **`#PBS -q ac`**

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

   RESPUESTA: Pues en total ejecuta 4 veces, en la primera vez usa 12 threads, en la segunda 6 threads, en la tercera 3 threads y en la última 1 thread.

   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 como hemos dicho antes, en cada ejecución del script se llama al ejecutable 4 veces, pintando 12, 6, 3 y 1 respectivamente.

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 pintado la variable `$PBS_O_WORKDIR` agregando la línea: echo `"Directorio trabajo: $PBS_O_WORKDIR"`

Al quitar la variable de donde estaba definida el script da error ya que no encuentra el ejecutable `HelloOMP` al no definirle el directorio de trabajo

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 los datos he lanzado el 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: 2 cores físicos, 4 lógicos

Se puede saber por el campo `cpu cores : 2` de la información de `cpuinfo`, teniendo en cuenta que tiene hyperthreading tiene 4 lógicos (que es el numero de procesadores que pinta), 2x2=4

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

RESPUESTA: 12 cores físicos, 24 lógicos

Se puede saber por el campo `cpu cores : 6` de la información de `cpuinfo`, teniendo en cuenta que tiene 2 procesadores y 24 lógicos porque cada core tiene hyperthreading y pinta 24 entradas en el cpuinfo, 2x6x2=24

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: 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, en la variable `ncgt` guardamos la diferencia entre la llamada `cgt1` al iniciar el bucle y la variable `cgt2` al finalizarlo, así sabemos el tiempo que ha tardado.

La estructura timespec donde se almacena está definida en la cabecera `time.h` y es la siguiente:

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

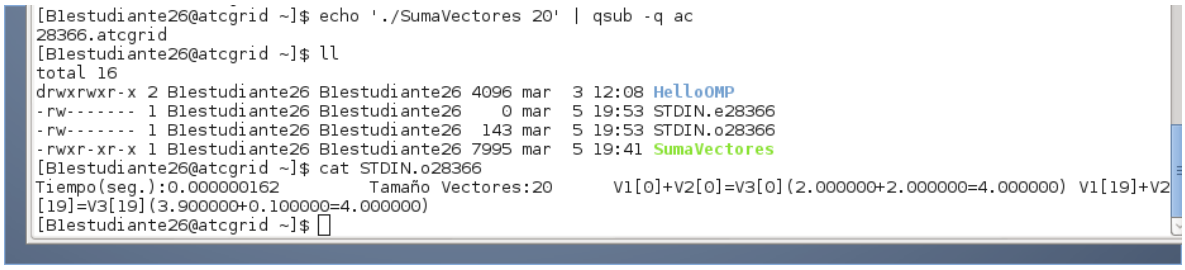
b. Escribir en el cuaderno de prácticas las diferencias que hay entre el código fuente C y el código fuente C++ para la suma de vectores.

RESPUESTA:

Descripción diferencia	En C	En C++
Salida de datos	printf()	cout <<
Definir tamaño de vectores	malloc()	new double[]
Liberar espacio de vectores	free()	delete []
Cabeceras	.h	
Espacios de nombres	no	si

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: Hemos compilado, subido y lanzado el comando `echo './SumaVectores 20' | qsub -q ac` vemos que nos ha dado resultado en la salida



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: Ejecutamos el script con echo `'./SumaVectores.sh' | qsub -q ac` vemos que nos da errores, adjunto captura del error, violación de segmento porque hemos desbordado el tamaño de la pila del programa

```
[B1estudiante26@atcgrid ~]$ cat STDIN.e28376
./SumaVectores.sh: line 20: 12525 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectores $N
./SumaVectores.sh: line 20: 12527 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectores $N
./SumaVectores.sh: line 20: 12530 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectores $N
./SumaVectores.sh: line 20: 12534 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectores $N
./SumaVectores.sh: line 20: 12536 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectores $N
./SumaVectores.sh: line 20: 12538 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectores $N
./SumaVectores.sh: line 20: 12540 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectores $N
./SumaVectores.sh: line 20: 12542 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectores $N
```

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, ya que las variables no se almacenan en la pila, sino en el heap o en la zona de datos del programa.

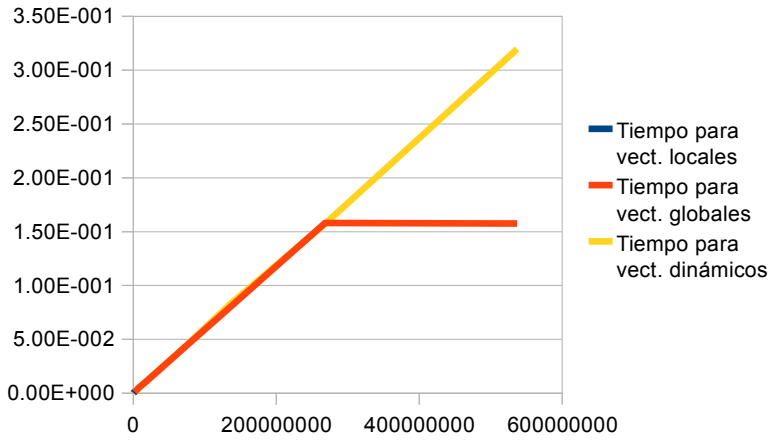
9. Rellenar una tabla como la Tabla 1 para atcgrid y otra para el PC local con los tiempos de ejecución obtenidos en los ejercicios anteriores para el trozo de código que realiza la suma de vectores. En la columna “Bytes de un vector” hay que poner el total de bytes reservado para un vector. Ayudándose 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 no parece apreciable, quizá habría que lanzar una tarea mas grande y mas optimizada, curiosamente mi pc me da unos tiempos superiores al atcgrid

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

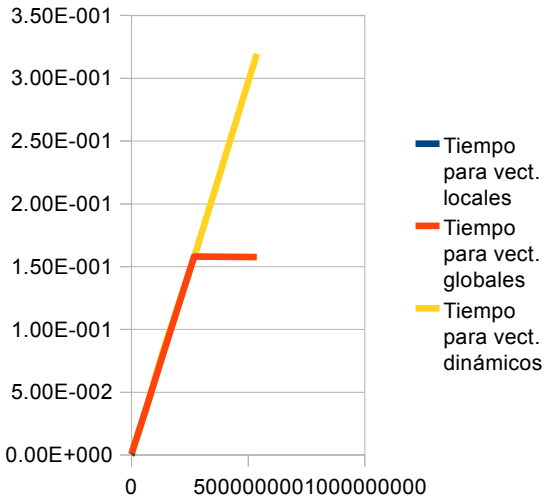
Tiempos en atcgrid

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000394877	0.000318923	0.000288084
131072	1048576	0.000686455	0.000764645	0.000791556
262144	2097152	0.001361396	0.001081361	0.001116657
524288	4194304		0.002646926	0.002521691
1048576	8388608		0.005454864	0.005099480
2097152	16777216		0.010030501	0.010067576
4194304	33554432		0.019807976	0.019975718
8388608	67108864		0.039730546	0.039581123
16777216	134217728		0.079151049	0.081805767
33554432	268435456		0.158100366	0.157304922
67108864	536870912		0.157520558	0.319538016



Tiempos en pc local

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000249858	0.000279680	0.000672138
131072	1048576	0.000503466	0.000521844	0.001200938
262144	2097152	0.001054091	0.000720745	0.001666253
524288	4194304		0.001592325	0.002395407
1048576	8388608		0.002458871	0.004349218
2097152	16777216		0.005345533	0.006476733
4194304	33554432		0.009303243	0.008886412
8388608	67108864		0.017395817	0.018960394
16777216	134217728		0.035401294	0.035364488
33554432	268435456		0.070913345	0.069949037
67108864	536870912		0.073368553	0.138614359



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: Hemos cambiado el codigo por `#define MAX 4294967295 //=2^32-1` , lo que courre es que al compilar me da el siguiente error: `SumaVectoresGmax.c: (.text.startup+0x66): relocation truncated to fit: R_X86_64_32S against symbol `v2' defined in COMMON section in /tmp/ccnWVJhb.o` esto es debido a que nos pasamos del tamaño y el compilador detecta que hay un desbordamiento, el maximo numero a almacenar en `N` es porque 2^{32}-1 es el máximo numero que se puede representar en un entero de 32bits y se le resta 1 a 2^{32} porque hay que tener en cuenta el 0.