Ejercicios Tema 3

Programación Paralela

José María Cecilia

Grado en Informática

Compilación y ejecución de programas

Para compilar los programas, vamos a utilizar el compilador de C/C++ de gnu (g++). Para compilar un programa de nombre prog.c, que incluya directivas y funciones de OpenMP, basta con ejecutar:
> g++ –fopenmp [-Ox] –o prog prog.c .

Si estamos interesados en el tiempo de ejecución, compilaremos con la optimización – O2 tanto en serie como en paralelo (está puesta por defecto y utiliza las optimizaciones más habituales: registros, loop unrolling, rutinas in-line…). Las otras opciones principales son –O0 (genera un código muy poco eficiente) y –O3 (realiza una optimización más agresiva que puede no dar mejores resultados, se puede probar para cada programa y ver si merece la pena). Si no hay errores, se habrá creado el correspondiente ejecutable. Recuerda que el número de hilos a utilizar se puede controlar, entre otras maneras, mediante una variable de entorno:

export OMP_NUM_THREADS=xx xx = número de threads

Toma de tiempos

Una posibilidad es utilizar la función gettimeofday de la librería sys/time.h. Por tanto es dependiente del sistema y se debe incluir la cabecera #include <sys/time.h>

```
struct timeval start, end;
gettimeofday(&start, NULL);

// benchmark code

gettimeofday(&end, NULL);

delta = ((end.tv_sec  - start.tv_sec) * 1000000u +
         end.tv_usec - start.tv_usec) / 1.e6;
```

Otra opción es utilizar la función de omp omp_get_wtime() Esta Liberia es portable entre SSOO

```
// omp_get_wtime.cpp
// compile with: /openmp
#include "omp.h"
#include <stdio.h>
#include <windows.h>

int main() {
    double start = omp_get_wtime( );
    Sleep(1000);
    double end = omp_get_wtime( );
    double wtick = omp_get_wtick( );

    printf_s("start = %.16g\nend = %.16g\ndiff = %.16g\n",
             start, end, end - start);

    printf_s("wtick = %.16g\n1/wtick = %.16g\n",
             wtick, 1.0 / wtick);
}
```


José M. Cecilia - Tlf: (+34) 968 27 85 87 - jmcecilia@ucam.edu

Ejercicios propuestos

Ejercicio 1. Realice un programa en openmp que pida al usuario el número de hilos a ejecutar y muestre por pantalla el número de hilos que se están ejecutando en paralelo, el identificador de hilo al ejecutarse en paralelo. Además en la parte paralela, se debe imprimir el número de hilos que hay en ejecución una sola vez.

Ejercicio 2. Realice un programa omp que muestre la diferencia entre firstprivate y lastprivate. ¿Cuál es esta diferencia?

Ejercicio 3. Realice un programa en omp que realice la suma de vectores de tamaño N (1024, 2048, 4096) utilizando 2, 4, 6 y 8 procesos. Tome tiempos de ejecución y muestre un gráfico que indique la escalabilidad del sistema.

Ejercicio 4. Realice un programa en omp que realice el código saxpy; y[i] = x[i]*alpha + y[i] para arrays de tamaño N (1024, 2048, 4096) utilizando 2, 4, 6 y 8 procesos. Tome tiempos de ejecución y muestre un gráfico que indique la escalabilidad del sistema.

Ejercicio 5. Realice un código que reduzca un vector de tamaño N (1024, 2048, 4096) utilizando 2, 4, 6 y 8 procesos. Tome tiempos de ejecución y muestre un gráfico que indique la escalabilidad del sistema.

Ejercicio 6. Realice un código paralelo en omp que contenga una sección paralela y dentro de esta sección imprima el id del master thread. El resto de hilos deben imprimir su idea después del maestro deben realizar ninguna acción.

Ejercicio 7. Dado el código secuencial de la Figura 1 para el cálculo del número pi, realice una implementación en omp que paralelice el código. Ejecute distintos configuración de hilos y muestre una tabla de tiempos. ¿Con cuántos hilos es más eficiente el programa? ¿Por qué?

```
static long num_steps = 100000;
double step;
void main ()
{     int i; double x, pi, sum = 0.0;
      step = 1.0/(double) num_steps;
      for (i=1;i<= num_steps; i++){
            x = (i-0.5)*step;
            sum = sum + 4.0/(1.0+x*x);
      }
      pi = step * sum;
}
```

Figura 1 Código secuencial para el cálculo del número pi

José M. Cecilia - Tlf: (+34) 968 27 85 87 - jmcecilia@ucam.edu