Universidad Nacional de San Agustin
Esc. Profesional de Ciencia de la Computación

Algoritmos Paralelos

Tema: Laboratorio

Profesor: Alvaro Henry Mamani Aliaga
Alumno: Diego André Ranilla Gallegos

5 de Junio del 2017

1. Macro _OPENMP

Escribir un programa que imprime su valor, explicar la significancia de ese valor.

```
#include <omp.h>
#include <stdio.h>

int main(){

  printf("_OPENMP = %d\n",_OPENMP);
}
```

Listing 1: Multiplicación matriz-vector

Salida: $\_OPENMP = 201107$.
La salida indica la fecha de lanzamiento de la versión del *openMP* instalado en la PC.

Para efectuar la multiplicación, la matriz A, los vectores x e y, m y n deben ser variables globales en otras palabras estas variables van hacer usadas por los distintos threads, por otro lado para que se efectúe la multiplicación vamos a repartir entre los threads las filas de la matriz A para que se encarguen de calcular la multiplicación y almacenar la respectiva respuesta en los distintos elementos de la matriz y.

2. Descargar omp_trap_1.c

Borrar de este código la directiva *critical*. Ejecutar con con más y más hilos y con valores más grandes para n. ¿Con cuantos hilos y con cuantos trapezoides toma antes que el resultado es **incorrecto**?

Si aplicamos los siguientes datos:

- **Prueba 1**
  hilos = 30
  a = 0.0
  b = 3.0
  n = 30000000

  With n = 30000000 trapezoids, our estimate of the integral from 0.000000 to 3.000000 = 9.00000000000002e+00

- **Prueba 2**
  hilos = 15
  a = 0.0
  b = 3.0
  n = 30000000

  With n = 30000000 trapezoids, our estimate of the integral from 0.000000 to 3.000000 = 8.99999999999998e+00

- **Prueba 3**
  hilos = 300

a = 0.0
b = 3.0
n = 30000000
With n = 30000000 trapezoids, our estimate of the integral from 0.000000 to 3.000000 = 8.85657200000001e+00

En es estas muestras podemos deducir que el resultado obtenido depende de la cantidad de hilos que empleemos, cuando los hilos son pocos, por el déficit de ellos el cálculo será impreciso, luego cuando hay gran cantidad de hilos el resultado también no será el correcto debido a que hemos retirado la directiva *critical* por lo que no hay un orden al emplear la variable *global_result* el cual es una variable compartida en el cual se suma los resultados de cada uno de los hilos.

3. Modificar omp_trap_1.c

```
1  double start = omp_get_wtime( );
2  #  pragma omp parallel num_threads(thread_count)
3     {
4  #     pragma omp barrier
5        {
6  #        pragma omp critical
7           global_result+=  Trap2(a, b, n);
8        }
9     }
10 double end = omp_get_wtime( );
```

Listing 2: Modificaciones en omp_trap_1.c

```
1  double start = omp_get_wtime( );
2
3  #  pragma omp parallel num_threads(thread_count)
4     {
5        double my_result = 0.0;
6        my_result += Local_trap(a, b, n);
7  #     pragma omp critical
8        global_result += my_result;
9     }
10
11 double end = omp_get_wtime( );
```

Listing 3: Modificaciones en omp_trap_2a.c

	Hilos x N		
	80 x 8000	8000 x 8000	8 x 8,000,000
Cod 1	0.003351531999214785	0.2011130300015793	0.1790352520001761
Cod 2	0.002445156998874154	0.1792486329977692	0.04875552699741093

Tabla 1: Tabla de resultados, Cod 1 (omp_trap_1.c) y Cod 2 (omp_trap_2a.c)

A partir de los resultados podemos deducir que *Cod*2 es más veloz debido a que a pesar que son similares, difieren en que Cod 1 posee la directiva *barrier*, por lo que con esta directiva, sincronizan los hilos antes de acceder a la sección crítica.