Tecnológico de Monterrey

Act 1.4

Periodo Semestral Agosto-Diciembre 2022
Fecha de entrega |*Agosto 25, 2022*

Multiprocesadores TE3061.1

Profesor:

Emmanuel Torres Rios

Abril Jimenez Alatriste | *A01732412*

Instituto Tecnológico de Estudios Superiores de Monterrey

Instrucción

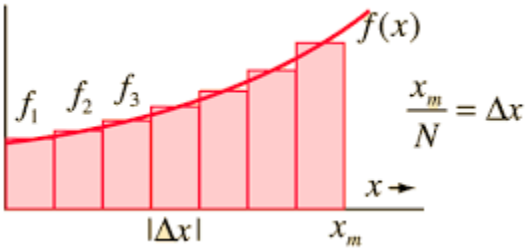
Usando el programa del área bajo la curva, determine el número de threads óptimo para el cálculo de la solución con el mayor número de puntos que permita la arquitectura de su procesador. Debe mostrar una gráfica que permita observar como determinó su resultado y la selección de número de threads en cada prueba. Este es un trabajo individual y deberá entregar un reporte como evidencia de la actividad.

Introducción

El área bajo la curva esta formada por el trazo de la función f(x) y el eje x que se puede obtener aproximadamente, dibujando rectángulos de anchura fina y altura f igual al valor de la función en el centro del intervalo, esa es la definicion de el area bajo la curva la cual tomamos en cuenta en esta ocasión para asi obtenerla solucion con el mayor numero de puntos de la arquitectura de el procesador intel core i7 que es el que el equipo OMEN 015 cuenta.

Marco Teórico

Para esta actividad se nos brindó un programa el cual, se basaba en la formula del área bajo la curva, la cual consiste en trazar rectángulos debajo de una curva como se muestra en la siguiente imagen.



Una vez entendido el concepto pasamos a el programa el cual se proporcionó en la clase y es adjuntado en la siguiente captura de pantalla.

```c
#include <stdio.h>
#include <omp.h>
static long numpasos = 10000;
double paso;
#define NUM_THREADS 20
void main()
{
    int i, nthreads;
    double pi, sum[NUM_THREADS], t1, t2, tiempo;
    paso = 1.0/numpasos;
    omp_set_num_threads(NUM_THREADS);
    const double startTime = omp_get_wtime();
    t1 = omp_get_wtime();
#pragma omp parallel
{
    int i, id, nthrds;
    double x;
    id = omp_get_thread_num();
    nthrds = omp_get_num_threads();
    if (id == 0) nthreads = nthrds;

    for (i = id, sum[id] = 0.0; i < numpasos; i = i + nthrds)
    {
        x = (i + 0.5) * paso;
        sum[id] += 4.0/(1.0 + (x*x));
    }
}
for (i = 0, pi = 0.0; i < nthreads; i++)
    pi += sum[i] * paso;
const double endTime = omp_get_wtime();
tiempo = t2-t1;
printf("Pi = %lf\n", pi);
printf("Tiempo total = %lf\n", (endTime - startTime));
}
```

Una vez el programa entendido pasamos a las pruebas en el procesador, primero utilizando las siguientes cantidades y resultados.

```c
static long numpasos = 100000;
double paso;
#define NUM_THREADS 5
```

```
:\Users\Yukin\Desktop\multi>gcc -fopenmp pi_parallel.c

:\Users\Yukin\Desktop\multi>.\a.exe
i = 3.141593
iempo total = 0.000000
```

```
static long numpasos = 100000;
double paso;
#define NUM_THREADS 10
```

```
C:\Users\Yukin\Desktop\multi>gcc -fopenmp pi_parallel.c

C:\Users\Yukin\Desktop\multi>.\a.exe
Pi = 3.141593
Tiempo total = 0.000000
```

```
#include <stdio.h>
#include <omp.h>
static long numpasos = 100000;
double paso;
#define NUM_THREADS 20
```

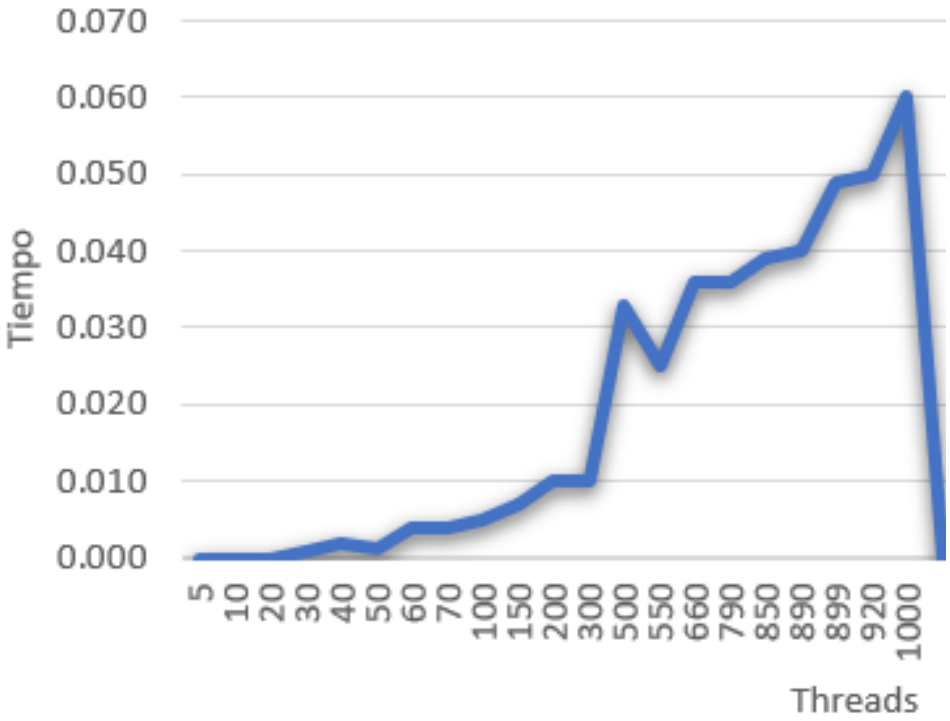
```
C:\Users\Yukin\Desktop\multi>gcc -fopenmp pi_parallel.c

C:\Users\Yukin\Desktop\multi>.\a.exe
Pi = 3.141593
Tiempo total = 0.000000
```

```
static long numpasos = 100000;
double paso;
#define NUM_THREADS 30
```

```
C:\Users\Yukin\Desktop\multi>.\a.exe
Pi = 3.141593
Tiempo total = 0.010000
```

Siguiendo de esa manera hasta obtener los siguientes resultados mostrados en la grafica.



Conclusión

Se concluye que apesar de sus limitaciones el procesador icore 7 es un muy buen procesador ya que aun que no se añadir en la grafica se probo con 999 y aun funciono el programa.