Tarea Laboratorio 4

Alegria Sallo Daniel Rodrigo (215270)

20 de mayo de 2024

Escribir un programa parallelo con OpemMP para calcular la integral por el metodo de los rectangulos y calcule el seepdup y la eficiencia.

$$\int_a^b \frac{x^3}{3} + 4x dx$$

Codigo Fuente

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

double f( double x ) {
    return pow(x,3) / 3 + 4*x;
}

int main(int argc, char *argv[])
{
    int ntest = 10; // number of tests
    double h = 0.1; // the width of each rectangle
    double a = 0;   // lower bound of the interval
    double b = 100; // upper bound of the interval

    printf("Nro Hilos, Tiempo Secuencial (ms), Tiempo Paralelo (ms), ");
    printf("Speedup, Eficiencia");
    printf("\n");

    double start, end;
    double seq_time, par_time;

    int num_threads = omp_get_max_threads();

    for ( int t = 0; t < num_threads; ++t ) {
        seq_time = 0;
        for ( int n = 0; n < ntest; ++n ) {
            double sum = 0; // la integral de f
            start = omp_get_wtime();
            // CODIGO SECUENCIAL
            for ( double x = a; x <= b; x += h ) {
                sum += x * f(x);
            }
            // END
            end = omp_get_wtime();
            seq_time += end - start;
        }
        // promedio de tiempo secuencial
        seq_time = seq_time / ntest;
```

```
        par_time = 0;
        for ( int n = 0; n < ntest; ++n ) {
            start = omp_get_wtime();
            // CODIGO EN PARALLELO
            double sum = 0; // la integral de f
            double mul = 1/h;
            double x;
            #pragma omp parallel for num_threads(t)
            // Como no podemos usar el tipo de datos 'double' en la
↪   estructura
            // 'for'. Nos ayudamos de un pequeño truco matematico ...
            for (int i = (int)(a*mul); i <= (int)(b*mul); i+=(int)(h*mul))
↪   {
                x = i*h;
                sum += x * f(x);
            }
            // END
            end = omp_get_wtime();
            par_time += end - start;
        }
        // promedio de tiempo parallelo
        par_time = par_time / ntest;

        // Results
        double speedup = seq_time / par_time;
        double efficiency = speedup / t;
        printf("%d, %lf, %lf, %lf, %lf\n",
                t+1, seq_time, par_time, speedup, efficiency
        );
    }

    return 0;
}
```

Resultados

Nro Hilos	Tiempo Secuencial (ms)	Tiempo Paralelo (ms)	Speedup	Eficiencia
1	0.000022	0.000103	0.211138	inf
2	0.000021	0.000022	0.966927	0.966927
3	0.000021	0.000049	0.424665	0.212333
4	0.000021	0.000060	0.347368	0.115789
5	0.000021	0.000062	0.333180	0.083295
6	0.000021	0.000046	0.462385	0.092477
7	0.000021	0.000058	0.362260	0.060377
8	0.000021	0.000070	0.295075	0.042154

Figura 1: Variacion del Speedup

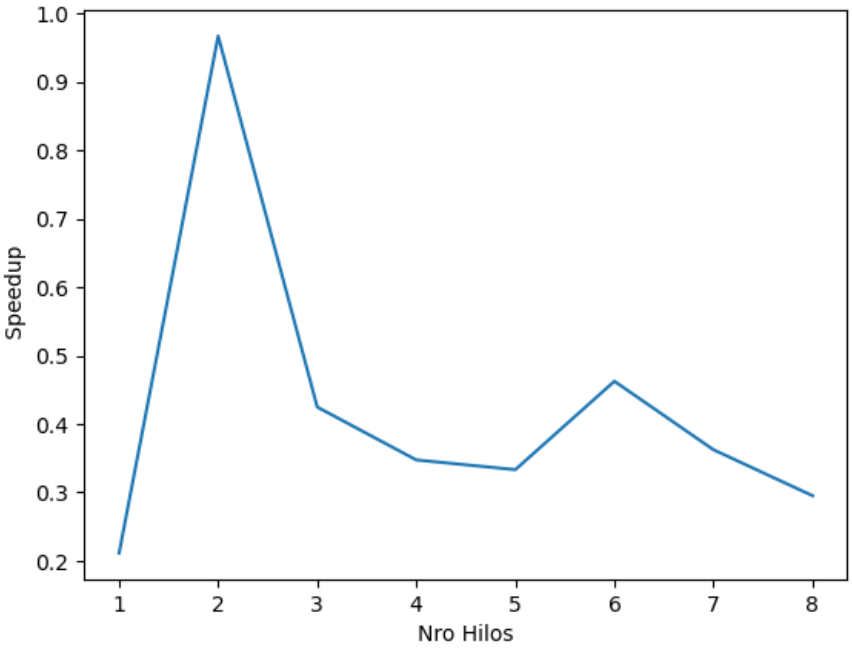

Figura 2: Variacion de la Eficiencia