# Tarea Laboratorio 4

Alegria Sallo Daniel Rodrigo (215270)

21 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 <string.h>
#include <omp.h>

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

int main(int argc, char *argv[])
{
    double a = 0;      // lower bound of the interval
    double b = 1000;   // upper bound of the interval
    double h = 0.01;   // the width of each rectangle
    int ntest = 10;    // number of tests

    if ( argc == 2 && strcmp("-i", argv[1]) == 0 ) {
        printf("a: "); scanf(" %lf", &a);
        printf("b: "); scanf(" %lf", &b);
        printf("h: "); scanf(" %lf", &h);
        printf("ntest: "); scanf(" %d", &ntest);
    }

    if ( argc > 1 ) {
        a = atof(argv[1]);
        b = atof(argv[2]);
        h = atof(argv[3]);
        ntest = atoi(argv[4]);
    }

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

    double start, end;
    double seq_time, par_time;

    int num_threads = omp_get_max_threads();

    for ( int t = 1; 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;
        }
        seq_time = seq_time / ntest;  // promedio de tiempo secuencial

        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;
        }
        par_time = par_time / ntest;  // promedio de tiempo parallelo

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

    return 0;
}
```

## Resultados

| Nro Hilos | Tiempo Secuencial (ms) | Tiempo Paralelo (ms) | Speedup | Eficiencia |
|---|---|---|---|---|
| 1 | 0.005880 | 0.006601 | 0.890734 | 0.890734 |
| 2 | 0.005870 | 0.014454 | 0.406143 | 0.203072 |
| 3 | 0.005868 | 0.018468 | 0.317729 | 0.105910 |
| 4 | 0.005871 | 0.015923 | 0.368694 | 0.092174 |
| 5 | 0.005869 | 0.013569 | 0.432499 | 0.086500 |
| 6 | 0.005867 | 0.011797 | 0.497315 | 0.082886 |
| 7 | 0.005864 | 0.012263 | 0.478182 | 0.068312 |
| 8 | 0.005872 | 0.012477 | 0.470597 | 0.058825 |

Figura 1: Variacion del Speedup

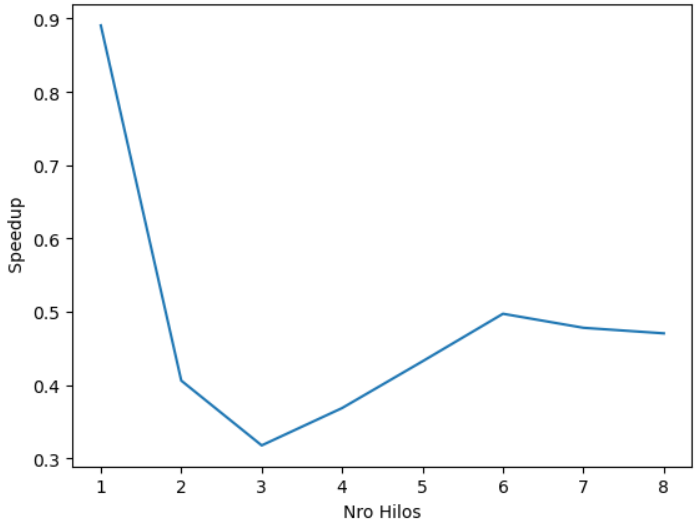


Figura 2: Variacion de la Eficiencia