CI-0117 Laboratorio 12 Paralela y Concurrente
Rodrigo Vilchez Ulloa | B78292

---

1. Implementación `MPI`

```
rigovil@rodrigo:~/Escritorio/U
 mpiexec -n 2 ./a.out
4 = 3 + 1
6 = 3 + 3
8 = 3 + 5
10 = 3 + 7
Tiempo: 0s, 341106199ns
```

Se prueba el programa con la medición de tiempo implementada en la clase `Chrono` y se utiliza un n pequeño para comprobar la correcta funcionalidad del programa utilizando `MPI`. La lógica del programa es que, existe un hilo principal: el hilo 0, el cual calcula los extremos de los intervalos que será utilizado por los demás hilos para saber con cuáles números trabajar. Estos extremos son enviados, uno por mensaje, a cada hilo, y cuando estos terminan de calcular su parte, le envían un mensaje al hilo principal de que ya concluyeron. El hilo principal también calcula su parte y espera a que el resto termine, para indicar el tiempo final tomado por el programa para hacer los cálculos.

2. Pruebas

Se muestran distinas pruebas para esta implementación. Se escogió un $n = 100000$ como límite superior y una cantidad de 2^k trabajadores. Se muestran algunas pruebas para llenar la tabla, en las otras solamente se indicará el tiempo de duración. El programa no imprime la solución para cada número par para medir el tiempo de duración real del cálculo. Pero dentro del programa se puede descomentar la línea 25 para imprimir los resultados. Para cambiar el valor de n, se debe hacer directamente desde la línea 35. Compilar utilizando el comando `make` y correr con el comando `mpiexec -n T ./a.out` donde `T` es la cantidad de trabajadores.

```
rigovil@rodrigo:~/Escritorio/UCR/I
$ mpiexec -n 2 ./a.out
Tiempo: 15s, 427254730ns
rigovil@rodrigo:~/Escritorio/UCR/I
```

Implementación `MPI` y 2 trabajadores

```
rigovil@rodrigo:~/Escritorio/U
$ mpiexec -n 4 ./a.out
Tiempo: 10s, 504125890ns
rigovil@rodrigo:~/Escritorio/U
```

Implementación `MPI` y 4 trabajadores

```
rigovil@rodrigo:~/Escritorio/UCR/
$ mpiexec -n 8 ./a.out
Tiempo: 8s, 518789994ns
rigovil@rodrigo:~/Escritorio/UCR/
```

Implementación `MPI` y 8 trabajadores

5. Tabla comparativa y conclusiones

Serial	`Pthreads`	`OpenMP`	`MPI`	**Trabajadores**
18s, 514935138ns	14s, 542521643ns	9s, 758637697ns	15s, 427254730ns	2
18s, 514935138ns	8s, 526936582ns	5s, 956124787ns	10s, 504125890ns	4
18s, 514935138ns	5s, 898312308ns	5s, 727516976ns	8s, 518789994ns	8
18s, 514935138ns	5s, 778442800ns	5s, 680402289ns	8s, 678757186ns	16
18s, 514935138ns	5s, 764609126ns	5s, 808271955ns	8s, 276221335ns	32
18s, 514935138ns	5s, 552529213ns	5s, 746974931ns	9s, 616242149ns	64

Se evidencia como una implementación en paralelo es considerablemente mejor que la implementación serial. La comparativa indica que `OpenMP` sigue siendo una mejor solución en cuanto a rendimiento. La solución utilizando `MPI` indica que es más malo que el resto de implementaciones en paralelo, pero aún así es mejor que la implementación serial. Utilizando 8 trabajadores se muestra el mejor desempeño, por lo que cuando se aumenta la cantidad, el rendimiento empeora, aunque en cantidades muy pequeñas.