Proyecto Final de Programación y Análisis de algoritmos

Catalina Díaz Hernández

13 de diciembre de 2021

1. Comparación de desempeño de algoritmos secuenciales y en paralelo en operaciones matriciales

1.0.1. Objetivos

- Programación de diversos algoritmos en secuencial.
- Paralelización utilizando pthreads.
- Comparación de tiempos de ejecución.

1.0.2. Estructura

Se creó la extructura Matrix.h ya que facilita el paso de parámetros a las funciones ejecutadas por los threads (hilos). Las matrices se inicializan con entradas entre 0 y 10.

```
struct Matrix{
long int nrow;
long int ncol;
int t_id; //identificador del thread que usa a la matriz
double *data;
Matrix(): nrow(0), ncol(0), t_id(-1), data(nullptr){}
};
typedef struct Matrix Matrix;
```

1.0.3. Algoritmos secuenciales

Se programaron algoritmos clásicos para realizar las operaciones entre matrices. Éstas se manejan como vectores unidimensionales, la entrada (i, j) se encuentra en la posición $i * ncol + j$.

- Suma.

```
        for (i = 0; i < suma->nrow; i++) {
for (j = 0; j < suma->ncol; j++) {
suma[i*suma->ncol+j] = data1[i*data1->ncol+j] + data2[i*data2->ncol+j];
}
}
```

- Multiplicación.

```
 for (i = 0; i < mult->nrow; i++) {
for (j = 0; j < mult->ncol; j++) {
for(int k = 0; k < ncol1; k++){
data_mult[i*mult->ncol+j] += data1[i*m1->ncol+k] * data2[k*m2->ncol+j];
}
}
}
```

- Transpuesta.

```
for i = 1 M,j = 1.... N do
A_j,i ← A_i,j
end
```

1.0.4. Algoritmos paralelos

Para distribuir la carga de trabajo entre los threads, se realizaron diferentes métodos que dependen de la operación que se está realizando. De igual manera se probaron repeticiones utilizando 2,8,32 y 128 threads.

- Paralelización Suma.

  para el thread id (tid) en numero de threads hacer:

  1.Se obteniene la posición del bloque asignado pos_{tid} a tid.

  2.Lanzar el hilo con las franjas $A[pos_{tid}]B[pos_{tid}]$

  3.Se realiza la suma $A[pos_tid]yB[pos_{tid}]$ de forma secuencial.

- Paralelización Multiplicación.

  para el thread id (tid) en numero de threads hacer:

  1. Se obtiene la posición del bloque asignado pos_{tid} a tid.

  2.Lanzar el hilo la franja $A[pos_{tid}]yB(completa)$

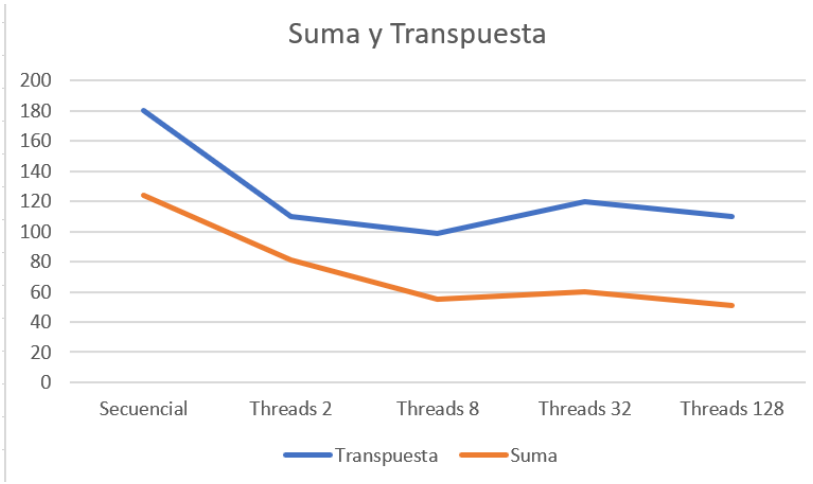
  3.Se reliza la multiplicación $A[pos_{tid}]yBdeformasecuencial$

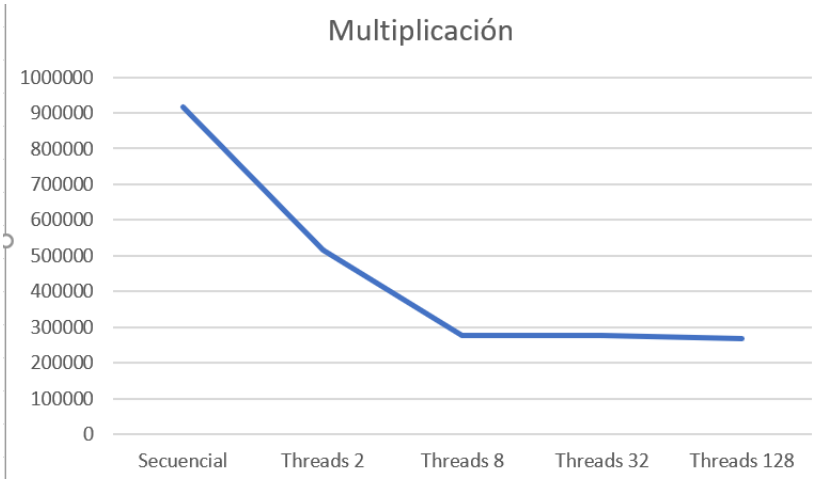
1.0.5. Rendimiento (Evaluación)

Se realizaron pruebas con matrices aleatorias de 5000x5000 con diferentes números de threads, a continuación se mostrarán los resultados.

	Secuencial	Threads 2	Threads 8	Threads 32	Threads 128
Transpuesta	180	110	99	120	110
Suma	124	81	55	60	51
Multiplicación	917065	517230	277648	275724	268037

A continuación se presentan las gráficas comparativas de tiempos para las distintas operaciones y distintos números de threads.





A partir de las gráficas, se puede concluir que el número de threads óptimo es 8, ya que presenta los mejores tiempos. Se puede concluir que la paralelización disminuye los tiempos de cómputo, sin embargo no siempre un mayor número de threads lleva a un menor tiempo.