UNIVERSIDAD
DE SANTIAGO
DE CHILE

Departamento de Matemática y Ciencia de la Computación

Pep 1

Algoritmos Distribuidos

Segundo Semestre 2016

Algoritmos Distribuidos 26106
Licenciatura en Ciencia de la Computación

Fernando Garcia Polgati
fernando.gp101@gmail.com

1 Plataforma Computacional

Los programa fueron ejecutados en un computador con las siguientes características:

- **Procesador:** Intel(R) Core(TM) i5-7300HQ CPU 2.50GHz
- **Memoria RAM:** 7994312 kB
- **Sistema Operativo:** Linux - Ubuntu 18.04

2 Pregunta 1

2.1 Descripción del problema

Para el problema 1 se busca estudiar el comportamiento que determina la existencia de estrellas a partir de una matriz por filas, construya una implementación que la recorra por columnas y haga una comparación de las versiones secuencial y paralelas.
Para esto se considerara el numero de hebras entre 2 a 6.

2.2 Procedimiento

Para el caso del algoritmo secuencial se tomará solamente el tiempo de la función en que se procesa las funciones principales del procedimiento, sin contar la declaración de variables ni la impresión de estas.
Para el algoritmo en paralelo se toma el tiempo que demora partiendo de su función principal hasta que las hebras entren a la función pthread_join

2.3 Estructuras de Datos Utilizadas

La estructura de datos utilizada en este algoritmo es:

```
    struct Message {
    int myid, rvalue, cvalue, size, opmode;
}
```

2.4 Análisis de Complejidad

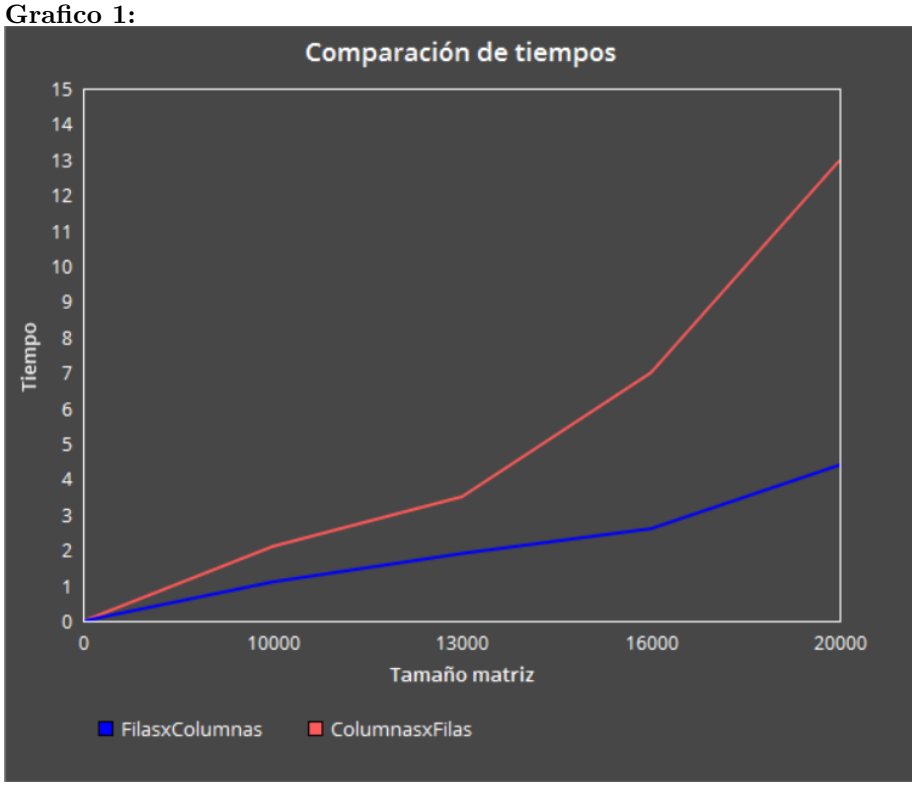
El algoritmo propuesto tiene costo O(n^2).

2.5 Implementación

El algoritmo está implementado en lenguaje C

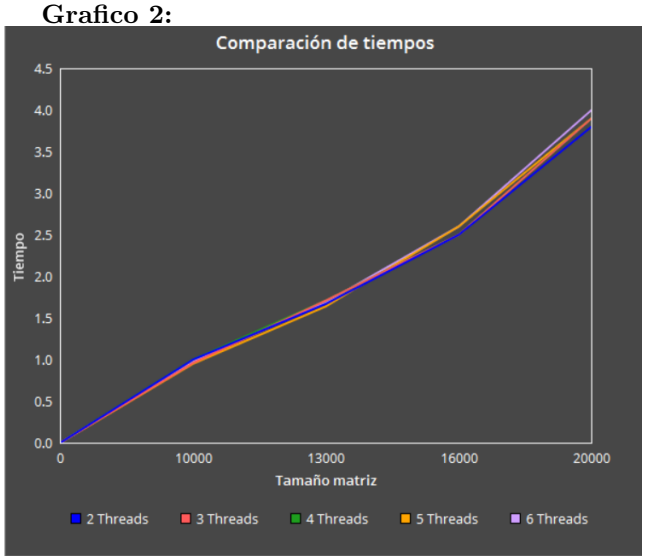
2.6 Resultados Experimentales

Grafico 1:



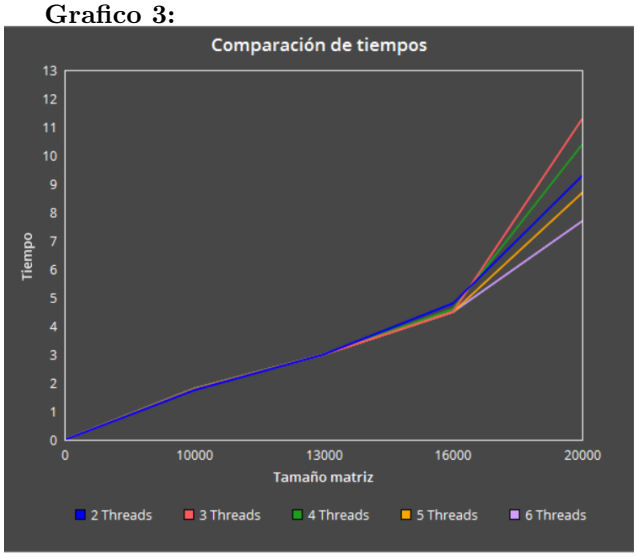
Se puede apreciar claramente en el grafico que el algoritmo que ejecuta las matrices columna x fila tiene mucho más tiempo de ejecución para resolver el problema que el algoritmo que ejecuta las matrices fila x columna.

Grafico 2:



Teniendo el algoritmo de estrellas por paralelo usando filas x columna tenemos resultados muy similares al usar distintas hebras para resolver el problema

Grafico 3:



Para el caso del algoritmo de estrellas por paralelo usando columna x filas podemos ver resultados similares hasta llegar al tamaño de matriz de 16.000, es en este momento donde el tiempo de ejecución de las distintas hebras empieza a variar.

De las tres graficas se observa claramente que la ejecución de Columna x fila es mucho más lenta.

2.7 Conclusiones

Al tener un algoritmo hay varias formas que se puede manipular para verificar si la ejecución mejora o empeora. El uso de las hebras nos ayuda para optimizar el uso del cpu y para lograr algunas mejoras dependiendo del caso en las cuales se usan.

3 Pregunta 2

3.1 Introducción

Para esta pregunta se busca analizar el desempeño que calcula las raices cuadradas de numeros entre 1 y n

3.2 Procedimiento

Para el algoritmo en paralelo se toma el tiempo que demora partiendo de su función principal hasta que las hebras entren a la función pthread$_join$

3.3 Estructuras de Datos Utilizadas

```
struct Message {
    int myid, nvalue, numthreads; float *data;
};
```

3.4 Análisis de Complejidad

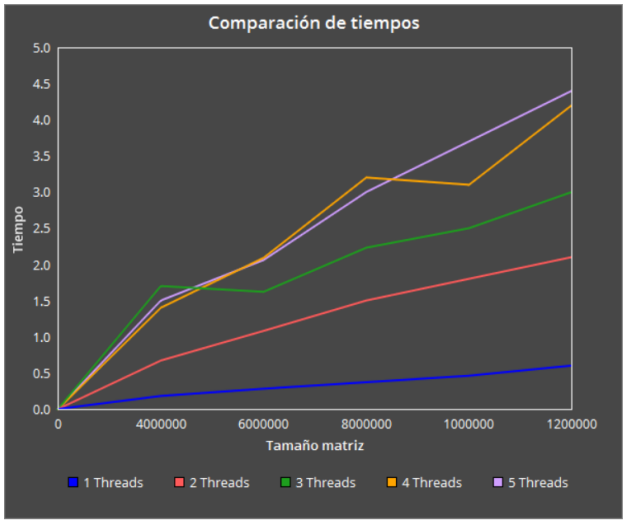
El algoritmo propuesto tiene costo O(n)

3.5 Implementación

El algoritmo está implementado en lenguaje C

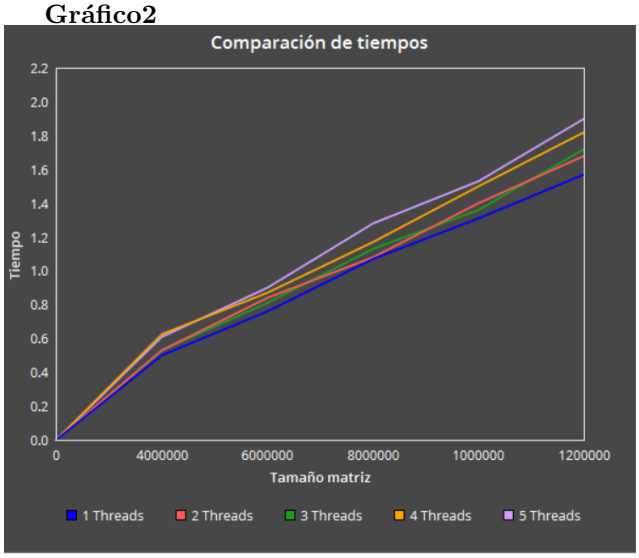
3.6 Resultados Experimentales

Gráfico 1



En el grafico del algoritmo de raices cuadradas sin variable global podemos ver como va aumentando el tiempo de ejecución al momento de usar más hebras

Gráfico2



En el grafico del algoritmo de raices cuadradas con variable global aumenta un poco el tiempo de ejecución cuando se utilizan más hebras

Se puede apreciar que el algoritmo que se ejecuta usando una variable global es más rapido en procesar la información.

3.7 Conclusiones

Podemos ver que para este caso las variables globales van a ser mejores para el momento de ejecutar el algoritmo y que el aumento de threads nos va a hacer aumentar el tiempo de ejecución total.

4 Pregunta 3

4.1 Introducción

Se busca construir un programa paralelo que reciba como entrada un conjunto de n elementos almacenados en una lista y genera su conjunto potencia.

4.2 Estructuras de Datos Utilizadas

```
Struct Nodo{
    unsigned int info;
    struct node *next
}
```

4.3 Implementación

El algoritmo está implementado en lenguaje C.