Analisis de Tiempos en algoritmos usando pthreads en C

Alfred A. Chavez [1]

[1]Ciencia de la computación – Universidad Católica San Pablo

alfred.chavez@ucsp.edu.pe

1. Introducción

En este documento se presentarán los resultados de las pruebas hechas en algoritmos seriales, técnicas y algoritmos en paralelo, usando pthreads

2. Pruebas

Las siguientes pruebas son de códigos hechos en C++, en una laptop con procesador Intel Core i5-7200U, 2.50GHz x 4(cores), RAM 11.6GB, con sistema operativo Linux 4.15.3, distro Fedora 27; comopilador usado es GNU GCC 7.3.1 y para la medida de tiempo se usa la herramienta brindada por OpenMP:

```
1 |  gcc -o a.out -fopenmp -lpthread -g
```

2.1. Resultados

2.1.1. Tiempo

	Matrix Dimension					
	8000000 x 8		8000 x 8000		8 x 8000000	
Threads	Time (s)	Eff	Time (s)	Eff	Time (s)	Eff
1	0,23643825	1,04666631	0,20490020	1,00154197	0,20433320	1,17123209
2	0,13299575	0,93037541	0,11091745	0,92508505	0,11413975	1,04837097
4	0,11014665	0,56168742	0,10532150	0,56807395	0,16700475	0,35825568

Figura 1. Tablas de tiempos matriz vector

Threads	Busy-Waiting	Mutex
1	0,00011180	0,00011095
2	0,00011910	0,00014050
4	0,00011940	0,00022445
8	0,0004686	0,0003161
16	0,0008137	0,0006320
32	0,007436950	0,0009784
64	0,0306556	0,0012693

Figura 2. Tablas de tiempos Busy-Waiting vs Mutex

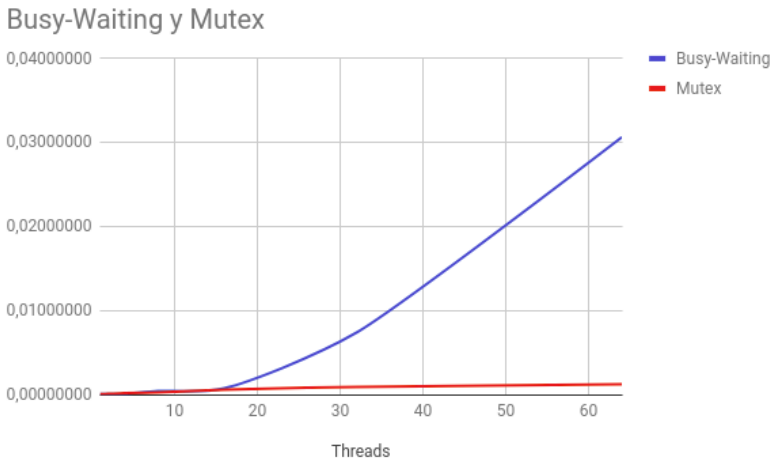

Figura 3. Comparacion de tiempos Busy-Waiting, Mutex

Implementation	Number of Threads			
	1	2	4	8
Read-Write Locks	0,17334495	0,09305200	0,07352520	0,07336750
One Mutex per entire list	0,17433890	0,29662850	0,30199070	0,30446780
One mutex per node	0,09283965	0,07855125	0,06345575	0,07256170

Figura 4. Tabla de Comparacion de tiempos pthreads en listas enlazadas

2.2. Análisis

Podemos ver en la multiplicación Matriz-Vector que los tiempos van disminuyendo a medida que aumenta el paralelismo y que este tiempo también va a depender del tamaño de las columnas y filas, aunque tengan el mismo número de elementos las matrices de prueba, el factor de columnas y filas es importante en la toma de tiempos, luego en Busy-Waiting vs Mutex podemos ver como con el Busy-Waiting el algoritmos a medida que aumenta el número de threads va a ir aumentando su tiempo y crece de gran manera, en cambio con el uso de Mutex, estos tiempos se mantienen en un rango, y no suben tan grandemente como podemos ver en el gráfico de esta comparación. y finalmente podemos notar en el caso de las listas enlazadas que si usamos solo 1 mutex por toda la lista el tiempo crece demasiado y cuando usamos un mutex por nodo o candado read-write este tiempo va disminuyendo conforme aumenta el numero de threads.

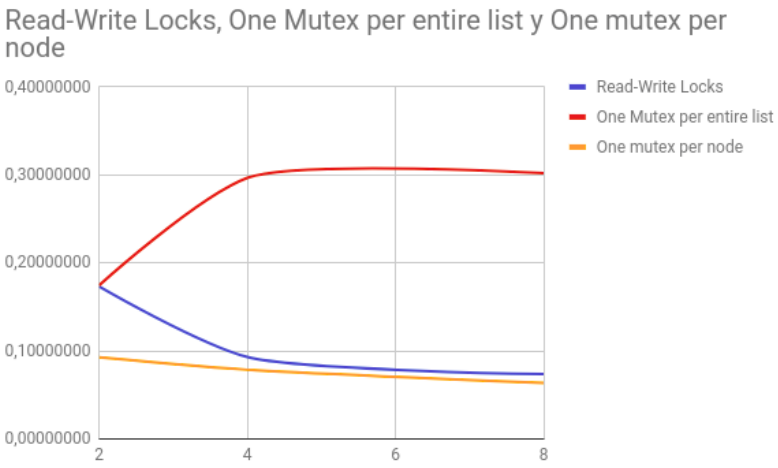

Figura 5. Comparacion de tiempos pthreads tn listas enlazadas