# Capítulo 4: Pthreads

Angela Keshia Talavera Ormeño

April 2018

## 1 Multiplicación matriz - vector

| Threads | Matrix Dimension | | | | | |
|---|---|---|---|---|---|---|
| | $8,000,000 \times 8$ | | $8000 \times 8000$ | | $8 \times 8,000,000$ | |
| | Time | Eff. | Time | Eff. | Time | Eff. |
| 1 | 0.393 | 1.000 | 0.345 | 1.000 | 0.441 | 1.000 |
| 2 | 0.217 | 0.906 | 0.188 | 0.918 | 0.300 | 0.735 |
| 4 | 0.139 | 0.707 | 0.115 | 0.750 | 0.388 | 0.290 |

Figure 1: Matriz-Vector

De acuerdo a los resultados podemos concluir que el tiempo de ejecución aumenta cada vez que se incrementa el número de hilos utilizados. La eficiencia máxima se logra al utilizar 2 hilos en una matriz de 8000x8000.

| Threads | Matriz Dimensión | | | | | |
|---|---|---|---|---|---|---|
| | 8000000 * 8 | | 8000 * 8000 | | 8 * 8000000 | |
| | Tiempo | Eficiencia | Tiempo | Eficiencia | Tiempo | Eficiencia |
| 1 | 0.363188 | 1 | 0.348104 | 1 | 0.344442 | 1 |
| 2 | 0.257912 | 0.249136 | 0.143124 | 0.490074 | 0.22803 | 0.488095 |
| 4 | 0.252813 | 0.054647 | 0.136933 | 0.234542 | 0.182569 | 0.226037 |

Figure 2: Matriz-Vector

## 2 Lista enlazada, tabla 4.3

En esta prueba se realizaron 99900 operaciones Member, 50 operaciones Insert y 50 operaciones Delete. Se puede observar que el mejor desempeño lo obtiene

Read Write Locks excepto en la ejecución secuencial donde el mejor desempeño lo obtiene One mutex for the entire list.

| | 1 | 2 | 4 | 8 |
|---|---|---|---|---|
| Read-Write Locks | 0,01211906667 | 0,0763754 | 0,413084 | 0,9542759 |
| One mutex for the entire list | 0,0114671 | 0,1200681 | 0,7968294 | 2,2586454 |
| One mutex per node | 0,0462275 | 0,3488624 | | |

Figure 3: Matriz-Vector

# 3 Lista enlazada, tabla 4.4

En esta prueba se realizaron 100000 operaciones 80 operaciones Member, 10 operaciones Insert y 10 operaciones Delete, el mejor desempeño se da con One mutex for the entire list seguido de Read-Write Locks, la diferencia no es amplia pues actúan de manera similar, sin embargo el mutex por nodo es lento ya que hace múltiples bloqueos en cada inserción en el nodo que se desea insertar, buscar o eliminar el elemento.

| | 1 | 2 | 4 | 8 |
|---|---|---|---|---|
| Read-Write Locks | 2,548376267 | 9,0727157 | 20,5475632 | 42,9285364 |
| One mutex for the entire list | 2,2214123 | 7,5118578 | 15,8351603 | 22,2687824 |
| One mutex per node | 4,3756151 | 19,7101552 | 53,8014024 | 113,6549608 |

Figure 4: Lista enlazada

# 4 Función strtok

La función strtok nos permite separar un string en partes, para ello recibe el string que queremos separar y a continuación debemos indicar los separadores que tendrá en cuenta.

Al intentar utilizar esta función en un ambiente paralelo, nos enfrentamos al problema en el que un hilo no sabe dónde se quedó el anterior hilo y por consiguiente separar una parte del string que otro hilo ya separó o, de lo contrario, no separar una parte del string que debía ser separado. Éste problema se soluciona utilizando la función strtok-r la cual es thread safe, debido a que tiene un parámetro más el cual es un puntero que indica a cada hilo en qué parte del string se quedó el hilo que se ejecutó antes.