# CUADROS LAB 3

Luis Alberto Sánchez Moreno Colque

May 3, 2018

**Abstract**

Aquí se haran la comparación de tiempo en mutex y busy waiting y el uso de phtread

## 1 Busy waiting y Mutex

El mutex es mejor porque al momento de hacer las operaciones no lo hace de forma ordenada si no cuando termina cada thread continua otro y no espera

**Table 4.1** Run-Times (in Seconds) of $\pi$ Programs Using $n = 10^8$ Terms on a System with Two Four-Core Processors

| Threads | Busy-Wait | Mutex |
|---|---|---|
| 1 | 2.90 | 2.90 |
| 2 | 1.45 | 1.45 |
| 4 | 0.73 | 0.73 |
| 8 | 0.38 | 0.38 |
| 16 | 0.50 | 0.38 |
| 32 | 0.80 | 0.40 |
| 64 | 3.56 | 0.38 |

## 2 Tiempo en Listas Enlazadas con Mutex

en el READ and Write es mas rapido porque mientras lee se pone cada pthread y va leyendo y escribiendo

**Table 4.3** Linked List Times: 1000 Initial Keys, 100,000 ops, 99.9% `Member`, 0.05% `Insert`, 0.05% `Delete`

| | Number of Threads | | | |
|---|---|---|---|---|
| **Implementation** | **1** | **2** | **4** | **8** |
| Read-Write Locks | 0.213 | 0.123 | 0.098 | 0.115 |
| One Mutex for Entire List | 0.211 | 0.450 | 0.385 | 0.457 |
| One Mutex per Node | 1.680 | 5.700 | 3.450 | 2.700 |

**Table 4.4** Linked List Times: 1000 Initial Keys, 100,000 ops, 80% `Member`, 10% `Insert`, 10% `Delete`

| | Number of Threads | | | |
|---|---|---|---|---|
| **Implementation** | **1** | **2** | **4** | **8** |
| Read-Write Locks | 2.48 | 4.97 | 4.69 | 4.71 |
| One Mutex for Entire List | 2.50 | 5.13 | 5.04 | 5.11 |
| One Mutex per Node | 12.00 | 29.60 | 17.00 | 12.00 |

# 3 Phtreads en matrix vector

Es mas eficiente cuando multiplicas 8 x 8M ya que solo trabaja en 2 componentes a diferencia de 8000x8000 y 8M x 8 sin embargo, es la enorme diferencia en la eficiencia a medida que aumenta el número de subprocesos. La eficiencia de dos hilos del programa con la entrada de 8 x 8, 000,000 es casi un 20% menor que la eficiencia del programa con las entradas 8, 000,000 x 8 y 8000 x 8000. La eficiencia de cuatro hilos del programa con la entrada de 8 x 8, 000,000 es casi un 60% menor que la eficiencia del programa con la entrada de 8, 000,000 x 8 y más del 60% menos que la eficiencia del programa con la entrada de 8000 x 8000.

**Table 4.5** Run-Times and Efficiencies of Matrix-Vector Multiplication (times are in seconds)

| | Matrix Dimension | | | | | |
|---|---|---|---|---|---|---|
| | $8,000,000 \times 8$ | | $8000 \times 8000$ | | $8 \times 8,000,000$ | |
| **Threads** | **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 |

# References

[1] Book. an introduction to parallel programming, nov 2011.