

# CS4532 Concurrent Programming

## Take Home Lab 1

### Learning Outcomes

In this lab we will learn how to develop parallel programs using the Pthread library. We will also compare the performance of different implementations. At the end of the lab you will be able to:

- develop a simple program to solve a concurrent problem using threads
- collect performance data and use them to compare the performance of different implementations used to solve a problem

### Challenge

Implement a linked list as a:

- a) Serial program
- b) Parallel program (based on Pthreads) with one mutex for the entire linked list
- c) Parallel program (based on Pthreads) with read-write locks for the entire linked list

Your implementation should support *Member()*, *Insert()*, and *Delete()* functions. Populate the linked list with  $n$  random, but unique values. Make sure to set a different random seed for each execution. Each value should be between 0 and  $2^{16} - 1$ . Then perform  $m$  random *Member*, *Insert*, and *Delete* operations (simultaneously) on the link list. Let  $m_{Member}$ ,  $m_{Insert}$ , and  $m_{Delete}$  be the fractions of operations of each type. You may use any values within 0 and  $2^{16} - 1$  while performing these three operations. However, to simplify the implementation, a new value inserted into the list cannot be a value already in the list (it could be a value that was initially added to the list, but later removed).

**Step 1:** Design a solution such that you can generate  $m_{Member}$ ,  $m_{Insert}$ , and  $m_{Delete}$  operations of each type using given number of threads. Briefly explain your design. [2 marks]

**Step 2:** Implement the link list using above three approaches.

You may use the code snippets discussed in the class slides as the starting point and complete the missing pieces. [4 marks]

**Step 3:** Run your program sufficient number of times under the following three cases, and then fill up the given tables using the execution time of your programs. While determining the execution time, consider only the time taken to perform  $m$  operations and ignore other overheads such as initially populating the link list. [4 marks]

Your performance results should be within an accuracy of  $\pm 5\%$  and 95% confidence level. See Slides 13-33 and 13-34 by Raj Jain ([http://www.cse.wustl.edu/~jain/cse567-08/ftp/k\\_13cs.pdf](http://www.cse.wustl.edu/~jain/cse567-08/ftp/k_13cs.pdf)) to find out how to calculate the number of samples.

You should run this program on a machine with at least four physical CPU cores (Hyper Threading is not considered as a separate core. Thus, use a 4/4 or 4/8 core/thread CPU).

Also, list the specification of the machine that you used to run your simulations (see <http://dilumb.blogspot.com/2013/12/describing-experimentalsimulation-setup.html> for some tips).

**Step 4:** Plot the average execution time against the number of threads for each case. [6 marks]

See <http://www.dbooth.net/mhs/common/graphrules.html> for some tips on “How to Make a Good Graph”.

**Step 5:** Comment your observations while critically evaluating your findings. You may plot additional graphs to support your discussion. [4 marks]

While it is identified that sometimes results do not show a consistent pattern, in most cases it is due to errors in code. Hence, you are advisable to double check your implementations in such cases.

### Case 1

$n = 1,000$  and  $m = 10,000$ ,  $m_{Member} = 0.99$ ,  $m_{Indert} = 0.005$ ,  $m_{Delete} = 0.005$

Implementation	No of threads							
	1		2		4		8	
	Average	Std	Average	Std	Average	Std	Average	Std
Serial	52.587	28.866						
One mutex for entire list	51.647	27.933	68.984	30.130	74.734	30.890	77.595	32.353
Read-Write lock	52.359	28.149	37.874	19.245	24.556	11.712	19.266	7.065

### Case 2

$n = 1,000$  and  $m = 10,000$ ,  $m_{Member} = 0.90$ ,  $m_{Indert} = 0.05$ ,  $m_{Delete} = 0.05$

Implementation	No of threads							
	1		2		4		8	
	Average	Std	Average	Std	Average	Std	Average	Std
Serial	285.977	123.306						
One mutex for entire list	284.875	112.857	308.624	114.860	316.258	119.205	327.641	123.308
Read-Write lock	282.715	115.123	237.290	105.596	202.087	121.570	138.534	45.054

### Case 3

$n = 1,000$  and  $m = 10,000$ ,  $m_{Member} = 0.50$ ,  $m_{Indert} = 0.25$ ,  $m_{Delete} = 0.25$

Implementation	No of threads							
	1		2		4		8	
	Average	Std	Average	Std	Average	Std	Average	Std
Serial	367.743	70.787						
One mutex for entire list	368.389	71.246	404.804	75.015	498.122	173.418	424.447	78.050
Read-Write lock	369.376	73.318	378.560	69.492	402.099	71.56	385.953	70.830

### Notes

- Each lab group consists of 2 students. You are free to select your lab buddy.
- While the students are expected to find out how to develop programs using Pthreads, students are also encouraged to talk to the lecturer in charge for any clarifications especially related to pre-generation of workload and proper timing.

### What to Submit

- Submit following files as a single .zip file
  - Source files
  - README.txt explaining how to run your program
- Answers to steps 2, 3, and 4 as a .pdf file
- Name the .zip file as **lab1\_<index no 1>\_<index no 2>.zip**. Replace <index no x> with your index number