CS 636 Semester 2024-2025-II

Assignment 2

Submitted by:
Vraj Patel
Roll Number: 241110080

Submission Date:
April 12, 2025

Department of Computer Science & Engineering
Your Institute Name

Contents

1 Introduction

This assignment contains following implementations:

1. A concurrent closed-chaining-based hash table using Pthreads
2. An unbounded, total, lock-free concurrent queue
3. A concurrent Bloom filter

1.1 Declaimer

PLEASE ADD ALL FOUR BIN FILES IN BIN DIRECTORY

1.2 Compilation Instructions

The provided Makefile supports compilation of all three problems using the following commands:

```
# To compile Problem 1
make p1

# To compile Problem 1 with TBB
make p1_tbb

# To compile Problem 2
make p2

# To compile Problem 3
make p3

# To compile all problems
make all

# To run tests for each problem
make p1_test
make p2_test
make p3_test

# To run benchmarks (with default 4 threads)
make p1_benchmark
make p2_benchmark
make p3_benchmark

# To compare pthread and TBB implementations
make p1_compare
make p2_compare

# To clean build files
make clean

# To clean binary data files
```

```
make clean_bin

# To clean everything
make clean_all
```

Listing 1: Compilation Commands

2 Problem 1: Concurrent Hash Table

2.1 Performance Analysis

Results of performance measurements for different batch sizes:

Table 1: Hash Table Performance Measurements on 4 threads

Operation	10^5 operations	10^6 operations	10^7 operations
batch_insert	1.00×10^8 ops/sec	3.33×10^8 ops/sec	3.57×10^8 ops/sec
batch_delete	1.00×10^8 ops/sec	5.00×10^8 ops/sec	3.70×10^8 ops/sec
batch_lookup	1.00×10^8 ops/sec	5.00×10^8 ops/sec	3.57×10^8 ops/sec

Table 2: Hash Table Performance Measurements on 8 threads

Operation	10^5 operations	10^6 operations	10^7 operations
batch_insert	1.00×10^8 ops/sec	3.33×10^8 ops/sec	4.55×10^8 ops/sec
batch_delete	1.00×10^8 ops/sec	5.00×10^8 ops/sec	4.76×10^8 ops/sec
batch_lookup	1.00×10^8 ops/sec	3.33×10^8 ops/sec	4.55×10^8 ops/sec

(Please note that due to lack of time, timings given below are not average but are in fact from only a single run)

2.2 Comparison with Intel TBB

Comparison of your implementation with Intel TBB's concurrent hash table:

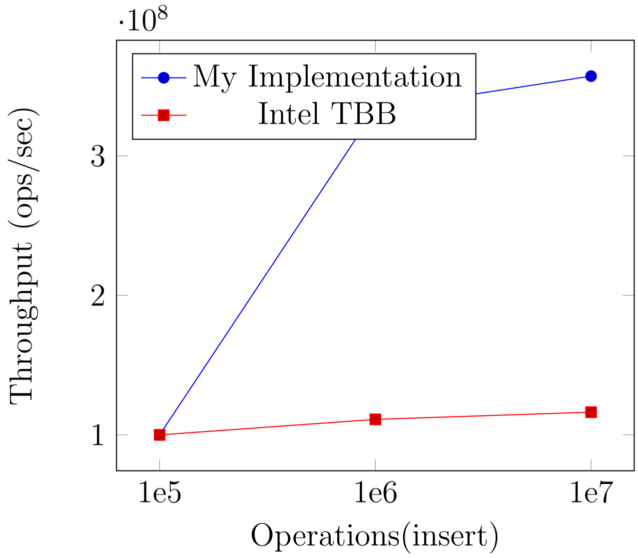


Figure 1: Performance comparison between custom implementation and Intel TBB

3 Problem 2: Lock-Free Queue

3.1 Performance Analysis

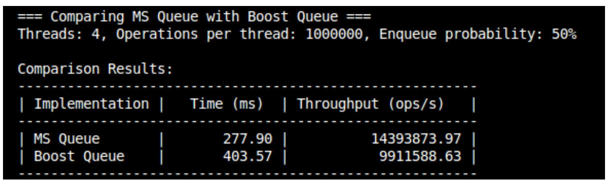
Results of performance measurements:

Table 3: Lock-Free Queue Performance Measurements with 4 threads

Operation Mix	10^5 operations	10^6 operations	10^7 operations
50% enq, 50% deq	1.35×10^7 ops/sec	1.34×10^7 ops/sec	1.35×10^7 ops/sec

3.2 Comparison with Boost Library

Comparison with Boost's lock-free queue:


```
=== Comparing MS Queue with Boost Queue ===
Threads: 4, Operations per thread: 1000000, Enqueue probability: 50%

Comparison Results:
------------------------------------------------------------
| Implementation |   Time (ms)  | Throughput (ops/s)   |
------------------------------------------------------------
| MS Queue       |       277.90 |          14393873.97 |
| Boost Queue    |       403.57 |           9911588.63 |
------------------------------------------------------------
```

Figure 2: Comparison with 3 threads and 10^6 operations

4 Problem 3: Concurrent Bloom Filter

4.1 Performance Analysis

Results of performance measurements:

Table 4: Bloom Filter Performance Measurements

Operation Mix	10^5 operations	10^6 operations	10^7 operations
50% add, 50% contains	8.62×10^7 ops/sec	8.60×10^7 ops/sec	8.60×10^7 ops/sec

4.2 False Positive Rate Analysis

Analysis of false positive rates:

Table 5: Bloom Filter False Positive Rates

Metric	10^5 operations	10^6 operations	10^7 operations
False Positive Rate	$1.32 \times 10^{-3}\%$	$2.44 \times 10^{-4}\%$	$6.72 \times 10^{-2}\%$
Theoretical FP Rate	$7.10 \times 10^{-7}\%$	$6.25 \times 10^{-4}\%$	$2.07 \times 10^{-1}\%$