# CS610: Assignment 2 Report

Name : Nishant
Roll No: 251110053

Semester 2025–2026-I

## Problem 1: Naive and Blocked 3D Convolution

### Implementation

Implemented both **naive 3D convolution** and **blocked convolution**. The blocked version improves cache locality by operating on sub-blocks of the input.

### How to Run

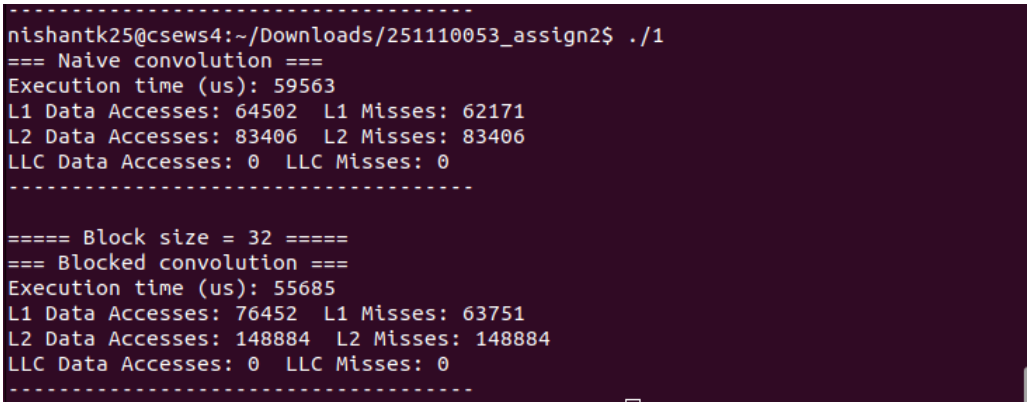
```
g++ 251110053-prob1.cpp -o prob1 -lpapi

# or simply
make

# Execution:
./prob1
```

### Experimental Setup

- Setup used for every problem is CSEWS 4.
- Input size: $64 \times 64 \times 64$.
- Each configuration was run 5 times and average was reported.

### Results

1. The least time taken was found for block size = 32. However, this minimum is not stable and changes across different runs.
2. Naive convolution average time = 61.9812 ms
   Blocked convolution average time = 55.5378 ms (for block size = 32).
3. Papi library counters are tested,as L1 cache misses are less for blocked convolution than for naive convolution.



```
-------------------------------------
nishantk25@csews4:~/Downloads/251110053_assign2$ ./1
=== Naive convolution ===
Execution time (us): 59563
L1 Data Accesses: 64502  L1 Misses: 62171
L2 Data Accesses: 83406  L2 Misses: 83406
LLC Data Accesses: 0  LLC Misses: 0
-------------------------------------

===== Block size = 32 =====
=== Blocked convolution ===
Execution time (us): 55685
L1 Data Accesses: 76452  L1 Misses: 63751
L2 Data Accesses: 148884  L2 Misses: 148884
LLC Data Accesses: 0  LLC Misses: 0
-------------------------------------
```

Figure 1: PAPI counters for naive vs blocked convolution.(time in microseconds)

### Observations

Blocking reduced cache misses and gave lower execution time compared to the naive version. Best performance was observed at block size 32, although the results were not strictly consistent across all runs. In some cases, the naive version appeared faster than the blocked version, which can happen due to increased load/store operations and variability in cache behavior.

## Problem 2: Multithreaded Word and Line Count

### Performance Bugs Identified

**False Sharing**
The `word_count` array in `struct tracker` caused false sharing since consecutive threads updated adjacent elements. To fix this, we padded the array to place each thread's counter on a separate cache line (`thread_id*8`). This eliminated false sharing. (Implemented in `prob2_padded.cpp`).

**True Sharing**
Locks were taken for every word/line update, causing high contention and HITMs. We reduced this by using thread-local counters and merging them only once, removing redundant locks. This nearly eliminated HITMs. (Implemented in `prob2_final.cpp`).

### Fixes Implemented

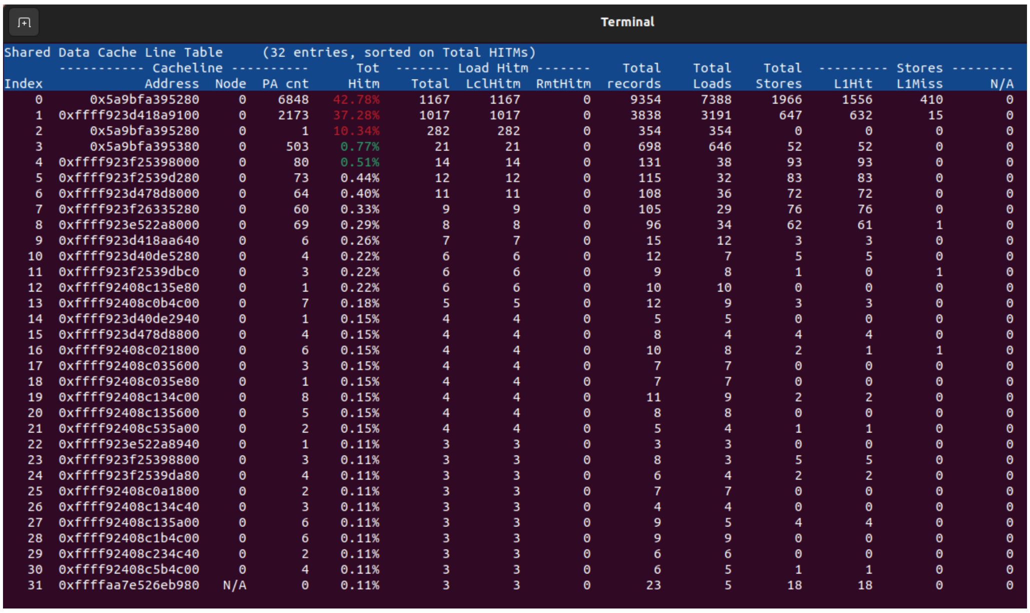
- `prob2_padded.cpp`: Added padding/alignment to remove **false sharing**.
- `prob2_final.cpp`: Instead of updating the shared counter for every word or line (causing contention), each thread maintained its own local count and only updated the global counter once at the end. This approach effectively removed **true sharing**.

Each thread was executed with **15 MB text files** as input for obtaining the results.

### How to Run

```
g++ -std=c++17 original.cpp -o original -pthread
g++ -std=c++17 prob2_padded.cpp -o prob2_padded -pthread
g++ -std=c++17 prob2_final.cpp -o prob2_final -pthread
```

```
# or simply

make

# Execution:
./original <num_of_threads> <path_to_file>
./prob2_padded <num_of_threads> <path_to_file>
./prob2_final <num_of_threads> <path_to_file>



# With performance counters:
perf c2c record ./prob2_padded 5 input.txt
perf c2c report
```



Terminal

Shared Data Cache Line Table (32 entries, sorted on Total HITMs)

| | Cacheline | | | Tot | Load Hitm | | | Total | Total | Total | Stores | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Index | Address | Node | PA cnt | Hitm | Total | LclHitm | RmtHitm | records | Loads | Stores | L1Hit | L1Miss | N/A |
| 0 | 0x5a9bfa395280 | 0 | 6848 | 42.78% | 1167 | 1167 | 0 | 9354 | 7388 | 1966 | 1556 | 410 | 0 |
| 1 | 0xffff923d418a9100 | 0 | 2173 | 37.28% | 1017 | 1017 | 0 | 3838 | 3191 | 647 | 632 | 15 | 0 |
| 2 | 0x5a9bfa395280 | 0 | 1 | 10.34% | 282 | 282 | 0 | 354 | 354 | 0 | 0 | 0 | 0 |
| 3 | 0x5a9bfa395380 | 0 | 503 | 0.77% | 21 | 21 | 0 | 698 | 646 | 52 | 52 | 0 | 0 |
| 4 | 0xffff923f25398000 | 0 | 80 | 0.51% | 14 | 14 | 0 | 131 | 38 | 93 | 93 | 0 | 0 |
| 5 | 0xffff923f2539d280 | 0 | 73 | 0.44% | 12 | 12 | 0 | 115 | 32 | 83 | 83 | 0 | 0 |
| 6 | 0xffff923d478d8000 | 0 | 64 | 0.40% | 11 | 11 | 0 | 108 | 36 | 72 | 72 | 0 | 0 |
| 7 | 0xffff923f26335280 | 0 | 60 | 0.33% | 9 | 9 | 0 | 105 | 29 | 76 | 76 | 0 | 0 |
| 8 | 0xffff923e522a8000 | 0 | 69 | 0.29% | 8 | 8 | 0 | 96 | 34 | 62 | 61 | 1 | 0 |
| 9 | 0xffff923d418aa640 | 0 | 6 | 0.26% | 7 | 7 | 0 | 15 | 12 | 3 | 3 | 0 | 0 |
| 10 | 0xffff923d40de5280 | 0 | 4 | 0.22% | 6 | 6 | 0 | 12 | 7 | 5 | 5 | 0 | 0 |
| 11 | 0xffff923f2539dbc0 | 0 | 3 | 0.22% | 6 | 6 | 0 | 9 | 8 | 1 | 0 | 1 | 0 |
| 12 | 0xffff92408c135e80 | 0 | 1 | 0.22% | 6 | 6 | 0 | 10 | 10 | 0 | 0 | 0 | 0 |
| 13 | 0xffff92408c0b4c00 | 0 | 7 | 0.18% | 5 | 5 | 0 | 12 | 9 | 3 | 3 | 0 | 0 |
| 14 | 0xffff923d40de2940 | 0 | 1 | 0.15% | 4 | 4 | 0 | 5 | 5 | 0 | 0 | 0 | 0 |
| 15 | 0xffff923d478d8800 | 0 | 4 | 0.15% | 4 | 4 | 0 | 8 | 4 | 4 | 4 | 0 | 0 |
| 16 | 0xffff92408c021800 | 0 | 6 | 0.15% | 4 | 4 | 0 | 10 | 8 | 2 | 1 | 1 | 0 |
| 17 | 0xffff92408c035600 | 0 | 3 | 0.15% | 4 | 4 | 0 | 7 | 7 | 0 | 0 | 0 | 0 |
| 18 | 0xffff92408c035e80 | 0 | 1 | 0.15% | 4 | 4 | 0 | 7 | 7 | 0 | 0 | 0 | 0 |
| 19 | 0xffff92408c134c00 | 0 | 8 | 0.15% | 4 | 4 | 0 | 11 | 9 | 2 | 2 | 0 | 0 |
| 20 | 0xffff92408c135600 | 0 | 5 | 0.15% | 4 | 4 | 0 | 8 | 8 | 0 | 0 | 0 | 0 |
| 21 | 0xffff92408c535a00 | 0 | 2 | 0.15% | 4 | 4 | 0 | 5 | 4 | 1 | 1 | 0 | 0 |
| 22 | 0xffff923e522a8940 | 0 | 1 | 0.11% | 3 | 3 | 0 | 3 | 3 | 0 | 0 | 0 | 0 |
| 23 | 0xffff923f25398800 | 0 | 3 | 0.11% | 3 | 3 | 0 | 8 | 3 | 5 | 5 | 0 | 0 |
| 24 | 0xffff923f2539da80 | 0 | 4 | 0.11% | 3 | 3 | 0 | 6 | 4 | 2 | 2 | 0 | 0 |
| 25 | 0xffff92408c0a1800 | 0 | 2 | 0.11% | 3 | 3 | 0 | 7 | 7 | 0 | 0 | 0 | 0 |
| 26 | 0xffff92408c134c40 | 0 | 3 | 0.11% | 3 | 3 | 0 | 4 | 4 | 0 | 0 | 0 | 0 |
| 27 | 0xffff92408c135a00 | 0 | 6 | 0.11% | 3 | 3 | 0 | 9 | 5 | 4 | 4 | 0 | 0 |
| 28 | 0xffff92408c1b4c00 | 0 | 6 | 0.11% | 3 | 3 | 0 | 9 | 9 | 0 | 0 | 0 | 0 |
| 29 | 0xffff92408c234c40 | 0 | 2 | 0.11% | 3 | 3 | 0 | 6 | 6 | 0 | 0 | 0 | 0 |
| 30 | 0xffff92408c5b4c00 | 0 | 4 | 0.11% | 3 | 3 | 0 | 6 | 5 | 1 | 1 | 0 | 0 |
| 31 | 0xffffaa7e526eb980 | N/A | 0 | 0.11% | 3 | 3 | 0 | 23 | 5 | 18 | 18 | 0 | 0 |

Figure 2: Perf c2c report for original implementation showing high HITM counts.

Terminal

Shared Data Cache Line Table (32 entries, sorted on Total HITMs)

| | Cacheline | | | Tot | Load Hitm | | | Total | Total | Total | Stores | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Index | Address | Node | PA cnt | Hitm | Total | LclHitm | RmtHitm | records | Loads | Stores | L1Hit | L1Miss | N/A |
| 0 | 0x591829a003c0 | 0 | 8118 | 44.44% | 1102 | 1102 | 0 | 11228 | 10027 | 1201 | 1183 | 18 | 0 |
| 1 | 0xffff923d4189adc0 | 0 | 1863 | 34.27% | 850 | 850 | 0 | 3333 | 2777 | 556 | 545 | 11 | 0 |
| 2 | 0x591829a003c0 | 0 | 1 | 11.37% | 282 | 282 | 0 | 366 | 366 | 0 | 0 | 0 | 0 |
| 3 | 0x591829a00480 | 0 | 450 | 0.97% | 24 | 24 | 0 | 642 | 595 | 47 | 47 | 0 | 0 |
| 4 | 0xffff923db2bca940 | 0 | 84 | 0.69% | 17 | 17 | 0 | 123 | 40 | 83 | 83 | 0 | 0 |
| 5 | 0xffff923e27e8d280 | 0 | 68 | 0.48% | 12 | 12 | 0 | 106 | 38 | 68 | 68 | 0 | 0 |
| 6 | 0xffff923db2bcd280 | 0 | 51 | 0.32% | 8 | 8 | 0 | 80 | 24 | 56 | 56 | 0 | 0 |
| 7 | 0xffff923f2dbd0000 | 0 | 60 | 0.24% | 6 | 6 | 0 | 102 | 25 | 77 | 77 | 0 | 0 |
| 8 | 0xffff92408c035e80 | 0 | 1 | 0.24% | 6 | 6 | 0 | 10 | 10 | 0 | 0 | 0 | 0 |
| 9 | 0xffff92408c2b5600 | 0 | 8 | 0.24% | 6 | 6 | 0 | 14 | 14 | 0 | 0 | 0 | 0 |
| 10 | 0xffff92408c4a1800 | 0 | 4 | 0.24% | 6 | 6 | 0 | 10 | 8 | 2 | 1 | 1 | 0 |
| 11 | 0xffff923db2bca980 | 0 | 3 | 0.20% | 5 | 5 | 0 | 6 | 6 | 0 | 0 | 0 | 0 |
| 12 | 0xffff92408c2b5e80 | 0 | 1 | 0.20% | 5 | 5 | 0 | 8 | 8 | 0 | 0 | 0 | 0 |
| 13 | 0xffff92408c3a1800 | 0 | 3 | 0.20% | 5 | 5 | 0 | 7 | 5 | 2 | 1 | 1 | 0 |
| 14 | 0xffff923d8b37a940 | 0 | 43 | 0.16% | 4 | 4 | 0 | 88 | 15 | 73 | 73 | 0 | 0 |
| 15 | 0xffff923f2dbd0800 | 0 | 3 | 0.16% | 4 | 4 | 0 | 6 | 5 | 1 | 1 | 0 | 0 |
| 16 | 0xffff92408c035a00 | 0 | 5 | 0.16% | 4 | 4 | 0 | 7 | 4 | 3 | 3 | 0 | 0 |
| 17 | 0xffff92408c2a1800 | 0 | 9 | 0.16% | 4 | 4 | 0 | 11 | 8 | 3 | 1 | 2 | 0 |
| 18 | 0xffff92408c2b4c00 | 0 | 11 | 0.16% | 4 | 4 | 0 | 17 | 14 | 3 | 3 | 0 | 0 |
| 19 | 0xffff92408c3b5a00 | 0 | 4 | 0.16% | 4 | 4 | 0 | 5 | 4 | 1 | 1 | 0 | 0 |
| 20 | 0xffff92408c434c00 | 0 | 3 | 0.16% | 4 | 4 | 0 | 6 | 6 | 0 | 0 | 0 | 0 |
| 21 | 0xffff92408c4b4c00 | 0 | 7 | 0.16% | 4 | 4 | 0 | 13 | 10 | 3 | 3 | 0 | 0 |
| 22 | 0xffff92408c4b5a00 | 0 | 7 | 0.16% | 4 | 4 | 0 | 11 | 6 | 5 | 4 | 1 | 0 |
| 23 | 0xffff92408c535600 | 0 | 2 | 0.16% | 4 | 4 | 0 | 6 | 6 | 0 | 0 | 0 | 0 |
| 24 | 0xffff923d40faa940 | 0 | 3 | 0.12% | 3 | 3 | 0 | 6 | 5 | 1 | 1 | 0 | 0 |
| 25 | 0xffff923d40fad280 | 0 | 3 | 0.12% | 3 | 3 | 0 | 8 | 5 | 3 | 3 | 0 | 0 |
| 26 | 0xffff923db2bcdbc0 | 0 | 1 | 0.12% | 3 | 3 | 0 | 4 | 4 | 0 | 0 | 0 | 0 |
| 27 | 0xffff92408c2b5a00 | 0 | 5 | 0.12% | 3 | 3 | 0 | 8 | 5 | 3 | 3 | 0 | 0 |
| 28 | 0xffff92408c3b4c40 | 0 | 2 | 0.12% | 3 | 3 | 0 | 3 | 3 | 0 | 0 | 0 | 0 |
| 29 | 0xffff92408c435a00 | 0 | 4 | 0.12% | 3 | 3 | 0 | 5 | 4 | 1 | 1 | 0 | 0 |
| 30 | 0xffff92408c4b4c40 | 0 | 1 | 0.12% | 3 | 3 | 0 | 3 | 3 | 0 | 0 | 0 | 0 |
| 31 | 0xffff92408c534c40 | 0 | 1 | 0.12% | 3 | 3 | 0 | 3 | 3 | 0 | 0 | 0 | 0 |

Figure 3: Perf c2c report for padded implementation, HITMs decreased.

Terminal

Shared Data Cache Line Table (0 entries, sorted on Total HITMs)

| | Cacheline | | | Tot | Load Hitm | | | Total | Total | Total | Stores | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Index | Address | Node | PA cnt | Hitm | Total | LclHitm | RmtHitm | records | Loads | Stores | L1Hit | L1Miss | N/A |

Figure 4: Perf c2c report for final implementation, removing both false and true sharing.

### Results

| Version | Runtime (s) | HITMs |
|---|---|---|
| Original | 1.781 | 2633 |
| Padded version | 1.213 | 2398 |
| Final version | 0.090 | 0 |

Table 1: Performance comparison of different versions for Problem 2

### Observations

Padding reduced false sharing, while batching updates in the final version eliminated true sharing. This gave the best scalability and lowest runtime.

## Problem 3: Lock Implementations

### Implementation

We implemented:

- **Filter Lock**: Uses a filter hierarchy to avoid simultaneous entry, ensures fairness but overhead grows with threads.

- **Bakery Lock**: Assigns tickets to threads, guaranteeing fairness, but can be slow under contention.
- **Spin Lock**: Simple busy-waiting lock, very fast in low contention but wastes CPU cycles when many threads compete.
- **Ticket Lock**: Each thread gets a ticket, ensures FIFO fairness, but suffers from cache-line bouncing at scale.
- **Array Queue Lock**: Threads enqueue themselves in an array-based queue; provides fairness but higher overhead.

### How to Run

```
g++ -std=c++17 251110053-prob3.cpp -o prob3 -pthread

# or simply
make

# Execution:
./prob3
```

### Results

Taken N=1000 to obtain results

| Threads | 1 | 2 | 4 | 8 | 16 | 32 | 64 |
|---|---|---|---|---|---|---|---|
| Pthread Mutex | 0 | 0 | 0 | 15 | 42 | 157 | 632 |
| Filter Lock | 0 | 2 | 16 | 56 | 460 | 10057 | 4645660 |
| Bakery Lock | 0 | 2 | 0 | 8 | 74184 | 2853862 | 35306462 |
| Spin Lock | 0 | 0 | 0 | 32 | 35 | 65 | 233 |
| Ticket Lock | 0 | 0 | 0 | 0 | 155780 | 2076062 | 25444157 |
| Array Q Lock | 0 | 2 | 0 | 0 | 90 | 2567431 | 20191492 |

Table 2: Performance of different lock implementations (measured in microseconds)

### Observations

- Spin locks were fastest in most cases.
- Bakery locks were observed to be the slowest in most cases.
- Bakery and Ticket locks ensure fairness but perform poorly with many threads.
- Array Queue Lock is fair but suffers at scale.
- Pthread mutex is stable and well-balanced.