COL380 A0 Report

Koduru Suchith (2021CS10572)

1 Execution Time vs Matrix Size

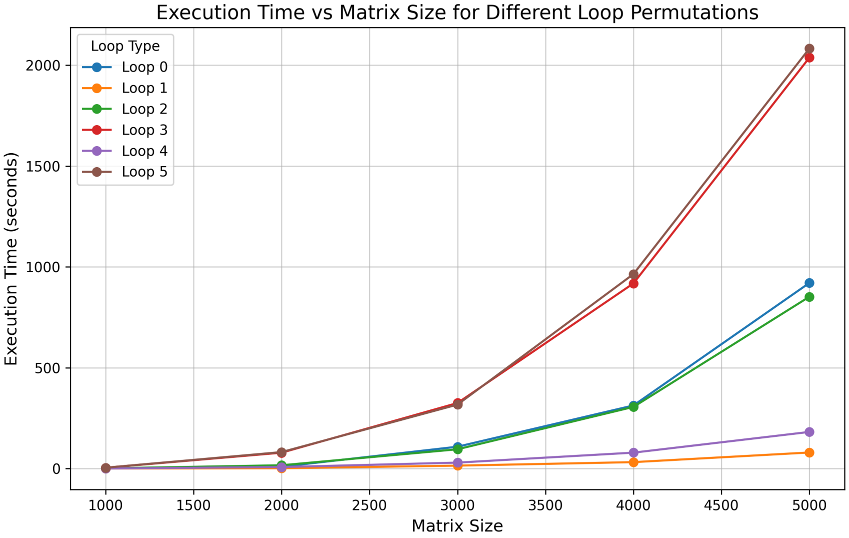

Figure 1: Execution Time vs Matrix Size

2 Execution Time vs Matrix Size with perf

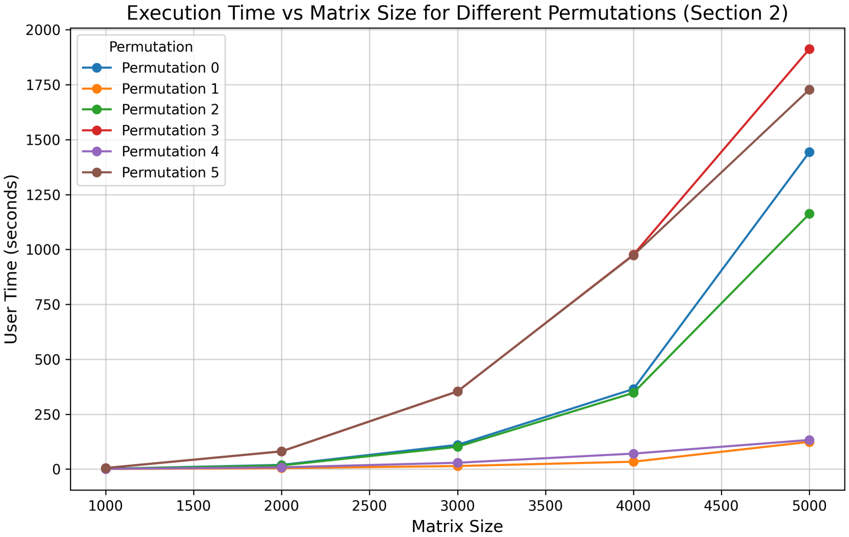

Figure 2: Execution Time vs Matrix Size

The times reported by `perf` differ because it captures the precise CPU time spent on computation, excluding overheads such as I/O operations that the Python script may include.

3 Matrix Multiplication Time Only

In this experiment, only the matrix multiplication time is considered, excluding file read/write times or memory allocation times. The graph is shown below:

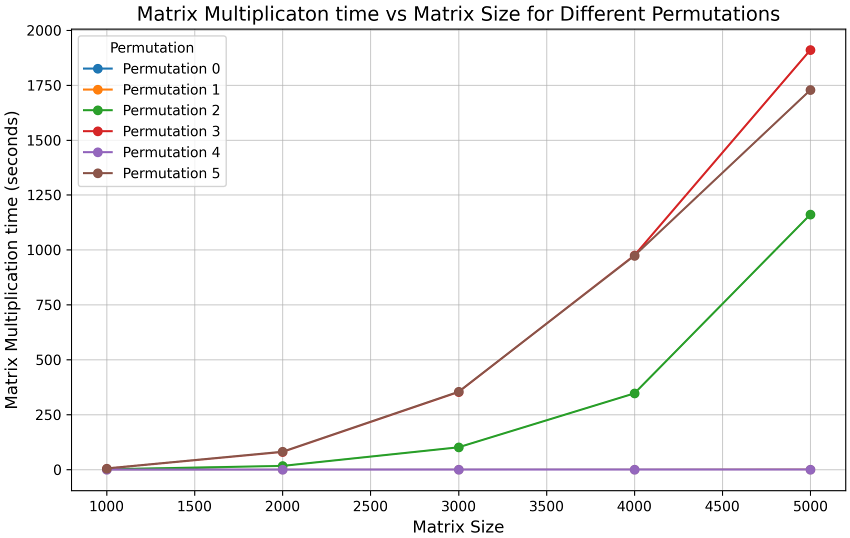

Figure 3: Matrix Multiplication Time Only

4 Cache-Hit Rate vs Matrix Size

This section shows the results of Experiment 2, but with cache-hit rate on the Y-axis instead of execution time. The plotted graph is given below:

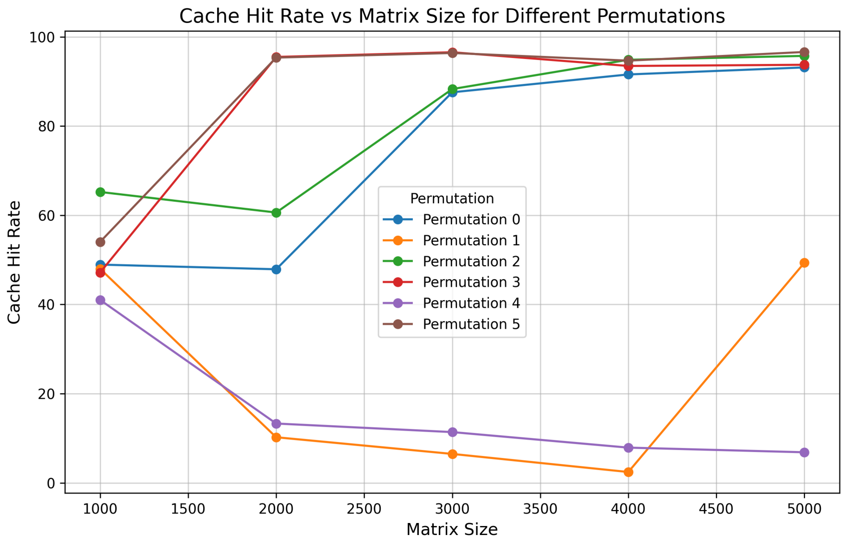

Figure 4: Cache-Hit Rate vs Matrix Size

5 Best Loop Permutation Analysis

The following table summarizes the average execution time (user time) and cache-hit rate for each loop permutation:

Permutation	**Execution Time (User Time)**	**Cache-Hit Rate (%)**
0	387.9304	73.8215
1	**35.4698**	23.3061
2	325.9446	80.9416
3	665.5014	85.2802
4	48.4220	16.1031
5	628.5444	**87.4036**

Observations:

1. **Execution Time (User Time)**: - The fastest execution time is achieved by **permutation 1** with an average time of **35.4698 units**. - The second-fastest is **permutation 4**, with an average time of **48.4220 units**. - The slowest execution time is for **permutation 3**, with an average time of **665.5014 units**.

2. **Cache-Hit Rate**: - The highest cache-hit rate is observed for **permutation 5**, achieving **87.4036%**. - The second-highest is for **permutation 3**, at **85.2802%**. - The lowest cache-hit rate is for **permutation 4**, with just **16.1031%**.

Key Findings:

- There is no single loop permutation that optimizes both execution time and cache-hit rate.

- **Permutation 1** achieves the lowest execution time but has a relatively poor cache-hit rate of **23.3061%**. This suggests that it is efficient in CPU usage but may not be utilizing the cache effectively.
- **Permutation 5** achieves the best cache-hit rate (**87.4036%**) but has a high execution time of **628.5444 units**. This indicates that while the cache usage is optimal, other factors such as computation or memory access patterns may slow it down.
- **Permutation 3** offers a balance, with a high cache-hit rate (**85.2802%**) and a moderate execution time (**665.5014 units**).
- **Permutation 4** has the second-fastest execution time (**48.4220 units**) but suffers from the lowest cache-hit rate (**16.1031%**), making it less efficient for cache-sensitive workloads.