A

MINI PROJECT REPORT

ON

"Implement the matrix multiplication. Also implement multithreaded matrix multiplication with either one thread per row or one thread per cell. Analyze and compare their performance.

In the partial fulfillment of the requirement for the Degree in

Computer Engineering

Submitted By

Gauri Vishwanath Dighe 4133
Pratik Sitaram Dighe 4134

Under the Guidance of Guide:-

Prof. Pandit S. R.

Submitted for the Course of Final Year in Computer Engineering for the Practical Lab "**Laboratory Practice III(Design and Analysis of Algorithm)**"

Department of Computer Engineering

AMRUTVAHINI COLLEGE OF ENGINEERING, SANGAMNER

For the academic year 2024-2025

1. **Title of Mini-Project:** Implement the matrix multiplication. Also implement multithreaded matrix multiplication with either one thread per row or one thread per cell. Analyze and compare their performance.

2. **Aims/Benefits of Mini-Project**: To implement matrix multiplication and multithreaded matrix multiplication with two different approaches: one thread per row and one thread per cell. Compare and analyze the performance of these methods. Matrix multiplication is a fundamental operation in many scientific, engineering, and mathematical applications. In this project, we implement standard matrix multiplication and two multithreaded variations, one using one thread per row and the other using one thread per cell. We aim to compare their performance and efficiency as the matrix size increases.

3. **Course Outcomes:**

1. Implement matrix multiplication using single-threaded and multithreaded approaches.
2. Compare the execution times of:
    1. Single-threaded matrix multiplication.
    2. Multithreaded matrix multiplication with one thread per row.
    3. Multithreaded matrix multiplication with one thread per cell.
3. Analyze the complexity and performance gains of the multithreaded approach.

4. **Actual Methodology Followed:**

Algorithm Implementation:

1. Single-Threaded Matrix Multiplication

- Initialize the result matrix with zeros.
- Perform matrix multiplication by iterating over rows and columns.
- Store the dot product of corresponding rows and columns in the result matrix.

2. Multithreaded Matrix Multiplication (Thread per Row)

- For each row in matrix **A**, create a thread to compute all corresponding elements in the result matrix row.
- Start the thread and wait for all threads to finish.

3. Multithreaded Matrix Multiplication (Thread per Cell)

- For each element in the result matrix, create a thread to compute the dot

product of the corresponding row from matrix **A** and the column from matrix **B**.

Start the thread and wait for all threads to finish

Performance Evaluation:

- **Single-threaded**: Time complexity is O(n3)O(n^3)O(n3), where nnn is the size of the matrix.
- **Multithreaded (Thread per Row)**: Complexity remains O(n3)O(n^3)O(n3) in terms of total work, but execution time reduces due to parallelism. Maximum speedup is nnn threads.
- **Multithreaded (Thread per Cell)**: Complexity is still O(n3)O(n^3)O(n3), but with maximum parallelism of n2n^2n2 threads. This approach provides the highest potential speedup.

5. Actual Resources used:

Sr. No	Name of resources/ Materials	Specifications	Qty.	Remark
1.	Computer System	10^{th} generation, i5, RAM 8 GB, ROM- ITB	**1**	__
2.	Software	Jupyter Notebook	**1**	__
3.	Text editor	Microsoft word	**1**	__
4.	Information Source	https://www.geeksforgeeks.org/multiplication-of-matrix-using-threads/	**1**	__

6. Theory:

Matrix Multiplication

Matrix multiplication involves the dot product of rows from the first matrix and columns from the second matrix. The result matrix has dimensions determined by the number of rows in the first matrix and the number of columns in the second matrix.

For matrices **A** (m x n) and **B** (n x p), the element Cij in the resulting matrix **C** (m x p) is given by:

$$C_{ij} = \sum_{k=1}^{n} A_{ik} \times B_{kj}$$

Multi-threaded Matrix Multiplication

Multithreading allows parallel computation of independent parts of the matrix. In matrix multiplication:

1. **Thread per row**: Each row of the result matrix is computed by a different thread.
2. **Thread per cell**: Each individual element of the result matrix is computed by a separate thread.

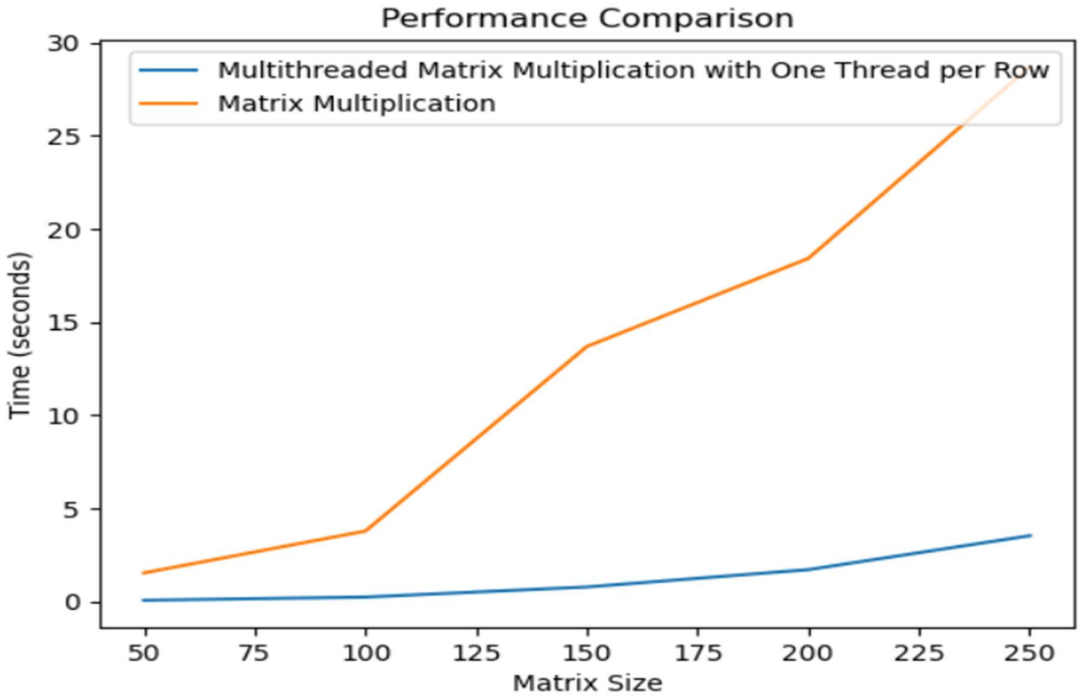
Performance Considerations

- **Single-threaded approach** processes all elements sequentially.
- **Thread-per-row** processes each row concurrently, offering a moderate speedup.
- **Thread-per-cell** maximizes concurrency by parallelizing every cell computation, potentially leading to higher speedups, especially for larger matrices.

OUTCOME:

- A standard matrix multiplication algorithm and two multithreaded variations were implemented.
- Multithreaded approaches provided noticeable performance improvements, particularly for larger matrix sizes.

7. **Output:**
   The performance analysis showed that the multithreaded approach, particularly the thread-per-cell version, significantly reduced computation time for larger matrices.



8. Conclusion:

In this project, we successfully implemented matrix multiplication using both single-threaded and multithreaded approaches. We evaluated and compared the performance of the standard method against two multithreaded variations: one thread per row and one thread per cell.

- The **single-threaded** method worked well for small matrices but struggled with larger matrices due to its O(n3)O(n^3)O(n3) complexity.
- The **thread-per-row** approach showed moderate speedup by parallelizing row-wise computations, making it efficient for medium-sized matrices.
- The **thread-per-cell** approach demonstrated the highest level of concurrency and provided the best performance gains for large matrices. However, the overhead of managing a large number of threads can become significant for small matrix sizes.