|  |  |  |
| --- | --- | --- |
| **MARATHWADA MITRA**  **MANDAL’S COLLEGE OF**  **ENGINEERING**  **Karvenagar, Pune**   Permnently Affiliated to SPPU | Accredited with ‘A++’ grade by NAAC | Recipient of ‘Best College’ award by SPPU, Accredited by NBA ( Electrical, Mechanical, E&TC, IT Engg.) | Recognized under 2(f) and 12(B) of UGC Act 1956 |ISO 9001:2015 | |  | | --- | |  | |

|  |
| --- |
| certified |

**Department of Computer Engineering**

|  |
| --- |
| BE DAA Mini Project |

|  |
| --- |
| **Team Members:** |

Mandar Joshi BC127   
Omkar Kadam BC128   
Akshay Malayil BC145

|  |
| --- |
| Aniket Lakhe BC141 |

|  |  |
| --- | --- |
| |  | | --- | | Problem Statement | |

|  |
| --- |
| Write a program to implement matrix multiplication. Also |

|  |
| --- |
| implement multithreaded matrix multiplication with either one |

thread per row or one thread per cell. Analyze and compare their performance.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Algorithm | | | | |  | | --- | | **Multithreaded Matrix Multiplication (Per Row)**: | | | |
| **Single-Threaded Matrix Multiplication**: | | |
| **1.** | **Input**: Matrices matA and matB, both of size MAX | | **1.** | |  | | --- | | **Input**: Matrices matA and matB, both of size MAX | | |
| x MAX. | | | x MAX. | | |
| **2.** | **Initialization**: | | **2.** | **Thread Assignment**: | |
| ○ | | |  | | --- | | Create NUM\_THREADS , each responsible for | |
| ○ | | Create a result matrix matC of size MAX |
| a specific row range. | | |
| x MAX, initialized with zeros. | | |
| ○ | | Divide rows among threads to calculate |
| **3.** | **Computation**: | |
| specific segments of the result matrix. | | |
| ○ | | Loop through each row i of matA. |
| |  | | --- | | 3. | | **Computation**: Each thread computes the product for its |   assigned rows using the same logic as the   |  | | --- | | single-threaded algorithm. |  |  |  | | --- | --- | | |  | | --- | | 4. | | | **Synchronization**: Use pthread\_join to ensure all |  |  | | --- | | threads complete before moving forward. | | | |
| ○ | | Loop through each column j of matB. |
| ○ | | For each element matC[i][j], calculate |
| the sum of the product of elements from row | | |
| i of matA and column j of matB. | | |
| ○ | | Update matC[i][j] with this value. |
| **4.** | **Output**: The resulting matrix matC. | |
| 5. | **Output**: The resulting matrix | |

matC.

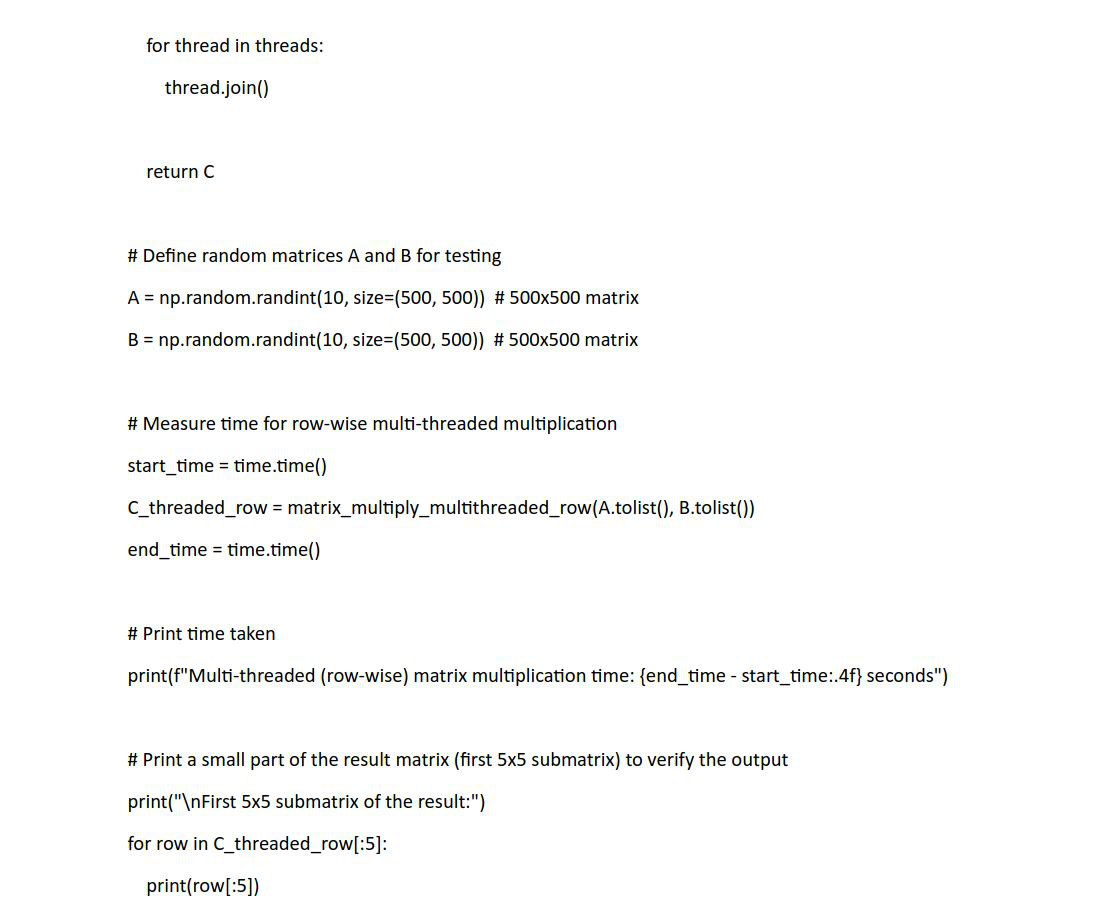
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Example | | | |  | | --- | | **Matrix C (Result of A x B)**: |   Calculate each element in matC: | | |
| Consider two 3x3 matrices: **Matrix A**: | |
| ● | |  | | --- | | **Element matC[0][0]**: | | |
| |  | | --- | | 1 | | 2 |
| ○ | | (1\*9) + (2\*6) + (3\*3) = 9 + 12 + 9 = 30 |
| 3 | |
| ● | **Element matC[0][1]**: | |
| 4 | 5 | ○ | | (1\*8) + (2\*5) + (3\*2) = 8 + 10 + 6 = 24 |
| ● | **Element matC[0][2]**: | |
| 6 | |
| ○ (1\*7) + (2\*4) + (3\*1) = 7 + 8 + 3 = 18  Continuing this way for all elements:  **Matrix C**: | | |
| 7 | 8 |
| **Matrix B**: 9 | |
| 9 | 8 | 30 | |  | | --- | | 24 | | |
| 7 | | 18 | | |
| 6 | 5 | 84 | 69 | |
| 54 | | |
| 4 | |
| 138 114  90 | | |
| 3 | 2 |
| 1 | |

|  |  |
| --- | --- |
| |  | | --- | | Comparative Analysis | |
| |  | | --- | | **Single-threaded: O(n³)**  across threads, the computation can be parallelized,  ideally reducing the overall time.  If there are p threads and p ≤ n, the time complexity fo  the computation phase is reduced to O(n³ / p).  However, this is an ideal scenario. In practice, due to  thread synchronization and overhead, the speedup is  sublinear.  **Final Complexity**: The practical time complexity r  **Multithreaded (Per Row)**  **Theoretical Time Complexity**: The main computation  involves multiplying matrices, which is still inherently  O(n³) since each element of the resulting matrix  requires O(n) operations, and there are n² elements  in total.  **Practical Complexity with Threads**: By dividing rows | |

remains O(n³ / p + T\_overhead) , where   
T\_overhead represents the overhead for managing   
threads and synchronization.

|  |  |
| --- | --- |
| |  | | --- | | Code | |

|  |  |
| --- | --- |
| |  | | --- | |  | |



|  |  |  |
| --- | --- | --- |
| |  | | --- | | Output | |  |
| |  | | --- | |  | | |  | | --- | |  | |