A. The codes, output files for mixed and chuck techniques of threading are separated to read time elapsed separately.

MIXED

Matrix Class:

- The `Matrix` class is designed to represent a square matrix.
- It reads matrix data from an input file ("inp.txt").
- The matrix multiplication result is stored in the matrix `P`.
- It has a method `multiply` to perform matrix multiplication for a given row.
- The `Mixed` static function is used as the thread function.

Threaded Matrix Multiplication:

- The matrix multiplication is parallelized using pthreads.
- The `Mixed` function is the entry point for each thread. It multiplies rows of the matrix concurrently.
- The `__sync_fetch_and_add` function is used to atomically increment the `count` variable, which represents the current row being processed by a thread.
- Threads are created and joined in the `main` function using pthreads.

File I/O:

- The program reads matrix data from "inp.txt" and writes the final result to "out_B.txt."
- The initial matrix data is read in the constructor of the `Matrix` class.
- The final result, along with the CPU time used, is written to "out_B.txt" in the `printFinal` method.

Timing:

- The program uses the `clock` function to measure the CPU time used for matrix multiplication.
- The start and end times are recorded, and the difference is calculated to determine the elapsed time.

Thread Management:

- Pthreads (`pthread_t`) are used for thread creation and management.
- The number of threads is determined by the value of `K`, which is read from the input file.

Improvements and Notes:

- The input matrix is initialized with simple values (i + j) for testing purposes. In a real application, it would be populated based on actual data.

Output:

- The final result of matrix multiplication and the CPU time used are printed to "out_B.txt."

This C++ code performs matrix multiplication in parallel using pthreads. Here's a simple report on its working and techniques:

CHUNK

Matrix Class:

- The `Matrix` class is designed to represent a square matrix.
- It reads matrix data from an input file ("inp.txt").
- The matrix multiplication result is stored in the matrix `P`.
- It has a method `multiply` to perform matrix multiplication for a given row.
- The `Chunk` static function is used as the thread function.

Threaded Matrix Multiplication:

- The matrix multiplication is parallelized using pthreads with a chunk-based approach.
- The `Chunk` function is the entry point for each thread. It multiplies a chunk of rows of the matrix concurrently.
- Each thread processes a specific range of rows, avoiding the need for explicit synchronization mechanisms.
- The `count` variable is used to keep track of the current chunk being processed, and each thread increments it atomically.

File I/O:

- The program reads matrix data from "inp.txt" and writes the final result to "out_A.txt."
- The initial matrix data is read in the constructor of the `Matrix` class.
- The final result, along with the CPU time used, is written to "out_A.txt" in the `printFinal` method.

Timing:

- The program uses the `clock` function to measure the CPU time used for matrix multiplication.
- The start and end times are recorded, and the difference is calculated to determine the elapsed time.

Thread Management:

- Pthreads (`pthread_t`) are used for thread creation and management.
- The number of threads is determined by the value of `K`, which is read from the input file.

Improvements and Notes:

- The input matrix is initialized with simple values (i + j) for testing purposes. In a real application, it would be populated based on actual data.
- Error handling for file operations and thread creation is minimal in this example and could be enhanced for robustness.
- The code uses a chunk-based approach to distribute work among threads efficiently.

Output:

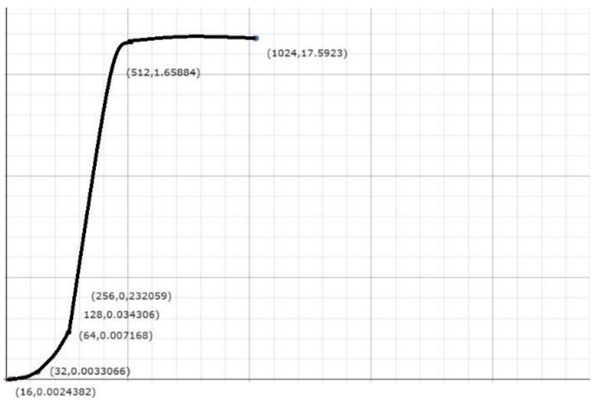
- The final result of matrix multiplication and the CPU time used are printed to "out_A.txt."

B.

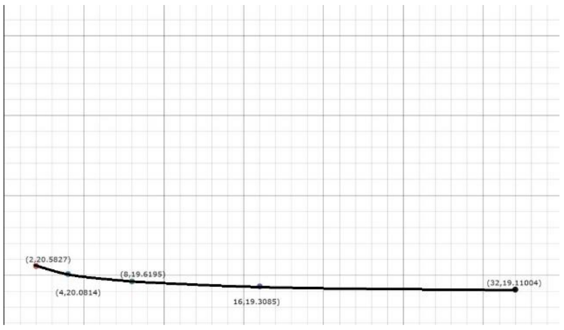
GRAPHS:

CHUNK:

T vs N:

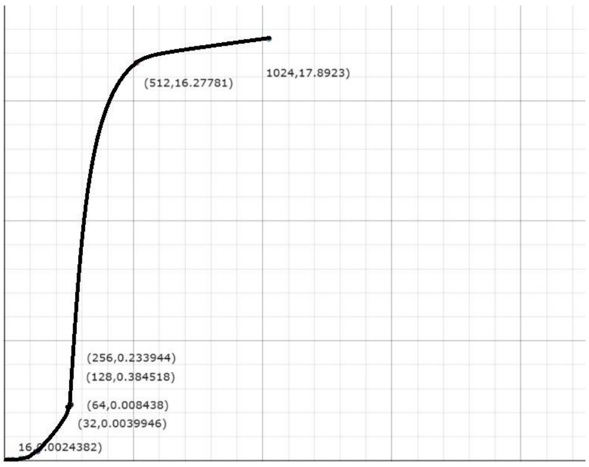


T vs K:

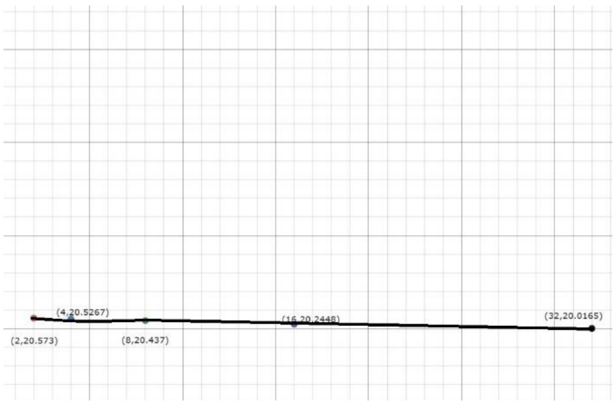


MIXED:

T vs N:



T vs K:



GRAPH ANALYSIS:

1.T vs N graph in both techniques increases as N increases T increases but there is a sudden spike in time taken at N=512 after that the time taken is almost same.

2.Mixed takes more time than Chunk in both T vs N and T vs K.

3.T vs V graph in both techniques decreases but the decrease in T is more in chunk than in mixed as the K increases.