CHUNK EXPERIMENT 1 :

1. Matrix Class: This class represents a square matrix and handles reading matrix data from a file, performing matrix multiplication, and storing the result. Key members include:

- N: Size of the square matrix.
- K: Number of threads to use for multiplication.
- C: Number of CPU cores to distribute threads across.
- BT: Number of threads to run per core.
- M: Pointer to store the matrix data.
- P: Pointer to store the result of matrix multiplication.
- count: A counter to keep track of the number of chunks processed by threads.

2. Constructor and Destructor:

- The constructor reads matrix data from a file ("inp.txt") and initializes member variables.
- The destructor deallocates dynamically allocated memory.

3. multiply Method: Performs matrix multiplication for a given row of the matrix.

4. Chunk Method: Static method to be used as the thread function. It takes a Matrix object pointer as an argument and performs matrix multiplication on chunks of rows. Each thread handles a portion of the rows based on the number of threads specified.

5. Main Function:

- Reads matrix size and thread count from "inp.txt", generates a matrix with dummy data, and initializes pthreads.
- Distributes threads and assigns affinity across CPU cores according to the specified number of cores (C) and the number of threads per core (BT).
- Joins threads after they finish execution.
- Calculates and prints the CPU time used for matrix multiplication.

6. Multithreading:

- The main function creates K pthreads, each assigned to execute the Chunk method.
- CPU affinity is set for each thread to distribute them evenly across CPU cores.

7. File Handling:

- The program reads matrix data from "inp.txt" and writes the resulting matrix to an output file ("out_A.txt").

- The output file also includes the CPU time used for matrix multiplication.

CHUNK EXPERIMENT 2 :

In main function the number of threads are divided into 2 halves and first half are assigned to first half of cores by setting affinity, rest half are assigned by CPU the codes for 2 are separated and the time taken also calculated separately. Rest are the same details as Experiment1 code.

MIXED EXPERIMENT1:

1. Matrix Class:

- The Matrix class represents a square matrix.

- Private member variables include:

  - N: Size of the matrix (number of rows/columns).

  - K: Number of threads to be used for matrix multiplication.

  - C: Number of cores to be utilized for thread affinity.

  - BT: Number of blocks to be used for thread affinity.

  - M: Pointer to the matrix data.

  - P: Pointer to store the result of matrix multiplication.

  - count: Counter for threads to synchronize their operations.

- Public member functions include:

  - Constructor: Reads matrix data from a file ("inp.txt") and initializes member variables.

  - Destructor: Frees dynamically allocated memory.

  - matrixPrint(): Prints the matrix.

  - multiply(int row): Performs matrix multiplication for a given row.

  - Getter functions to retrieve private member variables.

  - printFinal(const std::string& filename, double cpu_time_used): Prints the result matrix and CPU time used for computation to a file.

  - set_count(): Resets the count variable.

  - static void* Chunk(void* arg): Static method to be used as a thread function.

2. Main Function:

- Creates an instance of the `Matrix` class.

- Prints the input matrix.

- Obtains values of K, N, C, and BT.

- Updates the input file ("inp.txt") with the matrix size and parameters.

- Creates an array of pthreads and initializes pthread attributes.

- Divides the threads into two sets and assigns CPU affinity to each set.

- Measures CPU time used for each set of threads.

- Joins all threads.

- Prints the final result matrix and CPU time used for computation to an output file ("out_A.txt").

3. Thread Management:

- The matrix multiplication task is divided into chunks, with each thread responsible for processing a chunk of rows.

- The Chunk method is executed by each thread, which calculates the range of rows to process based on the current count value and performs matrix multiplication for those rows.

- Thread affinity is set using pthread_setaffinity_np to bind threads to specific cores for better performance.

- Threads are joined to wait for their completion before proceeding to print the final result.

MIXED EXPERIMENT 2 :

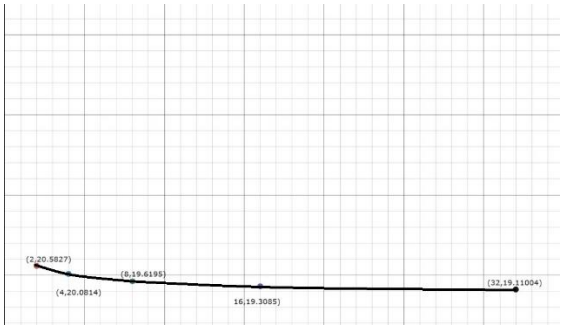
In main function the number of threads are divided into 2 halves and first half are assigned to first half of cores by setting affinity, rest half are assigned by CPU the codes for 2 are separated and the time taken also calculated separately. Rest are the same details as Experiment1 code.

GRAPHS:

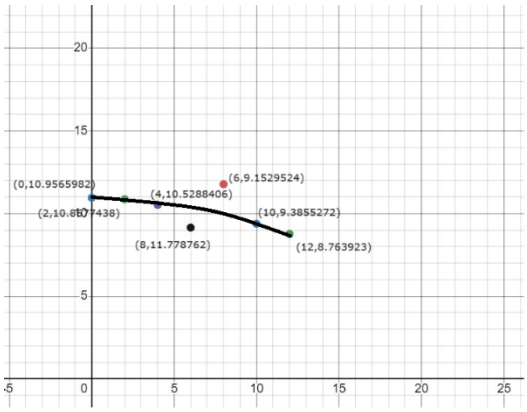
EXPERIMENT1:

CHUNK:

GRAPH1:

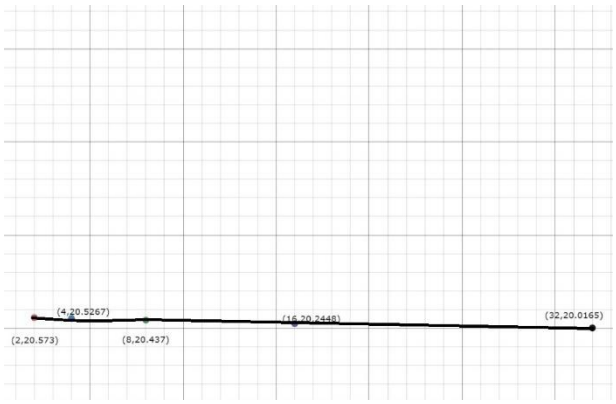


GRAPH2:

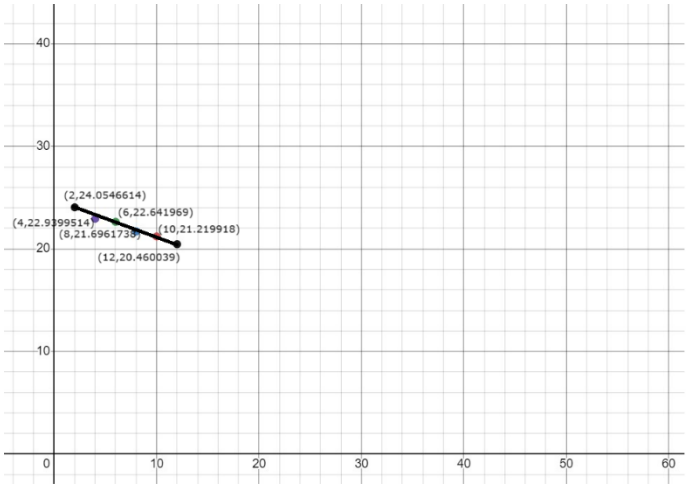


MIXED:

GRAPH3:



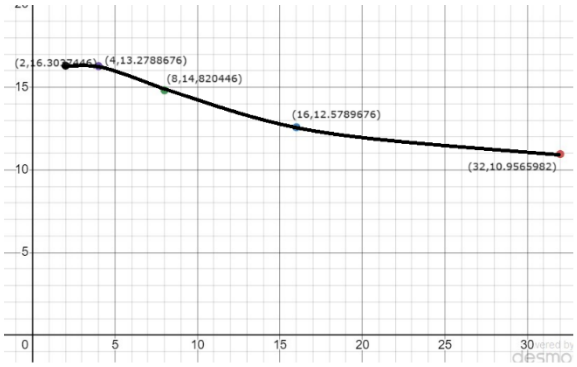
GRAPH4:



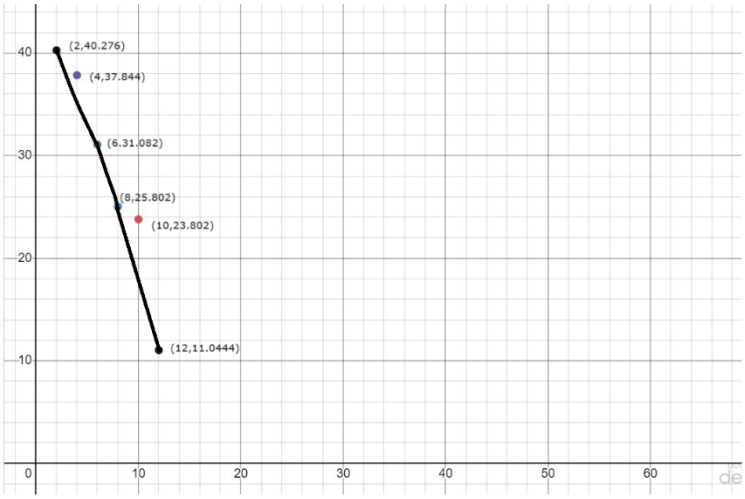
EXPERIMENT2:

CHUNK:

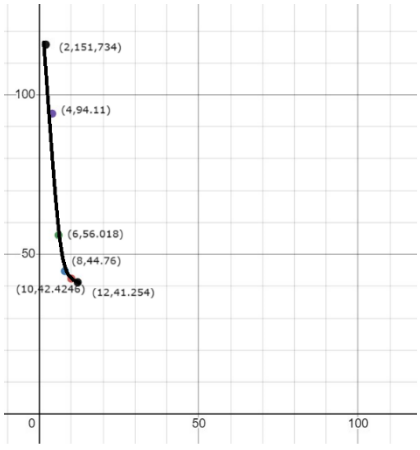
GRAPH1:



GRAPH2:

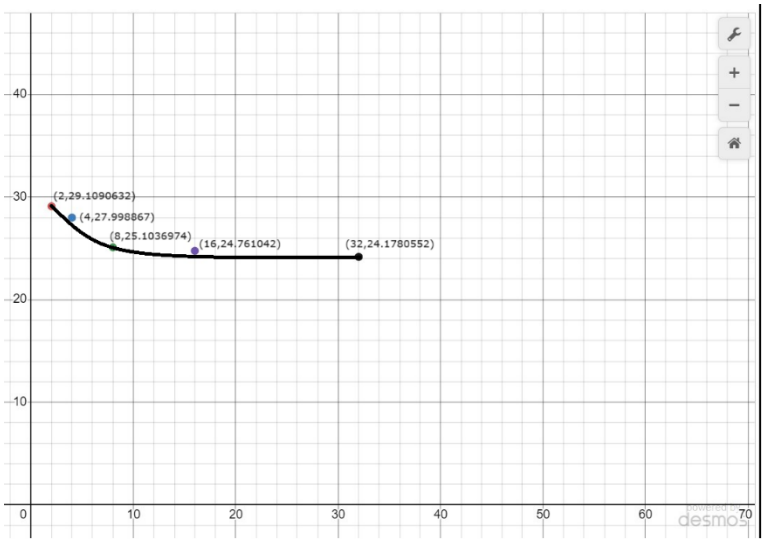


GRAPH3:

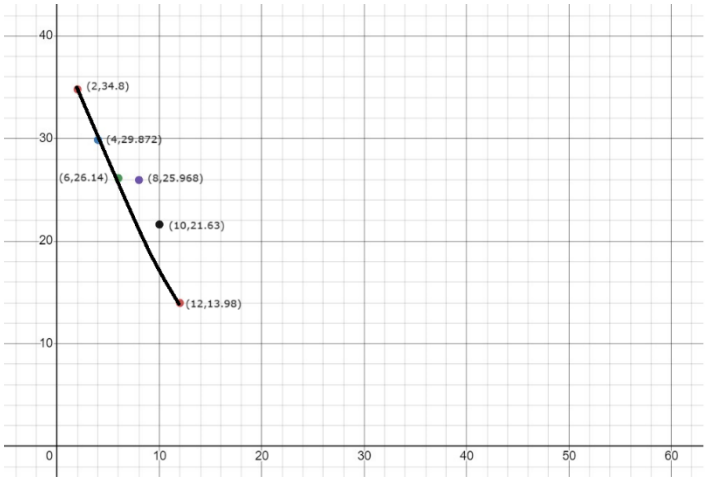


MIXED:

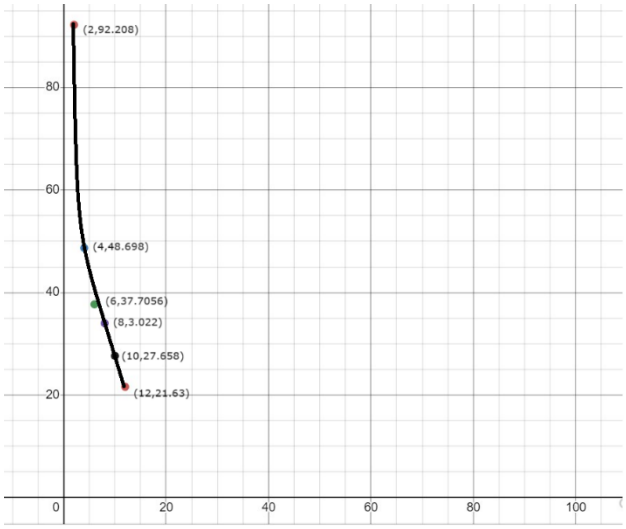
GRAPH4:



GRAPH5:



GRAPH6:



GRAPH ANALYSIS:

1.The time taken by assigning affinity to the CPU has decreased the time taken.As we can see the values in GRAPH1 are higher than values in GRAPH2 as well as the values in GRAPH4 are higher than values in GRAPH4.

2.Though the seems to be an increase in the time taken when k=32 from GRAPH3 TO 4

3.There seems to a linear decrease in time taken with respect to the increment in assignment of threads to cores in GRAPH4.

4.While in GRAPH2 there are points deviating from the expected decrement.

5.The taken for GRAPH1 is greater than GRAPH2 greater than GRAPH3 similarly the taken for GRAPH5 is greater than GRAPH5 greater than GRAPH6.

6.Both graph 3&6 seem to have a sudden drop in the time taken.

7.In GRAPH 2&4 the time taken is almost linearly decrementing.

8.The taken for CHUNK is greater than MIXED with respect to graph 3&6.