**CSE467 - Parallel and Distributed Computing**

**Assignment 5**

**Hiba Mallick - 24015**

**Objective: Write a program for parallel vector dot product by using multi threading.**

The complexity of vector dot product is O(N) and the execution time depends on the vector length. In this assignment you will learn how to pass and return values from a thread and how logically work can be divided into multiple threads to be performed in parallel and reduce total execution time. You need do device some logic for load balancing. **Use a Global variable for result and protect it from concurrent access by using mutex and use condition variables if desired.**

**Submit a word file containing all code, output snapshots and the comparison table and graph. Also provide your logic for data distribution and computing.**

**Input:**

1. Vectors Length: **N** (to be used to create and initialize 2 vectors)
2. No of Threads: **T**

**Processing:** Determine the execution time of vector addition with various values of T and N given in the table.

T0 = get clock cycle/system time

Perform DP = V2 . V3 with load balancing among T threads.

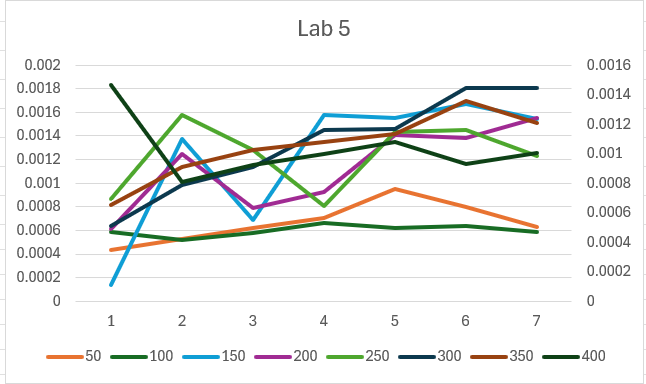
T1 = get clock cycle/system time

Time elapsed = T1 – T0

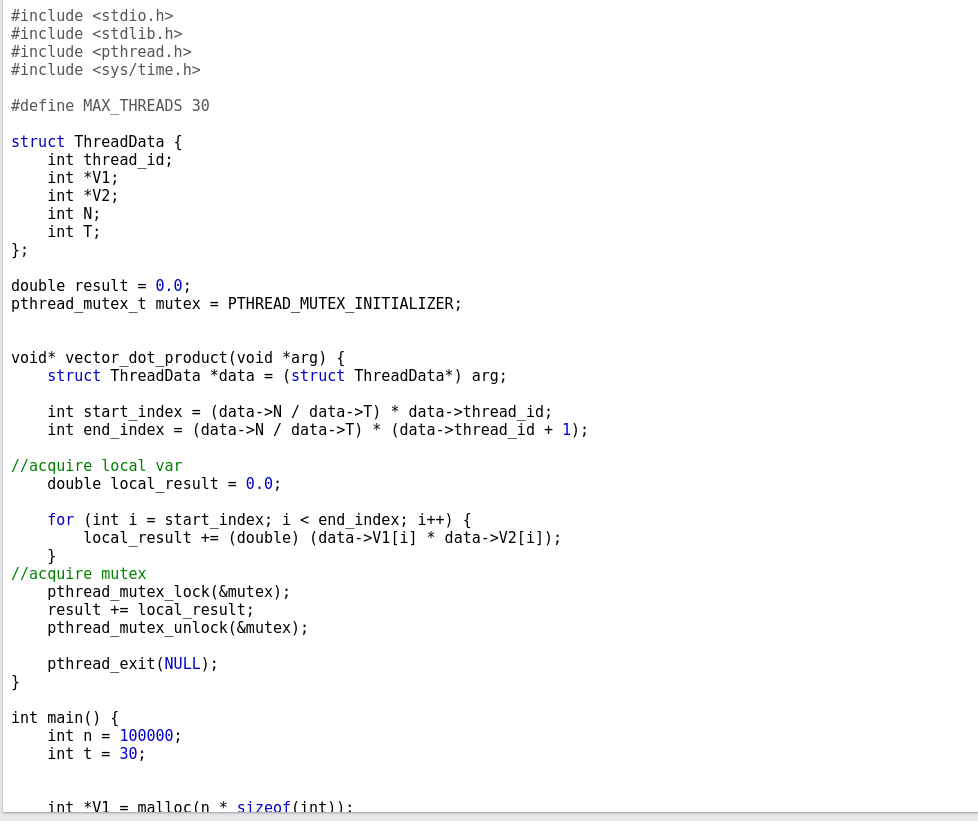
**Output:** Time elapsed - Tables show the comparison

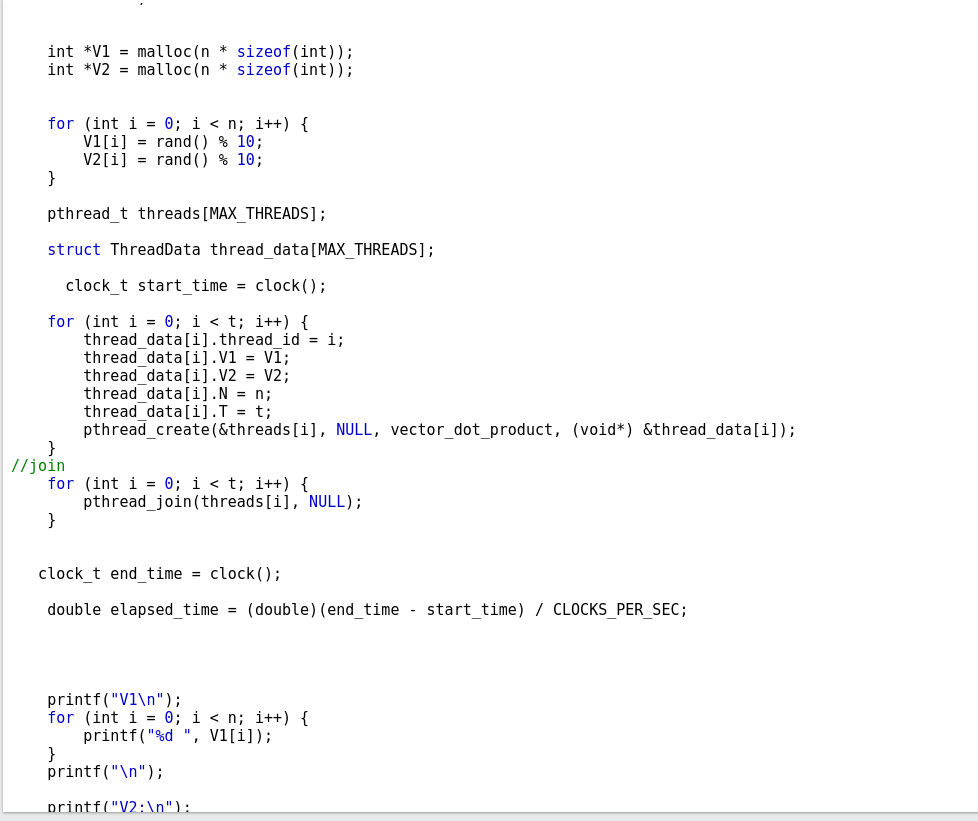
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | N (vector size in thousands) | | | | | | | |
|  |  | 50 | 100 | 150 | 200 | 250 | 300 | 350 | 400 |
| T (no of threads) | 1 | 0.000437 | 0.000584 | 0.000138 | 0.000615 | 0.000869 | 0.000636 | 0.000654 | 0.001467 |
| 5 | 0.000528 | 0.000518 | 0.001371 | 0.001245 | 0.001577 | 0.001116 | 0.000912 | 0.000805 |
| 10 | 0.000622 | 0.000578 | 0.000689 | 0.000788 | 0.001282 | 0.001139 | 0.001026 | 0.000921 |
| 15 | 0.000708 | 0.00066 | 0.001575 | 0.000928 | 0.000807 | 0.001447 | 0.00108 | 0.000996 |
| 20 | 0.000951 | 0.000623 | 0.001551 | 0.001412 | 0.001432 | 0.001462 | 0.001134 | 0.001082 |
| 25 | 0.000795 | 0.000639 | 0.001667 | 0.001385 | 0.001453 | 0.00181 | 0.001359 | 0.000929 |
| 30 | 0.000628 | 0.000584 | 0.001542 | 0.00155 | 0.001232 | 0.001805 | 0.00121 | 0.001007 |

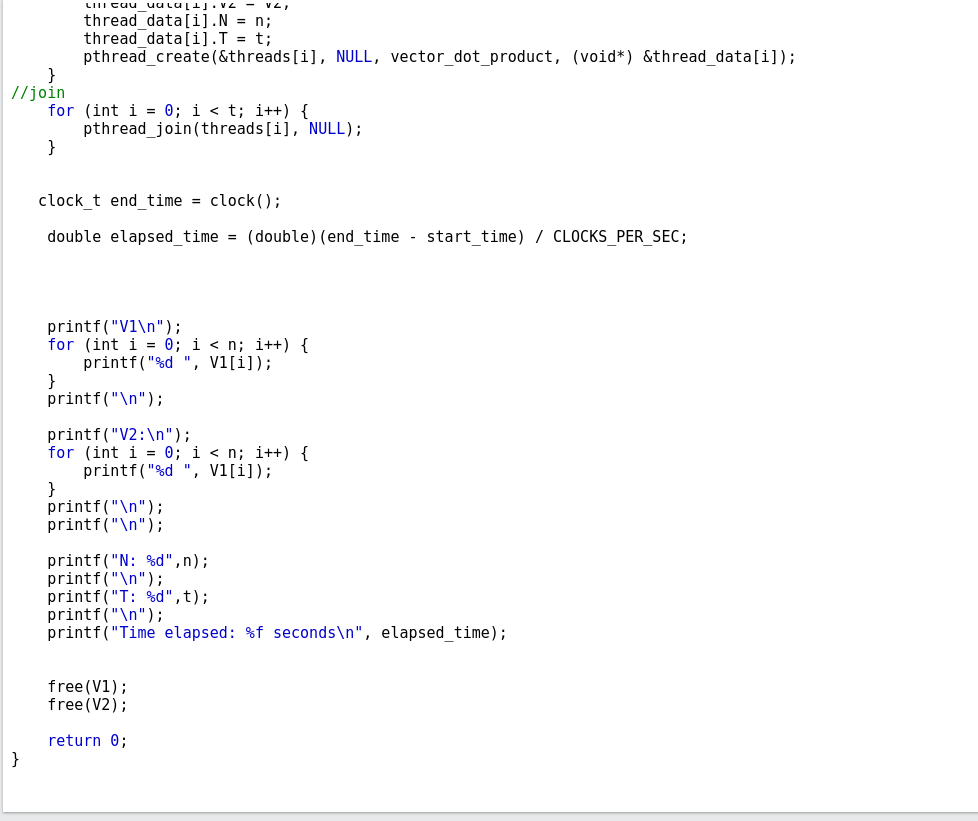
**Draw the Graph for above table.**



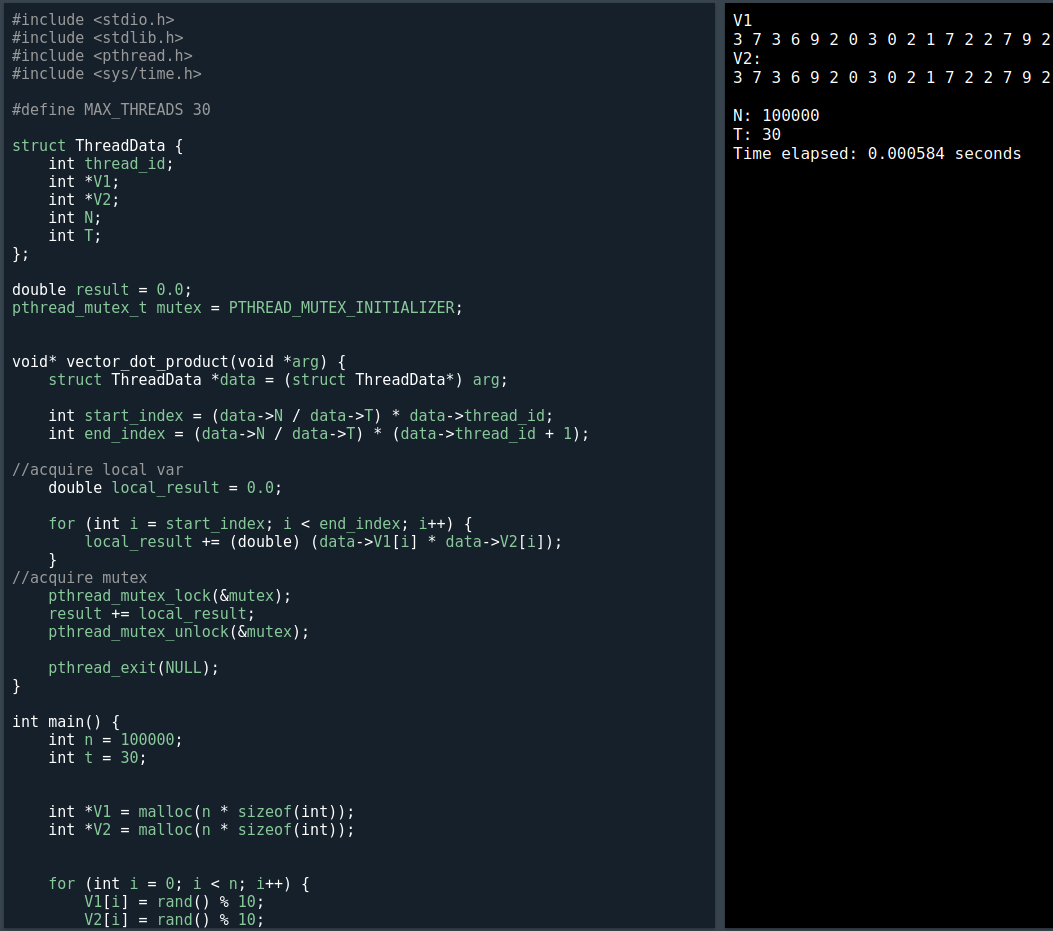
**Code:**



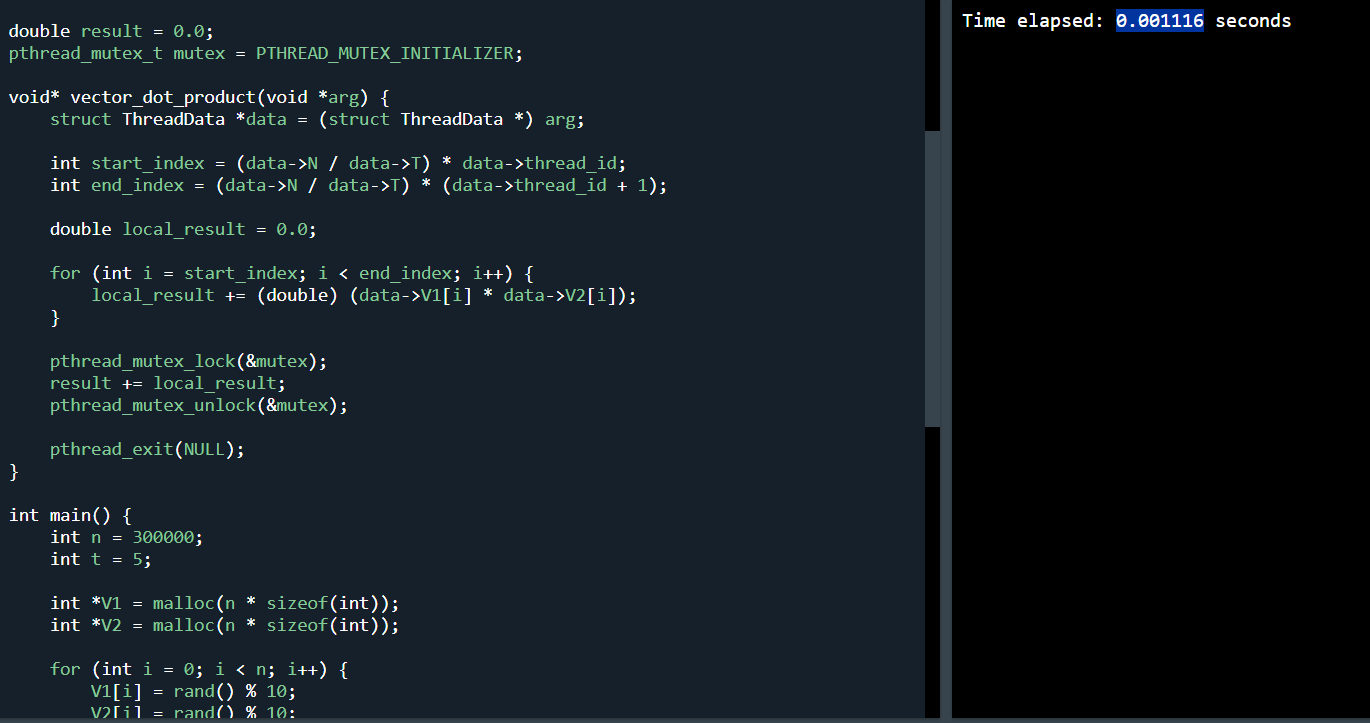




**Example Output:**



0.001116

****