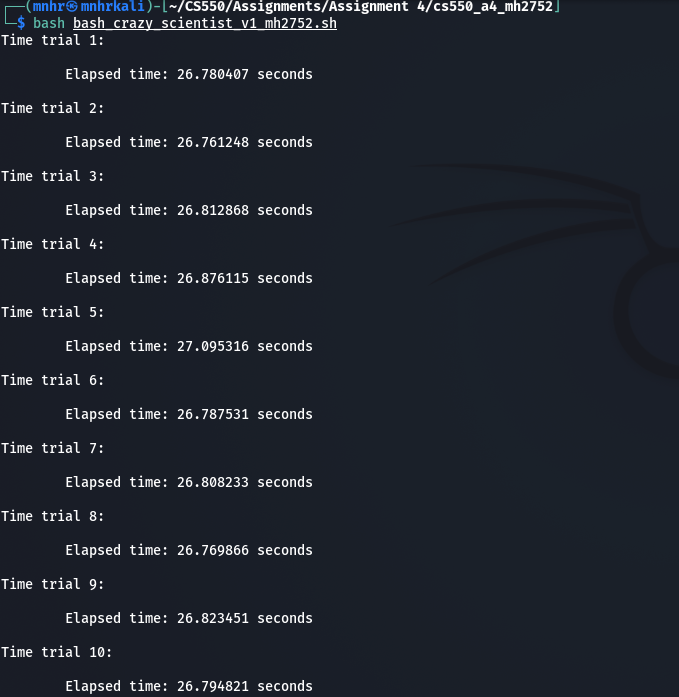
**Master Table:**

| **Question** | **Total Time** | **Individual Thread Response Times and Load Imbalance** |
| --- | --- | --- |
| 1 | 26.8309856 seconds | N/A |
| 2 | 26.5630927 seconds | Time t1 = 4.3462261 seconds  Time t2 = 26.5628915 seconds  Load imbalance = 22.2166648 seconds |
| 3 | 16.5639202 seconds | Time t1 = 15.8671298 seconds  Time t2 = 16.5637391 seconds  Load imbalance = 0.6966262 seconds |
| 4 | 16.529894 seconds | Time t1 = 16.3900127 seconds  Time t2 = 15.9638579 seconds  Load imbalance = 0.7055924 seconds |

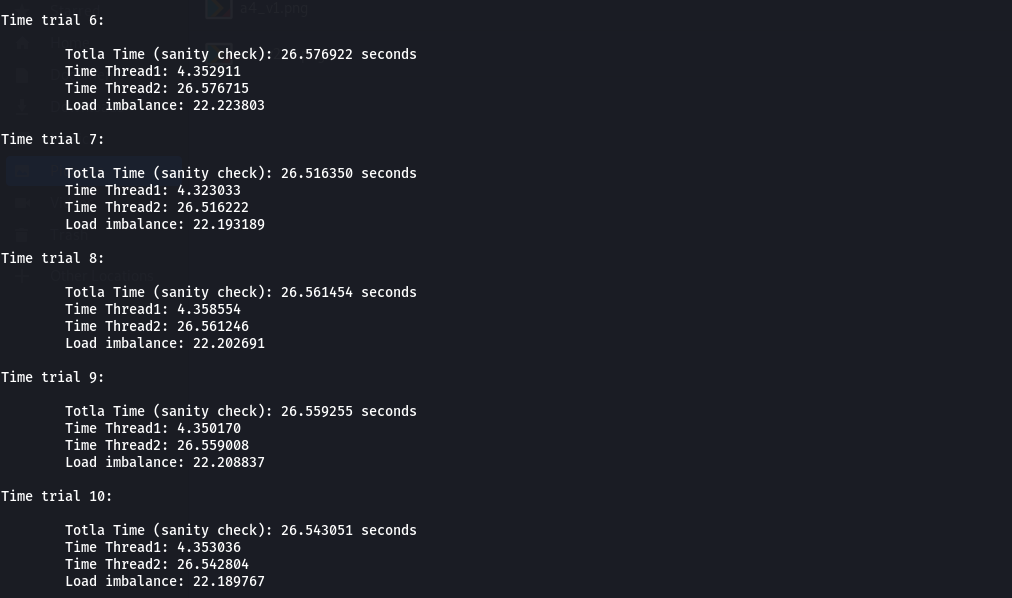
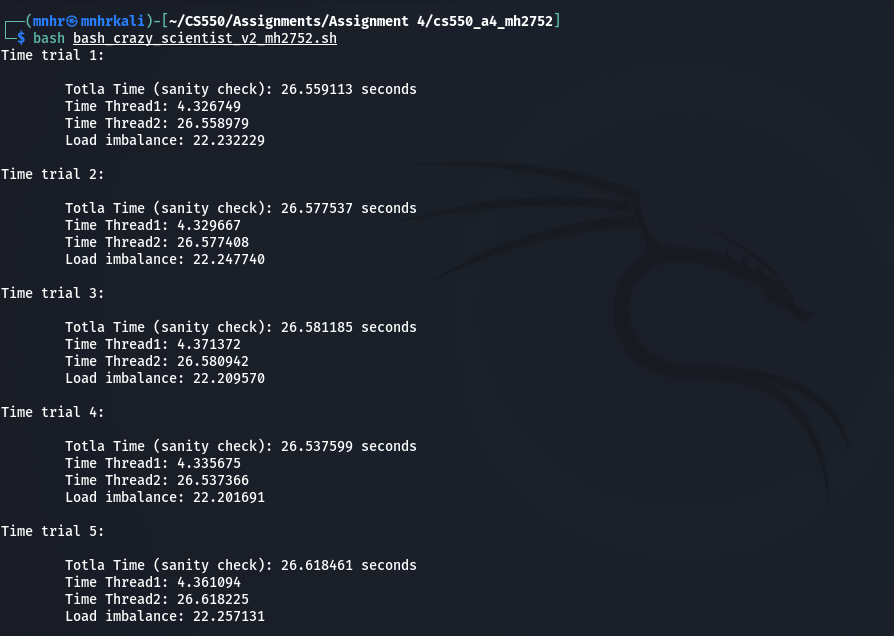
**Question-1:**



As demonstrated in the screenshot above, the overall execution time for the problem in Question-1 is:

**(**26.780407+26.761248+26.812868+26.876115+27.095316+26.787531+26.808233+26.769866+26.823451+26.794821**)**/10 = 268.309856/10 = **26.8309856 seconds**

**Question-2:**



Using the timestamps from the above screenshots,

***Average response time of Thread1 is:*****(**4.326749+4.329667+4.371372+4.335675+4.361094+4.352911+4.323033+4.358554+4.350170+4.353036**)/**10 = 43.462261/10 = **4.3462261 seconds**

***Average response time of Thread2 is:***

**(**26.558979+26.577408+26.580942+26.537366+26.618225+26.576715+26.516222+26.561246+26.559008+26.542804**)**/10 = 265.628915/10 = **26.5628915 seconds**

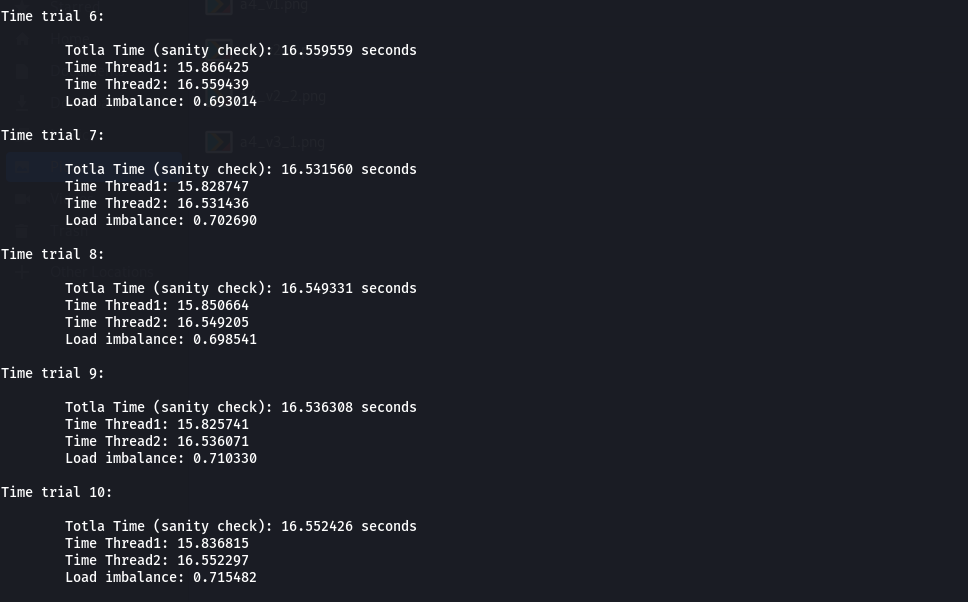
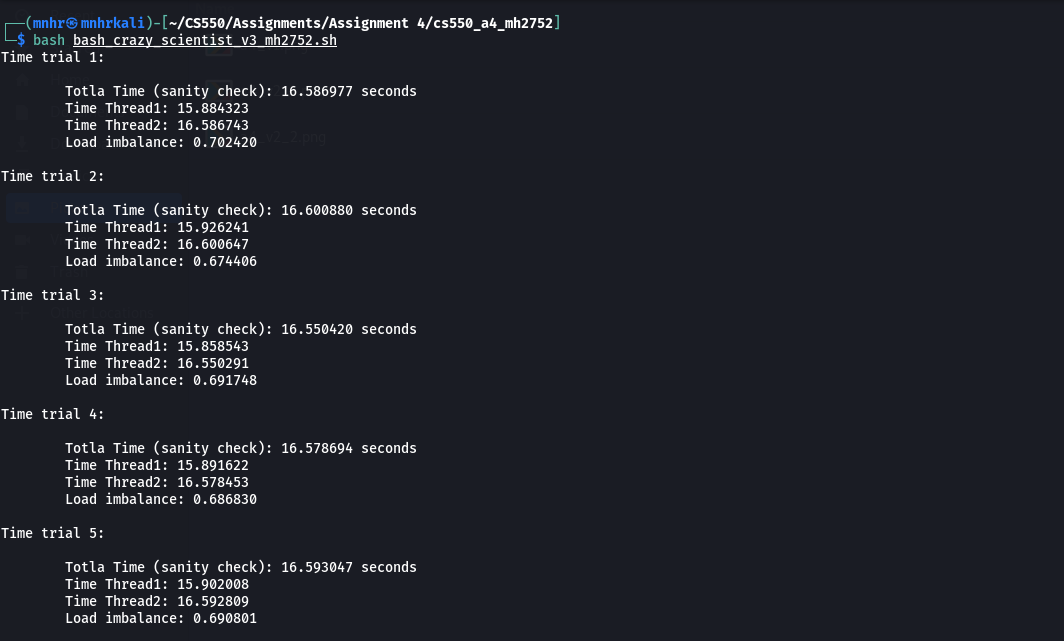
***Average load imbalance is:***

**(**22.232229+22.247740+22.209570+22.201691+22.257131+22.223803+22.193189+22.202691+22.208837+22.189767**)/**10 = 222.166648/10 = **22.2166648 seconds**

***Average of total response time is:***

**(**26.559113+26.577537+26.581185+26.537599+26.618461+26.576922+26.516350+26.561454+26.559255+26.543051**)**/10 = 265.630927/10 = **26.5630927 seconds**

**Question-3:**



Using the timestamps from the above screenshots,

***Average response time of Thread1 is:***

**(**15.884323+15.92641+15.858543+15.891622+15.902008+15.866425+15.828747+15.850664+15.825741+15.836815**)/**10 = 158.671298/10 = **15.8671298 seconds**

***Average response time of Thread2 is:***

**(**16.586743+16.600647+16.550291+16.578453+16.592809+16.559439+16.531436+16.549205+16.536071+16.552297**)**/10 = 165.637391/10 = **16.5637391 seconds**

***Average load imbalance is:***

**(**0.702420+0.674406+0.691748+0.686830+0.690801+0.693014+0.702690+0.698541+0.710330+0.715482**)/**10 = 6.966262/10 = **0.6966262 seconds**

***Average of total response time is:***

**(**16.586977+16.600880+16.550420+16.578694+16.593047+16.559559+16.531560+16.549331+16.536308+16.552426**)**/10 = 165.639202/10 = **16.5639202 seconds**

* Compared to Question-2, we can see a significant improvement in terms of load balance here in Question-3.

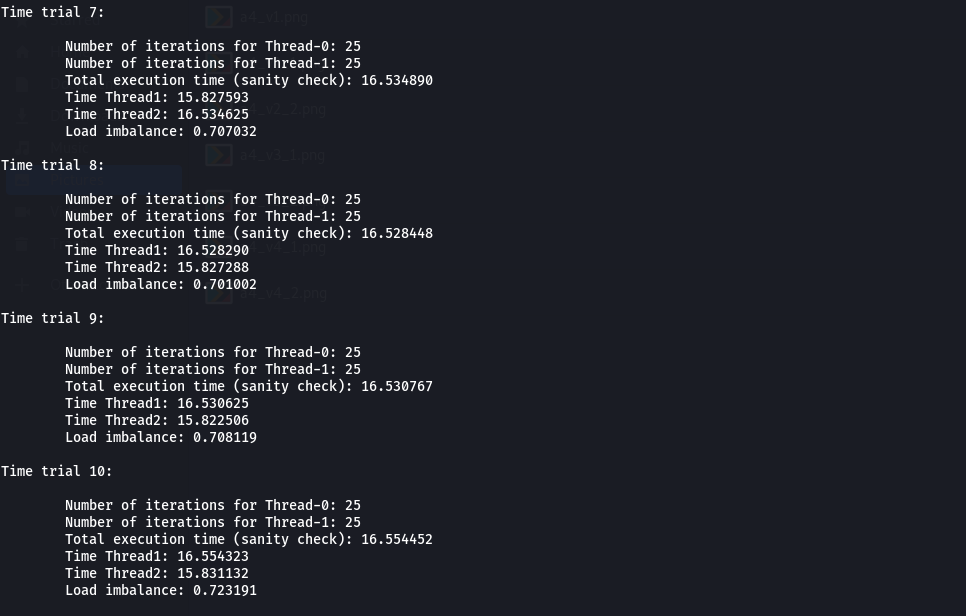
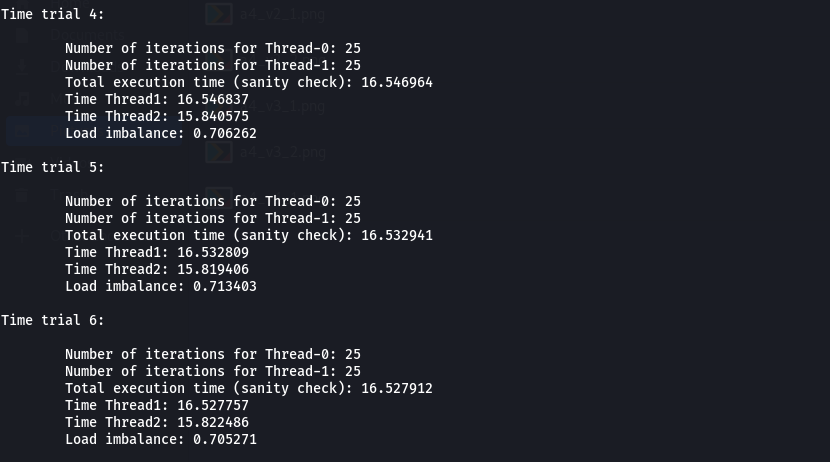
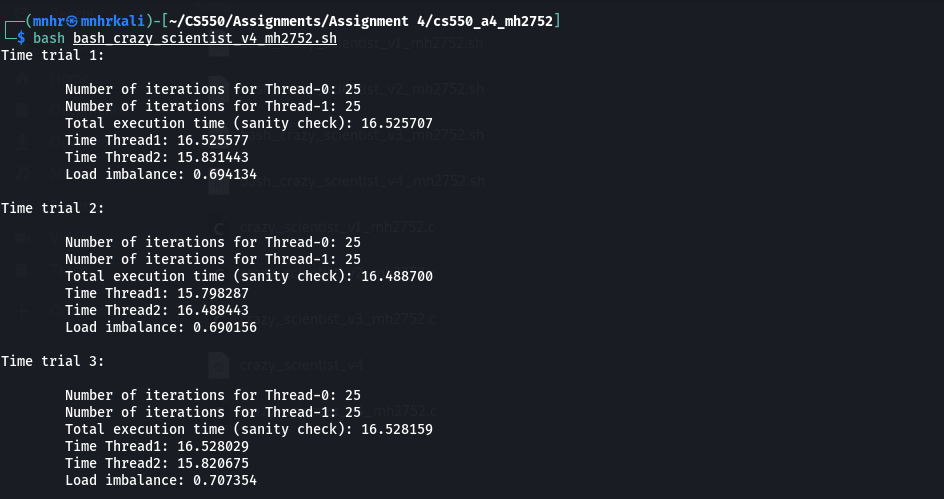
In the entire program, the most computationally heavy code portion is actually the for loop inside the do\_crazy\_calculation() function. Because of the **iter < 5\*x\*x\*x+1 + y\*y\*y+1** statement, the complexity of that loop is **O**(**N^3**) as x and y are variables.

In Question-2, Thread-1 always executed the function for lower values of x (as x gets the row index values assigned to Thread-1 that are in the range of 0 to 24) and Thread-2 always executed the function for higher values of x (as x gets the row index values assigned to Thread-2 that are in the range of 25 to 49).

As the complexity of the loop rose exponentially as **O**(**N^3**) with the increasing values of x, Thread-2 always had to perform the largest calculations. That is why we see large load imbalances in Question-2.

In Question-3, as the threads process the rows on an on demand basis, the workload gets quite evenly distributed among the threads. That is why we see very minute load imbalances compared to Question-2.

**Question-4:**



Using timestamps from the screenshots above,

***Average response time of Thread1 is:***

**(**16.525577+15.798287+16.528029+16.546837+16.532809+16.527757+15.827593+16.528290+16.530625+16.554323**)**/10 = 163.900127/10 = **16.3900127 seconds**

***Average response time of Thread2 is:***

**(**15.831443+16.488443+15.820675+15.840575+15.819406+15.822486+16.534625+15.827288+15.822506+15.831132**)** = 159.638579/10 = **15.9638579 seconds**

***Average load balance is:***

**(**0.694134+0.690156+0.707354+0.706262+0.713403+0.705271+0.707032+0.701002+0.708119+0.723191**)** = 7.055924/10 = **0.7055924 seconds**

***Average of total response time is:***

(16.525707+16.488700+16.528159+16.546964+16.532941+16.527912+16.534890+16.528448+16.530767+16.554452) = 165.29894/10 = **16.529894 seconds**

* Compared to Question-2, we can definitely see significant improvements in terms of load balancing here as the computationally exhaustive iterations are shared equally between two threads.

However, compared to Question-3, we observe slight but negligible differences in terms of thread response times as well as average load balance. We can attribute these differences to the use of a different library (pthread instead of openmp).