SIT315

PROGRAMMING PARADIGMS

TASK:M2\_S3P

NAME:TINA

ROLL NO:2210994851

TEAMMATES: SANCHIT(2210994836), SACHIN(2210994830), NISHANT(2210994821)

ACTIVITY1:

1. Modify the sequential program to use OpenMP to achieve parallelism using omp parallel and omp for directives.

Link to the modifies code using OpenMP:

<https://github.com/Tina1409/m2_s3p>

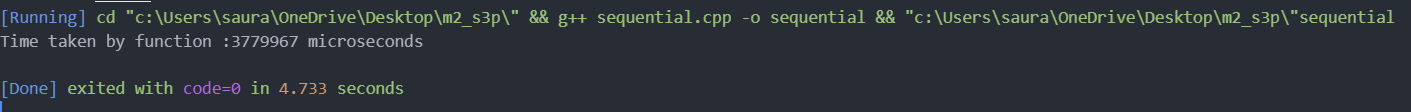
1. Evaluate the performance of the OpenMP implementation vs pthread implementation vs the sequential program. Discuss your findings.

***Answer:*** After the findings and proper evaluation of the code running I found that OpenMP runs faster than the rest. Though it can be noticed that PThread is also used for task parallelization but OpenMP is even faster than that because OpenMp use implicit threading, where the developer decided the parallel regions and it automatically manages the synchronization of threads which leads to less manual management and hence it takes lesser time. Also openMP supports SIMD , which allows the compiler to generate parallelize code.

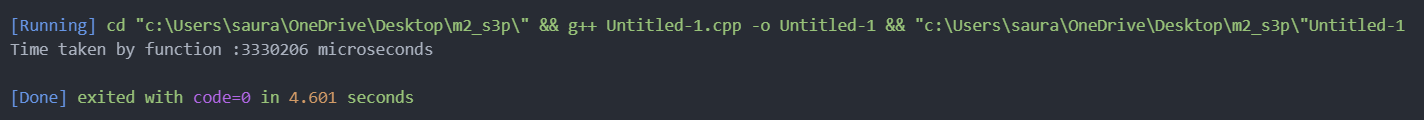
Whereas on the other hand sequential program is slower than both OpenMP and Pthread because in sequential each statement is executed one by one that is sequentially which results in one task at a time rather than in that of parallel where multiple tasks can run concurrently.

Below is the screenshot of the output time of sequential and OpenMP which is a proof that sequential takes more time than OpenMP.

Sequential:



OpenMP:



ACTIVITY2

1. Add the default(none)attribute to the #pargma omp parallel directive. Compile and run your code. If you get any compilation error, try to identify the reason. As required, add any of the shared, private or firstprivate attributes to fix the compilation error. Try different variations of data sharing (e.g. shared(size) private(v1) or private(size) shared(v1) or ...). Is the outcome.
2. of your program different? Explain why.
3. Compute the total sum of all the elements in v3 using a shared variable called total and atomic
4. update directive.
5. Use the reduction clause to compute the total sum of all the elements in v3.
6. Implement an alternative version where each thread computes its own part to a private variable and then use a critical section (#pragma omp critical) after the loop to calculate the total sum. Do you get the exact same results in all cases?
7. Try different OpenMP Scheduling techniques by adding schedule(type[,chunk]) attribute to #pargma omp for directive. Experiment with the chunk size to understand how each Scheduling
8. technique works. Does changing the scheduling techniques or chunk size impact the execution time. Briefly explain your observations.