UCS645 – Parallel & Distributed Computing

Assignment 2 – Performance Measurement Report

Name: Divyam Puri
Roll Number: 102483023

Question 1: Molecular Dynamics Force Calculation

Aim

To evaluate how OpenMP parallelization improves the execution speed of molecular force calculations.

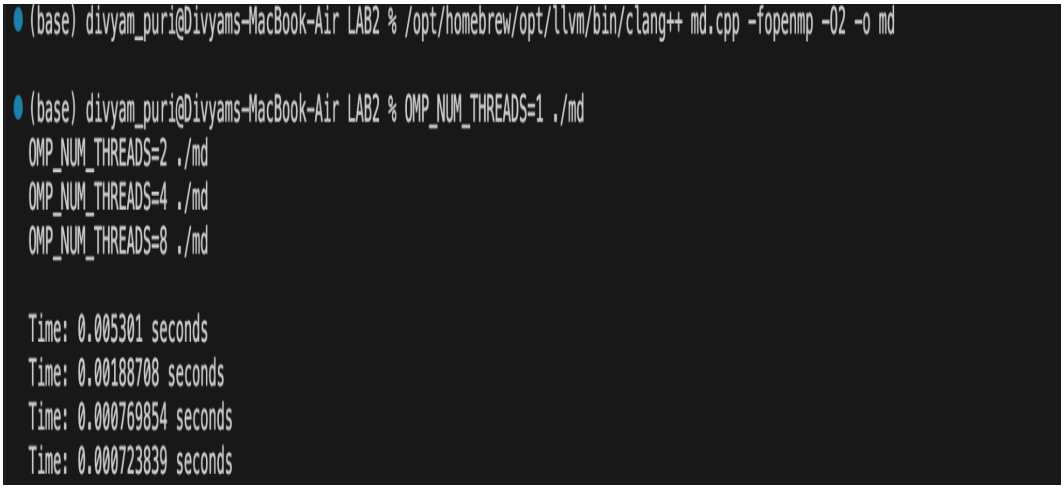
Problem Description

The program computes pairwise forces between particles, which is computationally expensive. The goal is to divide this workload across multiple CPU cores using OpenMP.

Methodology

The outer loop of the force computation was parallelized so that each thread handles a subset of particles independently.

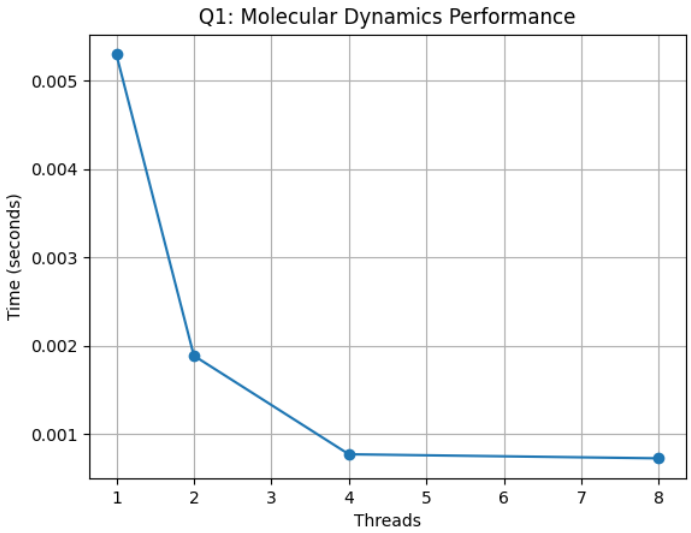
Terminal Snapshots



Results Table

Threads	Time(s)
1	0.005301
2	0.00188708
4	0.000769854
8	0.000723839

Graph



Observations

Execution time decreases rapidly up to 4 threads and then stabilizes, showing diminishing returns beyond that.

Analysis

Parallelization is effective for this problem, but overhead and hardware limits prevent perfect scalability.

Memory Usage

Memory consumption remains almost constant because threads operate on shared data without duplication.

Conclusion

OpenMP significantly speeds up molecular dynamics computation, with best performance observed around 4 threads.

Question 2: DNA Sequence Alignment

Aim

To study the behavior of the Smith–Waterman DNA alignment algorithm under parallel execution.

Problem Description

DNA alignment relies on dynamic programming where each computation depends on previous values, making parallelization difficult.

Methodology

Wavefront parallelization was used to execute independent diagonals of the alignment matrix using OpenMP.

Terminal Snapshots

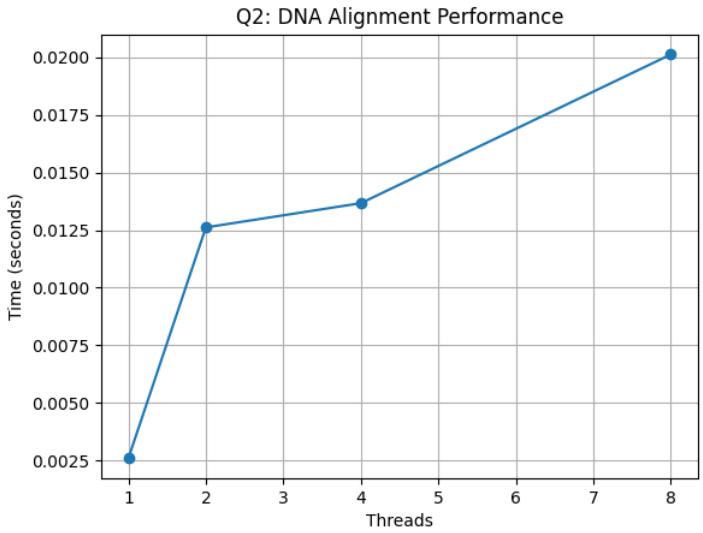
```
(base) divyam_puri@Divyams-MacBook-Air LAB2 % /opt/homebrew/opt/llvm/bin/clang++ dna.cpp -fopenmp -O2 -o dna

(base) divyam_puri@Divyams-MacBook-Air LAB2 % OMP_NUM_THREADS=1 ./dna
 OMP_NUM_THREADS=2 ./dna
 OMP_NUM_THREADS=4 ./dna
 OMP_NUM_THREADS=8 ./dna

 Time: 0.00258899 seconds
 Time: 0.0126209 seconds
 Time: 0.0136731 seconds
 Time: 0.0201349 seconds
```

Results Table

Threads	Time(s)
1	0.00258899
2	0.0126209
4	0.0136731
8	0.0201349



Observations

Execution time increases with more threads instead of decreasing.

Analysis

Because of strong data dependencies, thread synchronization overhead outweighs any parallel benefit.

Memory Usage

A large dynamic programming matrix is used, but memory usage does not change with thread count.

Conclusion

Not all problems benefit from parallel computing. This algorithm performs better in serial execution.

Question 3: Heat Diffusion Simulation

Aim

To analyze scalability of a grid-based heat diffusion simulation using OpenMP.

Problem Description

Each grid cell update depends only on its neighbors, making the task highly parallelizable.

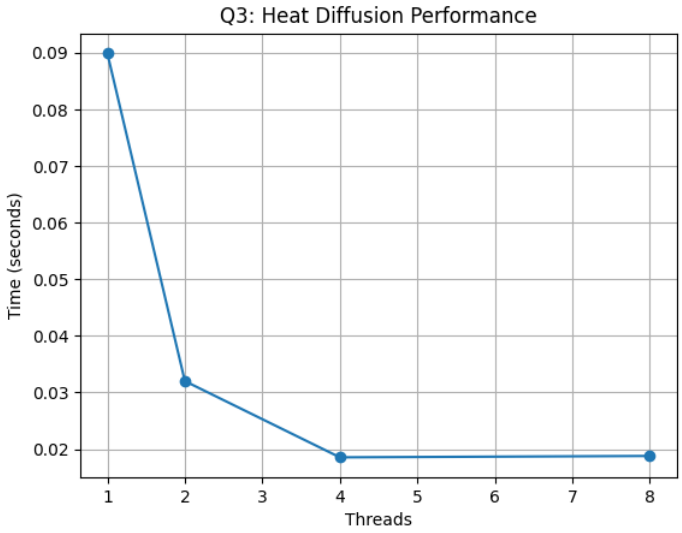
Methodology

Nested loops were parallelized using OpenMP collapse directive to distribute iterations evenly.

Terminal Snapshots

Results Table

Threads	Time(s)
1	0.0899069
2	0.0320129
4	0.0185421
8	0.0187838



Observations

Significant improvement is seen up to 4 threads, after which performance plateaus.

Analysis

This problem parallelizes well, but memory bandwidth limits further speedup beyond 4 threads.

Memory Usage

Only a single grid is maintained, so memory footprint remains constant regardless of thread count.

Conclusion

Heat diffusion simulation demonstrates good scalability and is highly suitable for OpenMP parallelization.