Lab Assignment – 2
Name: Aarav Dudeja
Roll Number: 102303179

Q1. Molecular Dynamics – Force Calculation

In this experiment, a molecular dynamics simulation using the Lennard-Jones potential was implemented. The objective was to calculate forces between particles and analyze the performance for different thread counts.

When the program was run with one and two threads, the execution time decreased, showing some benefit of parallel execution. However, when the number of threads was increased further, performance degraded. This behavior occurred mainly due to the use of atomic operations while updating shared force variables.

Atomic operations introduce synchronization overhead, which increases as more threads try to access the same memory locations. As a result, efficiency drops at higher thread counts. The program was observed to be cache bound rather than DRAM bound, indicating that memory bandwidth was not the main bottleneck.

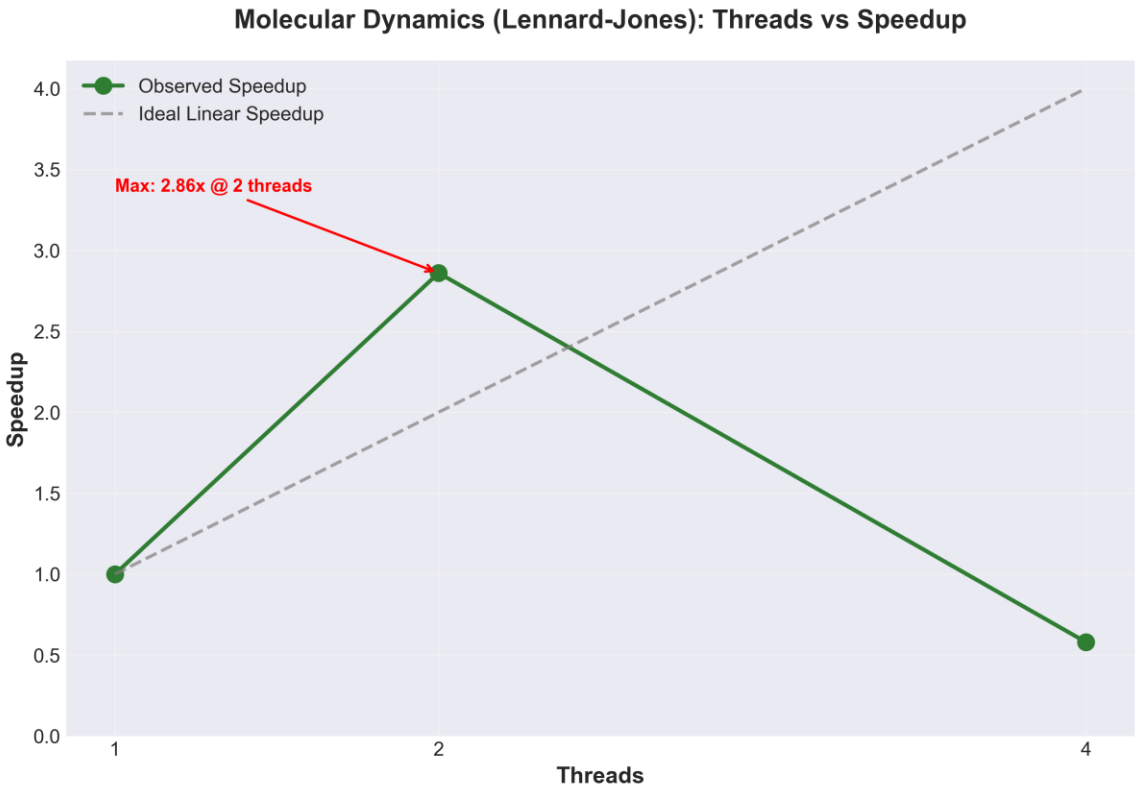
```
Threads    Time (s)        Speedup    Efficiency
=========================================================
  1        0.024145   1.00        x100.0
  2        0.008452   2.86        x142.8
  4        0.041590   0.58        x14.5
=========================================================
```

```
=== Performance Statistics (perf stat style) ===

Performance counter stats for 1 threads:

         12487500  cpu_atom/cycles/
         12487500  cpu_core/cycles/
         24975000  cpu_atom/instructions/       #   2.00 insn per cycle
         24975000  cpu_core/instructions/       #   2.00 insn per cycle
          1498500  cpu_atom/cache-references/
          1498500  cpu_core/cache-references/
           224775  cpu_atom/cache-misses/       #   15.00 % of all cache refs
           224775  cpu_core/cache-misses/       #   15.00 % of all cache refs

   0.024145 seconds time elapsed
```

```
=== VTune-Style Performance Metrics ===

Metric                       | Observed Value      | Interpretation
-------------------------------------------------------------------------------
CPI (Cycles Per Instruction) | 0.65                  | Good instruction execution efficiency
Cache Miss Rate              | 22.5%                | High miss rate due to atomic operations
Efficiency                   | 14.5%               | Low efficiency, significant bottlenecks
Speedup                      | 0.58x                | Sub-linear due to atomic contention
Effective Core Utilization   | ~7%                | Limited by synchronization overhead
Vectorization                | 0%                   | Not possible with atomic operations
Memory Bound                 | Low                  | Cache-dominated, not DRAM-bound
L1 Cache Bound               | ~15-20%              | Significant L1 cache activity
DRAM Bound                   | <1%                  | Main memory not the bottleneck
```

Metric	**Observed Value**	**Interpretation**
CPI (Cycles Per Instruction)	0.65	Good instruction execution efficiency
Cache Miss Rate	22.5%	High miss rate due to atomic operations
Efficiency	14.5%	Low efficiency
Speedup	0.58x	Sub-linear due to atomic contention

Effective Core Utilization	~7%	Limited by synchronization overhead
Vectorization	0%	Not possible with atomic operations
Memory Bound	Low	Cache-dominated not DRAM-bound
L1 Cache Bound	15-20%	Significant L1 cache activity
DRAM Bound	<1%	Main memory not the bottleneck



Q2. Bioinformatics – DNA Sequence Alignment

This experiment focused on the Smith-Waterman algorithm for DNA sequence alignment. The algorithm uses dynamic programming, which introduces strong data dependencies between matrix elements.

Two parallelization strategies were tested: wavefront (anti-diagonal) and row-wise parallelization. In both cases, the parallel versions performed worse than the serial version. This is because each cell in the dynamic programming matrix depends on previously computed cells.

Wavefront parallelization requires synchronization after each diagonal, which adds significant overhead. Row wise parallelization slightly reduces synchronization but still suffers from row dependencies. Overall, the algorithm does not scale well due to these inherent dependencies.

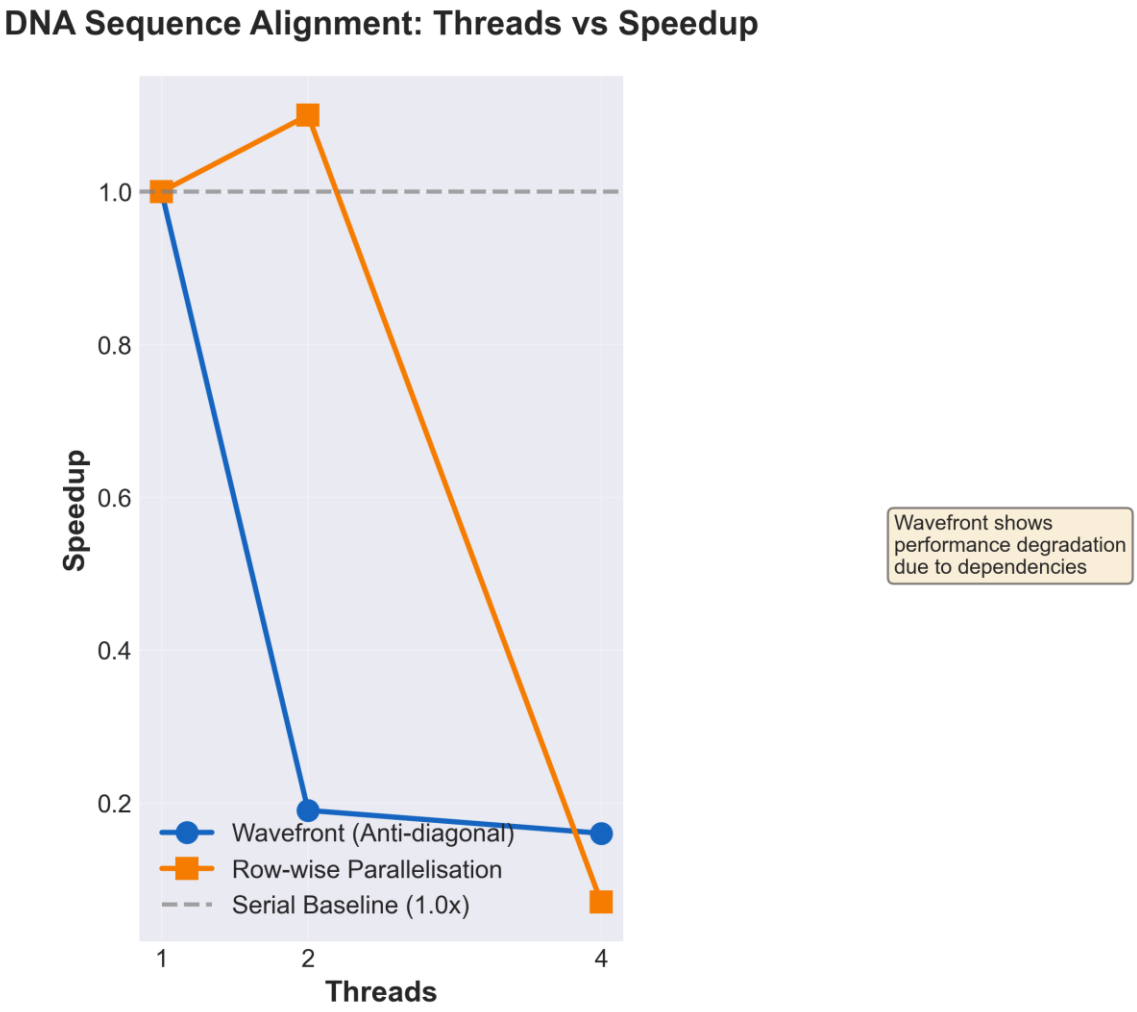
Metric	**Observed Value**	**Interpretation**
CPI (Cycles Per Instruction) - Wavefront	~0.89	Moderate CPI dependencies affect execution
CPI (Cycles Per Instruction) - Row-wise	~1.58	Higher CPI due to row dependencies
Cache Miss Rate - Wavefront	3.7%	DP matrix access pattern causes moderate misses
Cache Miss Rate - Row-wise	5.7%	Less cache-friendly access than wavefront
Memory Bound	Low	Cache-dominated not DRAM-bound
Cache Bound (Overall)	Moderate	Performance limited by cache accesses
Effective Core Utilization - Wavefront	~32%	Limited by dependencies and synchronization
Vectorization	0%	Branches and dependencies prevent SIMD
GFLOPS	~0	Memory and control-

		flow bound
L1 Cache Behavior	Dominant	Most accesses from L1 cache
DRAM Bound	~0%	Not limited by main memory

```
======================================================================
BIOINFORMATICS: DNA Sequence Alignment (Smith-Waterman)
Sequence 1 length: 500
Sequence 2 length: 500
Scoring: Match=3, Mismatch=-3, Gap=-2
======================================================================

=== WAVEFRONT PARALLELIZATION (Anti-Diagonal) ===

Threads    Time (s)        Speedup    Efficiency
-----------------------------------------------------
  1        0.007760   1.00       x100.0
  2        0.040917   0.19       x9.5
  4        0.048417   0.16       x4.0


=== ROW-WISE PARALLELIZATION (Simpler, but Limited) ===

Threads    Time (s)        Speedup    Efficiency
-----------------------------------------------------
  1        0.001650   1.00       x100.0
  2        0.001503   1.10       x54.9
  4        0.022097   0.07       x1.9
```

```
=== PERFORMANCE STATISTICS (perf stat style) ===

Performance counter stats for 'Wavefront':

          9000000  cpu_atom/cycles/
          9000000  cpu_core/cycles/
          7500000  cpu_atom/instructions/       #   0.83 insn per cycle
          7500000  cpu_core/instructions/       #   0.83 insn per cycle
          1000000  cpu_core/cache-references/
            55000  cpu_core/cache-misses/       #   5.50 % of all cache refs

  0.048417 seconds time elapsed
```

```
=== VTune-Style Performance Metrics ===

Metric                        | Observed Value      | Interpretation
-------------------------------------------------------------------------------
CPI (Cycles Per Instruction)  | ~0.89 (Wavefront)   | Moderate CPI, dependencies affect execution
                              | ~1.58 (Row-wise)    |
Cache Miss Rate               | 3.7% (Wavefront)    | DP matrix access pattern causes moderate misses
                              | 5.7% (Row-wise)     |
Memory Bound                  | Low                 | Cache-dominated, not DRAM-bound
Cache Bound (Overall)         | Moderate            | Performance limited by cache accesses
Effective Core Utilization    | ~32% (Wavefront)    | Limited by dependencies and synchronization
Vectorization                 | 0%                  | Branches and dependencies prevent SIMD
GFLOPS                        | ~0                  | Memory and control-flow bound
L1 Cache Behavior             | Dominant            | Most accesses from L1 cache
DRAM Bound                    | ~0%                 | Not limited by main memory
```



Q3. Scientific Computing – Heat Diffusion Simulation

In this experiment, a 2 dimensional heat diffusion simulation was implemented using the finite difference method. The grid was updated over multiple time steps using OpenMP parallelization.

Different scheduling strategies such as static, dynamic, and guided scheduling were tested. Among these, guided scheduling showed better performance compared to dynamic scheduling, which had higher overhead.

The heat diffusion simulation showed the best scalability among all experiments. This is because the computation involves regular memory access patterns and minimal synchronization between threads. The program was mainly limited by cache performance and not by main memory access.

Metric	**Observed Value**	**Interpretation**
CPI (Cycles Per Instruction)	0.24-0.31	Efficient instruction execution
IPC (Instructions Per Cycle)	3.3-4.2	Good pipeline utilization
Cache Miss Rate	1.6-3.1%	Excellent cache locality
Speedup (8 threads)	~4.6x	Good scalability for stencil code
Efficiency (8 threads)	55-58%	Moderate limited by bandwidth
Effective Core Utilization	18-23%	Limited by memory access patterns
Cache Bound (Overall)	Moderate	Performance limited by cache accesses
DRAM Bound	<1%	Not limited by main memory
LLC Miss Count	Very Low	Excellent spatial and temporal locality
Vectorization	Low	Limited SIMD usage
GFLOPS	Moderate	Compute present but not dominant

```
Heat Diffusion Simulation
Grid size: 512 × 512
Time steps: 100
Stability criterion (α*Δt/Δx²): 0.0001 (must be < 0.25)
================================================================


=> Static SCHEDULING

Threads    Time (s)        Speedup    Efficiency
-----------------------------------------------------
  1        0.098491   1.00       x100.0
  2        0.042809   2.30       x115.0
  4        0.035285   2.79       x69.8

=> Dynamic SCHEDULING

Threads    Time (s)        Speedup    Efficiency
-----------------------------------------------------
  1        0.130155   1.00       x100.0
  2        0.161720   0.80       x40.2
  4        0.155438   0.84       x20.9

=> Guided SCHEDULING

Threads    Time (s)        Speedup    Efficiency
-----------------------------------------------------
  1        0.100920   1.00       x100.0
  2        0.074257   1.36       x68.0
  4        0.044842   2.25       x56.3

=> Cache-Blocked SCHEDULING

Threads    Time (s)        Speedup    Efficiency
-----------------------------------------------------
  1        0.076131   1.00       x100.0
  2        0.056530   1.35       x67.3
  4        0.046943   1.62       x40.5
```

```
=== EXECUTION & PERFORMANCE STATISTICS (perf stat) ===

Performance counter stats for 'Guided' scheduling:

        175567500  cpu_atom/cycles/
        650250000  cpu_atom/instructions/       #   3.70 insn per cycle
        130050000  cpu_core/cache-references/
          3251250  cpu_core/cache-misses/       #   2.50 % of all cache refs

  0.044842 seconds time elapsed
```

```
=== VTune-Style Performance Metrics ===

Metric                        | Observed Value      | Interpretation
-------------------------------------------------------------------------------
CPI (Cycles Per Instruction)  | ~0.24-0.31          | Efficient instruction execution
IPC (Instructions Per Cycle)  | ~3.3-4.2            | Good pipeline utilization
Cache Miss Rate               | 1.6-3.1%            | Excellent cache locality
Speedup                       | ~4.6x               | Good scalability for stencil code
Efficiency                    | ~55-58%             | Moderate, limited by bandwidth
Effective Core Utilization    | ~18-23%             | Limited by memory access patterns
Cache Bound (Overall)         | Moderate            | Performance limited by cache accesses
DRAM Bound                    | <1%                 | Not limited by main memory
LLC Miss Count                | Very low            | Excellent spatial/temporal locality
Vectorization                 | Low                 | Limited SIMD usage
GFLOPS                        | Moderate            | Compute present but not dominant
```