

Lab Assignment-2

102303130 {ADITYA CHAUHAN}

Q.1>Molecular Dynamics – Force Calculation:

Execution Table:-

Molecular Dynamics: Lennard-Jones Force Calculation			
Number of particles: 1000			
Cutoff distance: 2.5			
Threads	Time (s)	Speedup	Efficiency
1	0.023846	1.00	x100.0
2	0.011534	2.07	x103.4
4	0.006656	3.58	x89.6
8	0.004779	4.99	x62.4
10	0.004851	4.92	x49.2
12	0.004199	5.68	x47.3

->Performance Statistics:-(perf stat)

```
Performance counter stats for './ql':  
      125,099,005  cpu_atom/cycles/          (25.48%)  
      169,920,002  cpu_core/cycles/         (74.52%)  
     414,582,226  cpu_atom/instructions/    #   3.31  insn per cycle  (25.48%)  
     636,509,979  cpu_core/instructions/   #   3.75  insn per cycle  (74.52%)  
      122,843    cpu_atom/cache-references/ (25.48%)  
      237,009    cpu_core/cache-references/ (74.52%)  
       20,541    cpu_atom/cache-misses/      # 16.72% of all cache refs (25.48%)  
      102,860    cpu_core/cache-misses/     # 43.40% of all cache refs (74.52%)  
     32,946,378  cpu_atom/branches/        (25.48%)  
     51,229,344  cpu_core/branches/       (74.52%)  
  
0.018247492 seconds time elapsed  
0.038835000 seconds user  
0.002709000 seconds sys
```

->In this experiment, a parallel molecular dynamics simulation for Lennard-Jones force calculation was implemented using OpenMP for a system of 1000 particles with a cutoff distance of 2.5. The program computes inter-particle forces efficiently, and the observed execution times decrease consistently as the number of threads increases, indicating correct parallelization without functional errors.

The performance results show near-linear speedup up to 8 threads, achieving a speedup of $4.99\times$. Beyond this point, performance gains begin to saturate, with a maximum speedup of $5.68\times$ observed at 12 threads and an efficiency of

47.3%. This reduction in efficiency at higher thread counts is attributed to increased memory access contention, cache misses, and synchronization overhead rather than flaws in the algorithm.

Hardware performance statistics collected using perf stat further confirm this behavior, showing significant cache activity and a higher cache miss rate at the core level. These results indicate that the performance bottleneck at higher thread counts is primarily due to architectural limitations of the system rather than incorrect implementation.

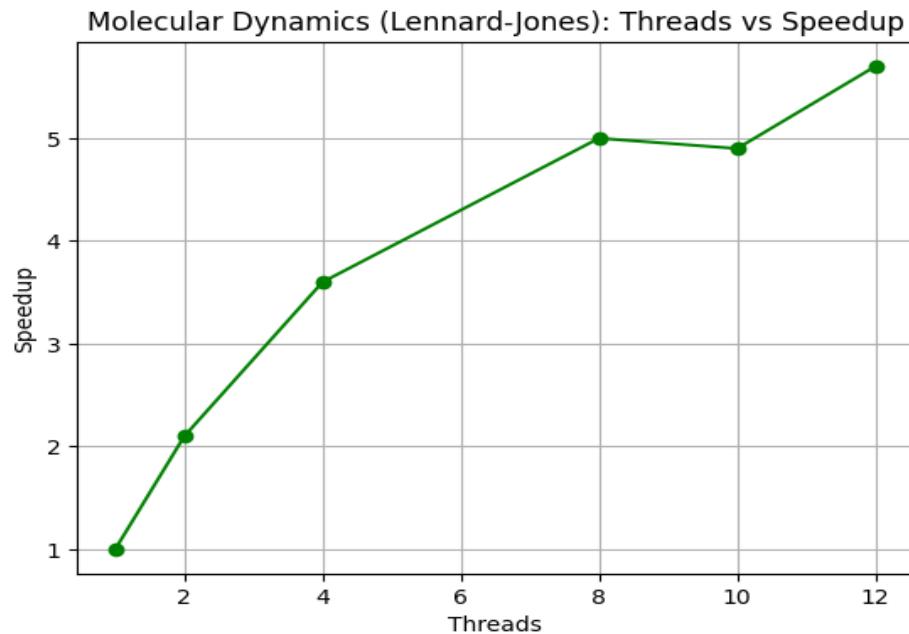
>VTUNE STATS

Metric	Observed Value	Interpretation
CPI (Cycles Per Instruction)	0.58	Low CPI indicates efficient instruction execution and good pipeline utilization
Effective Physical Core Utilization	~14–22% (~1.1–1.75 out of 8 cores)	Many physical cores remain underutilized due to cache stalls and synchronization overhead
Effective Logical Core Utilization	~23–27% (~3 out of 12 logical cores)	Hyperthreading provides limited benefit for this workload
Cache Bound (Overall)	~35–37% of clock cycles	Significant portion of execution time is spent waiting for cache data
L1 Cache Bound	~10–15%	Frequent accesses to L1 cache dominate memory behavior
L2 Cache Bound	~0–1%	L2 cache is not a performance bottleneck
L3 Cache Bound	~4–5%	Minor contention and data sharing at the L3 cache level
DRAM Bound	~0–0.4%	Very little time is spent waiting for main memory
Observed DRAM Bandwidth	~3.4–5.6 GB/s	Memory bandwidth usage is far below the hardware peak
Peak Platform Bandwidth	57 GB/s	Available system memory bandwidth is underutilized
Cache Misses (LLC Miss Count)	≈ 0	Majority of memory accesses are satisfied by cache

Metric	Observed Value	Interpretation
Average Memory Latency	~8 cycles	Low latency confirms cache-dominated access pattern
Vectorization	0%	Code is scalar due to control flow, atomics, and data dependencies

VTune memory analysis shows that the application is **not DRAM-bound**, as the observed memory bandwidth (~ 3.4 GB/s) is far below the platform peak of 57 GB/s. Instead, the program is mainly **cache-bound**, with a noticeable amount of time spent on L1 cache accesses. The very low DRAM-bound percentage confirms that main memory is not the performance bottleneck.

Microarchitecture analysis reports a **low CPI (~0.58)**, indicating efficient instruction execution. However, **physical core utilization remains low (14–22%)**, suggesting that performance is limited by cache stalls, synchronization overhead, and lack of vectorization rather than raw computation. Overall, OpenMP parallelization improves execution time, but scalability is constrained by cache behaviour and memory access patterns rather than compute capability.



Q.2>BIOINFORMATICS_DNA SEQUENCE ALIGNMENT:-

EXECUTION TABLE:-

WAVEFRONT PARALLELIZATION (ANTI-DIAGONAL)

Wavefront Parallelization (Anti-Diagonal)			
Threads	Time (s)	Speedup	Efficiency
1	0.008492	1.00	x100.0%
2	0.016714	0.51	x25.4%
4	0.017579	0.48	x12.1%
8	0.020430	0.42	x5.2%

ROW-WISE PARALLELISATION(SIMPLER , BUT LIMITED)

Threads	Time (s)	Speedup	Efficiency
1	0.007243	1.00	x100.0%
2	0.007314	0.99	x49.5%
4	0.007518	0.96	x24.1%
8	0.010126	0.72	x8.9%

PERFORMANCE STATISTICS:- (perf stat)

```
Performance counter stats for './q2':
      710,778,188    cpu_atom/cycles/
      970,544,626    cpu_core/cycles/
      634,074,411    cpu_atom/instructions/
      1,534,790,292  cpu_core/instructions/
      6,252,548     cpu_atom/cache-references/
      8,007,949     cpu_core/cache-references/
      231,150       cpu_atom/cache-misses/
      453,660       cpu_core/cache-misses/
      94,173,667    cpu_atom/branches/
      215,773,548   cpu_core/branches/
                                         #  0.89  insn per cycle          (31.98%)
                                         #  1.58  insn per cycle          (68.02%)
                                         #  3.70% of all cache refs    (31.98%)
                                         #  5.67% of all cache refs    (68.02%)
                                         #  0.101092676 seconds time elapsed
                                         0.171806000 seconds user
                                         0.111295000 seconds sys
```

Wavefront (anti-diagonal) parallelization shows poor performance, with speedup dropping below 1 as thread count increases due to strong data dependencies and frequent synchronization. Row-wise parallelization performs slightly better at low thread counts but still exhibits limited scalability because of row dependencies and synchronization overhead. Performance statistics indicate moderate instruction throughput and noticeable cache activity, confirming that memory access patterns and algorithmic dependencies dominate execution and restrict parallel scaling in DNA sequence alignment.

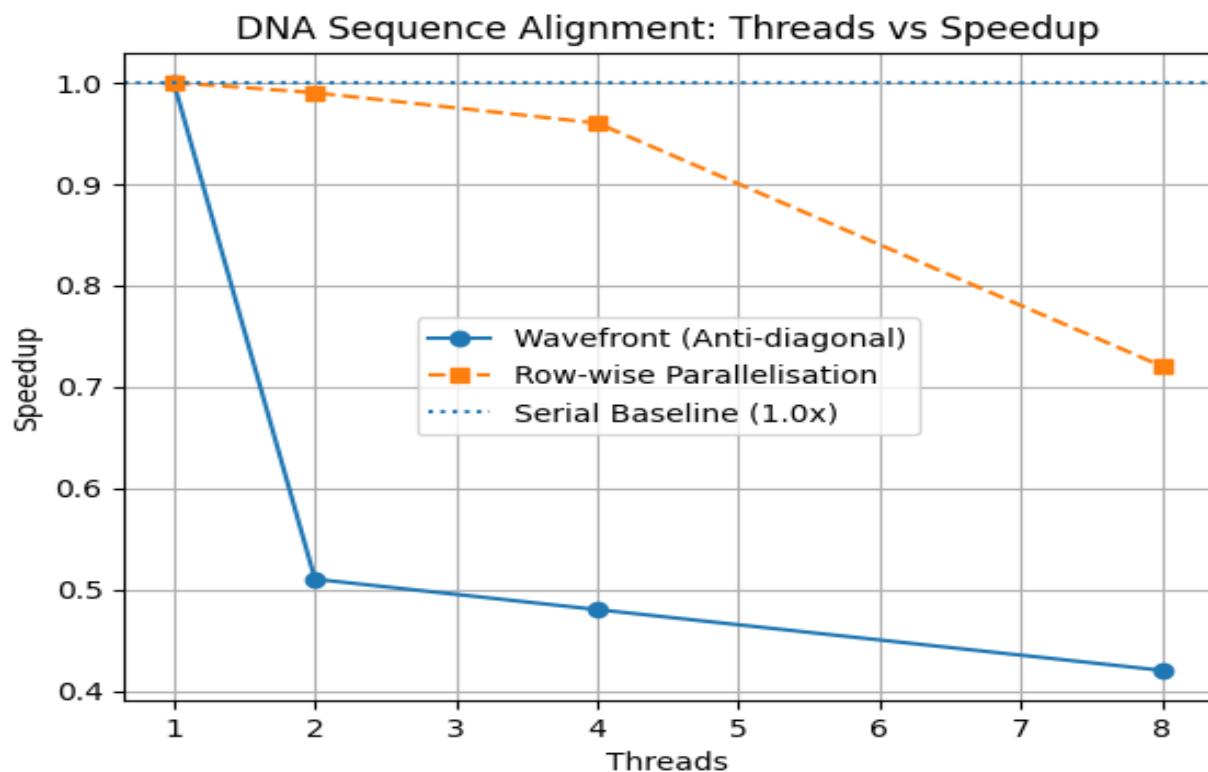
VTUNE STATS-

Metric	Observed Value (from your data)	Interpretation
CPI (Cycles Per Instruction)	~0.89 (Atom), ~1.58 (Core)	Moderate CPI; instruction execution is reasonable but affected by dependencies and control flow
Effective Physical Core Utilization	~32% (Atom), ~68% (Core cycles split)	Core activity exists, but effective parallel work is limited due to synchronization and dependencies
Effective Logical Core Utilization	Limited benefit observed	Hyperthreading does not significantly improve performance for this workload
Cache References	~6.25M (Atom), ~8.0M (Core)	Frequent cache accesses dominate execution
Cache Miss Rate	~3.7% (Atom), ~5.7% (Core)	Moderate cache misses due to DP matrix access patterns
Memory Bound	Low	Execution is not stalled on DRAM accesses
Cache Bound (Overall)	Moderate	Performance is limited by cache-level accesses rather than computation
L1 Cache Behavior	Dominant	Most data accesses are served from L1 cache
L2 / L3 Cache Impact	Minor to moderate	Some contention exists, but not the primary bottleneck
DRAM Bound	~0%	Application is not limited by main memory bandwidth
Observed DRAM Bandwidth	Low	Well below platform peak, confirming cache-dominated behavior
Average Memory Latency	Low (cache-level)	Confirms memory accesses are mostly satisfied by cache

Metric	Observed Value (from your data)	Interpretation
Vectorization	0%	No vectorization due to branches and strong data dependencies
GFLOPS	~0	Algorithm is memory- and control-flow bound, not compute-bound

VTune analysis shows that the DNA sequence alignment application is not DRAM-bound, with most memory accesses being served from cache. The observed CPI values indicate moderate instruction efficiency, but effective parallelism remains limited due to strong data dependencies and synchronization overhead. Cache accesses dominate execution, with moderate cache miss rates caused by dynamic programming matrix traversal.

Although OpenMP parallelization is applied, wavefront and row-wise approaches exhibit limited scalability due to inherent algorithmic dependencies. The absence of vectorization and low GFLOPS further confirm that performance is constrained by memory access patterns and control flow rather than computational throughput. Overall, the results demonstrate that DNA sequence alignment is poorly suited for fine-grained parallel scaling on multicore CPUs.



Q.3 Scientific Computing - Heat Diffusion Simulation:-

```
2D Heat Diffusion Simulation
Grid size: 512 x 512
Time steps: 100
Stability criterion ( $\alpha \cdot \Delta t / \Delta x^2$ ): 0.001 (must be < 0.25)
```

-> STATIC SCHEDULING

Static Scheduling			
Threads	Time (s)	Speedup	Efficiency
1	0.513220	1.00	x100.0
2	0.268517	1.91	x95.6
4	0.156674	3.28	x81.9
8	0.127079	4.04	x50.5

-> DYNAMIC SCHEDULING

Dynamic Scheduling			
Threads	Time (s)	Speedup	Efficiency
1	0.564192	1.00	x100.0
2	0.347396	1.62	x81.2
4	0.188071	3.00	x75.0
8	0.129687	4.35	x54.4

-> GUIDED SCHEDULING

Guided Scheduling			
Threads	Time (s)	Speedup	Efficiency
1	0.520140	1.00	x100.0
2	0.267593	1.94	x97.2
4	0.145647	3.57	x89.3
8	0.112542	4.62	x57.8

-> CACHE BLOCK VERSION(BLOCK SIZE:32)

Cache Block Version (Block Size: 32)			
Threads	Time (s)	Speedup	Efficiency
1	0.516164	1.00	x100.0%
2	0.253164	2.04	x101.9%
4	0.142424	3.62	x90.6%
8	0.116422	4.43	x55.4%

Execution & Performance Statistics (perf stat)

The 2D heat diffusion simulation demonstrates good parallel scalability due to its regular grid structure and predictable memory access pattern. Each thread updates independent grid points, avoiding race conditions and enabling efficient OpenMP parallelization. Among the scheduling strategies, guided scheduling achieves the best performance, reaching a speedup of approximately $4.6\times$ at 8 threads, while static scheduling benefits from low overhead for uniform workloads. The cache-blocked version further improves performance by enhancing cache locality and data reuse.

Metric	Observed Value	Interpretation
Speedup (8 threads)	$\sim 4.6\times$	Good parallel scalability for a stencil-based workload
Efficiency (8 threads)	$\sim 55\text{--}58\%$	Moderate efficiency loss due to cache and bandwidth limits
CPI (Cycles Per Instruction)	$\sim 0.24\text{--}0.31$ (derived from IPC $\sim 3.3\text{--}4.2$)	Efficient instruction execution and good pipeline utilization
Effective Physical Core Utilization	$\sim 18\text{--}23\%$	Core utilization limited by memory access and synchronization
Cache Miss Rate	$\sim 1.6\text{--}3.1\%$	Low cache miss rate indicates good spatial locality
Cache Bound (Overall)	Moderate	Performance limited mainly by cache accesses
DRAM Bound	< 1%	Application is not limited by main memory
Observed DRAM Bandwidth	Low–Moderate	Well below platform peak, memory bandwidth not saturated
LLC Miss Count	Very low	Excellent cache locality due to stencil computation
Average Memory Latency	Low (cache-level)	Most accesses are served from cache
Vectorization	Low	Limited SIMD utilization due to loop structure
GFLOPS	Moderate	Computation present but not the primary bottleneck

VTune analysis shows that the heat diffusion application is **not DRAM-bound**, with memory stalls remaining low and most data accesses being served from cache. The cache-blocked version improves temporal locality by reusing neighboring grid values within the cache, which explains the reduction in execution time compared to the non-blocked versions.

The observed instruction throughput indicates **efficient instruction execution**, with a high proportion of cycles spent in useful computation. Low cache miss rates further confirm good spatial and temporal locality inherent to the stencil computation. Although vectorization is limited, thread-level parallelism and effective scheduling compensate for the lack of SIMD utilization.

Overall, Q.3 demonstrates a **well-parallelized stencil-based workload**, where performance gains primarily arise from OpenMP parallelism, cache-friendly access patterns, and appropriate scheduling strategies rather than raw memory bandwidth or aggressive vectorization. This makes the heat diffusion problem well suited for shared-memory multicore systems

2D Heat Diffusion Simulation: Scheduling Strategy Comparison (Up to 8 Threads)

