Assignment 2

Saksham Mittal-102303135

Ques 1)

```
Lennard-Jones Force Simulation
Particles: 1000

Thr       Time(s)       Speedup       Eff(%)        Energy
---------------------------------------------------------------
1         0.014827      1.00          100.0         %521104.860
2         0.007443      1.99          99.6          %521104.860
4         0.006989      2.12          53.0          %521104.860
8         0.004519      3.28          41.0          %521104.860
12        0.007761      1.91          15.9          %521104.860
16        0.009576      1.55          9.7           %521104.860
---------------------------------------------------------------

 Performance counter stats for './ques1':

     25,66,53,584      task-clock                       #    4.851 CPUs utilized
              158      context-switches                 #  615.616 /sec
               35      cpu-migrations                   #  136.371 /sec
              190      page-faults                      #  740.297 /sec
     14,49,40,383      cpu_atom/instructions/           #    0.33  insn per cycle              (43.45%)
     29,66,46,079      cpu_core/instructions/           #    0.53  insn per cycle              (49.36%)
     43,53,75,365      cpu_atom/cycles/                 #    1.696 GHz                         (43.83%)
     56,19,92,886      cpu_core/cycles/                 #    2.190 GHz                         (49.36%)
      1,49,03,899      cpu_atom/branches/               #   58.070 M/sec                       (43.31%)
      3,07,59,711      cpu_core/branches/               #  119.849 M/sec                       (49.36%)
           54,208      cpu_atom/branch-misses/          #    0.36% of all branches             (44.53%)
           98,911      cpu_core/branch-misses/          #    0.32% of all branches             (49.36%)
 #     73.0 %  tma_backend_bound
                                                  #      3.5 %  tma_bad_speculation
                                                  #      6.2 %  tma_frontend_bound
                                                  #     17.2 %  tma_retiring             (49.36%)
 #     14.9 %  tma_bad_speculation
                                                  #      8.2 %  tma_retiring             (44.67%)
 #     70.5 %  tma_backend_bound
                                                  #      6.3 %  tma_frontend_bound       (44.69%)

       0.052909636 seconds time elapsed

       0.252449000 seconds user
       0.006058000 seconds sys
```

Observations

- The potential energy value remained the same for all thread counts, which shows that the parallel force and energy calculations are correct and free from race conditions.
- Execution time decreased as the number of threads increased up to 8 threads, after which the time started increasing again.
- The best speedup was achieved around 8 threads, while adding more threads reduced performance.
- Efficiency dropped as thread count increased, which is expected due to thread management overhead and memory bandwidth limits.

- perf statistics indicate the program is largely **memory-bound**, with moderate IPC and high backend stall percentage.

Conclusion

The OpenMP parallelization is working correctly and produces consistent physical results. Performance improves only up to the number of effective CPU cores, after which overhead and hardware limits reduce gains. The experiment demonstrates that parallel programs do not scale linearly and that optimal thread count depends on system architecture and memory behavior.

Ques 2)

```
saksham@saksham:~/Desktop/parallel-assignments/Lab2$ sudo perf stat ./ques2
[sudo] password for saksham:
Parallel Smith-Waterman Test
Length = 500


[Wavefront]
Thr       Time           Speedup       Eff(%)
-----------------------------------------------
1         0.001978       1.00          100.0
2         0.020432       0.10          4.8
4         0.020527       0.10          2.4
8         0.027291       0.07          0.9
12        0.042148       0.05          0.4
16        0.052692       0.04          0.2

[Row Method]
Thr       Time           Speedup       Eff(%)
-----------------------------------------------
1         0.000451       1.00          100.0
2         0.008497       0.05          2.7
4         0.010883       0.04          1.0
8         0.011467       0.04          0.5
12        0.016203       0.03          0.2
16        0.032184       0.01          0.1

 Performance counter stats for './ques2':

    1,15,22,00,067      task-clock                       #    4.597 CPUs utilized
          1,02,911      context-switches                 #   89.317 K/sec
             2,798      cpu-migrations                   #    2.428 K/sec
               678      page-faults                      #  588.439 /sec
    1,04,28,59,632      cpu_atom/instructions/           #    1.03  insn per cycle              (50.56%)
    1,19,63,31,931      cpu_core/instructions/           #    1.08  insn per cycle              (42.53%)
    1,01,33,88,952      cpu_atom/cycles/                 #    0.880 GHz                         (50.43%)
    1,10,80,37,805      cpu_core/cycles/                 #    0.962 GHz                         (42.53%)
      17,48,98,887      cpu_atom/branches/               #  151.796 M/sec                       (50.00%)
      20,19,64,668      cpu_core/branches/               #  175.286 M/sec                       (42.53%)
         20,80,637      cpu_atom/branch-misses/          #    1.19% of all branches             (50.16%)
         13,16,290      cpu_core/branch-misses/          #    0.65% of all branches             (42.53%)
 #     37.6 %  tma_backend_bound
                                                  #      6.2 %  tma_bad_speculation
                                                  #     27.7 %  tma_frontend_bound
                                                  #     28.5 %  tma_retiring             (42.53%)
 #     10.5 %  tma_bad_speculation
                                                  #     24.5 %  tma_retiring             (50.00%)
 #     34.6 %  tma_backend_bound
                                                  #     30.4 %  tma_frontend_bound       (50.25%)

       0.250657386 seconds time elapsed

       0.148270000 seconds user
       1.095613000 seconds sys
```

Observations

- Both the Wavefront and Row-based methods produced valid execution results, but neither showed performance improvement with higher thread counts.
- Execution time increased as threads increased, leading to speedup values less than 1 and very low efficiency.
- The Wavefront method performed slower than the row method overall because of higher synchronization and scheduling overhead on each diagonal.
- The Row method had slightly better base time but still degraded with more threads due to data dependencies and thread management overhead.
- perf statistics indicate moderate IPC (~1.0) but noticeable frontend and backend stalls, showing that the workload is not compute-heavy enough to benefit from many threads.
- High system time compared to user time suggests additional OS and scheduling overhead when many threads are used.

Conclusion

The Smith Waterman implementation is functionally correct, but the problem size is too small and the dynamic programming dependencies limit effective parallelism. Increasing the number of threads introduces more overhead than benefit, resulting in negative scaling. The experiment demonstrates that algorithms with strong data dependencies and small workloads may not achieve speedup with naive parallelization, and that optimal performance depends on both algorithm structure and input size rather than simply increasing thread count.

Ques 3)

```
2D Heat Diffusion Simulation
Grid: 512x512
Steps: 100

[static scheduling]
Thr       Time          Speedup       Eff%
-----------------------------------------------
1         0.05768       1.00          100.0%
2         0.04409       1.31          65.4%
4         0.04303       1.34          33.5%
8         0.02449       2.36          29.4%
12        0.01651       3.49          29.1%
16        0.02485       2.32          14.5%

[dynamic scheduling]
Thr       Time          Speedup       Eff%
-----------------------------------------------
1         0.07747       1.00          100.0%
2         0.06025       1.29          64.3%
4         0.04666       1.66          41.5%
8         0.04315       1.80          22.4%
12        0.04079       1.90          15.8%
16        0.03337       2.32          14.5%

[guided scheduling]
Thr       Time          Speedup       Eff%
-----------------------------------------------
1         0.05714       1.00          100.0%
2         0.02996       1.91          95.4%
4         0.03214       1.78          44.5%
8         0.02260       2.53          31.6%
12        0.02697       2.12          17.7%
16        0.02285       2.50          15.6%

[Cache Blocking]
Thr       Time          Speedup       Eff%
-----------------------------------------------
1         0.02921       1.00          100.0%
2         0.01583       1.85          92.3%
4         0.02333       1.25          31.3%
8         0.02412       1.21          15.1%
12        0.01722       1.70          14.1%
16        0.01535       1.90          11.9%
```

```
Performance counter stats for './ques3':

   3,67,38,92,794      task-clock                       #    4.210 CPUs utilized
          26,449      context-switches                 #    7.199 K/sec
           1,181      cpu-migrations                   #  321.457 /sec
          14,317      page-faults                      #    3.897 K/sec
   8,91,81,63,854      cpu_atom/instructions/           #    1.01  insn per cycle              (45.64%)
  25,12,29,40,015      cpu_core/instructions/           #    1.98  insn per cycle              (47.30%)
   8,83,18,72,640      cpu_atom/cycles/                 #    2.404 GHz                         (45.92%)
  12,69,12,73,022      cpu_core/cycles/                 #    3.454 GHz                         (47.30%)
      63,84,22,666      cpu_atom/branches/               #  173.773 M/sec                       (46.04%)
   1,76,84,51,314      cpu_core/branches/               #  481.356 M/sec                       (47.30%)
       16,54,669      cpu_atom/branch-misses/          #    0.26% of all branches             (46.48%)
       97,49,278      cpu_core/branch-misses/          #    0.55% of all branches             (47.30%)
#     54.8 %  tma_backend_bound
                                                  #      1.7 %  tma_bad_speculation
                                                  #      3.7 %  tma_frontend_bound
                                                  #     39.9 %  tma_retiring             (47.30%)
#      1.2 %  tma_bad_speculation
                                                  #     20.4 %  tma_retiring             (46.66%)
#     76.4 %  tma_backend_bound
                                                  #      2.0 %  tma_frontend_bound       (45.98%)

      0.872691172 seconds time elapsed

      3.456999000 seconds user
      0.237866000 seconds sys
```

Observations

- All three scheduling strategies (static, dynamic, guided) produced correct and stable temperature values, indicating that the parallel heat update has no race conditions.
- Execution time decreased as thread count increased up to around 12 threads, after which improvement slowed or slightly reversed.
- Guided scheduling generally showed the best balance between speedup and efficiency, especially at mid-range thread counts.
- Static scheduling performed well for moderate threads but lost efficiency at higher counts due to workload imbalance and memory contention.
- Dynamic scheduling had higher overhead at low thread counts but improved gradually as threads increased.
- Cache blocking significantly reduced execution time compared to normal scheduling, showing better cache utilization and memory locality.
- perf statistics indicate a high backend bound percentage, meaning the simulation is largely memory-bound rather than compute-bound. Branch misses are low, so control flow is efficient.

Conclusion

The 2D heat diffusion parallel implementation is correct and benefits from multithreading, but scaling is limited by memory bandwidth and cache behavior rather than pure computation. Guided scheduling and cache blocking provide the best overall performance improvements, while very high thread counts yield diminishing returns. The experiment demonstrates that in numerical grid-based simulations, memory access patterns and scheduling strategy have a stronger impact on performance than simply increasing the number of threads.