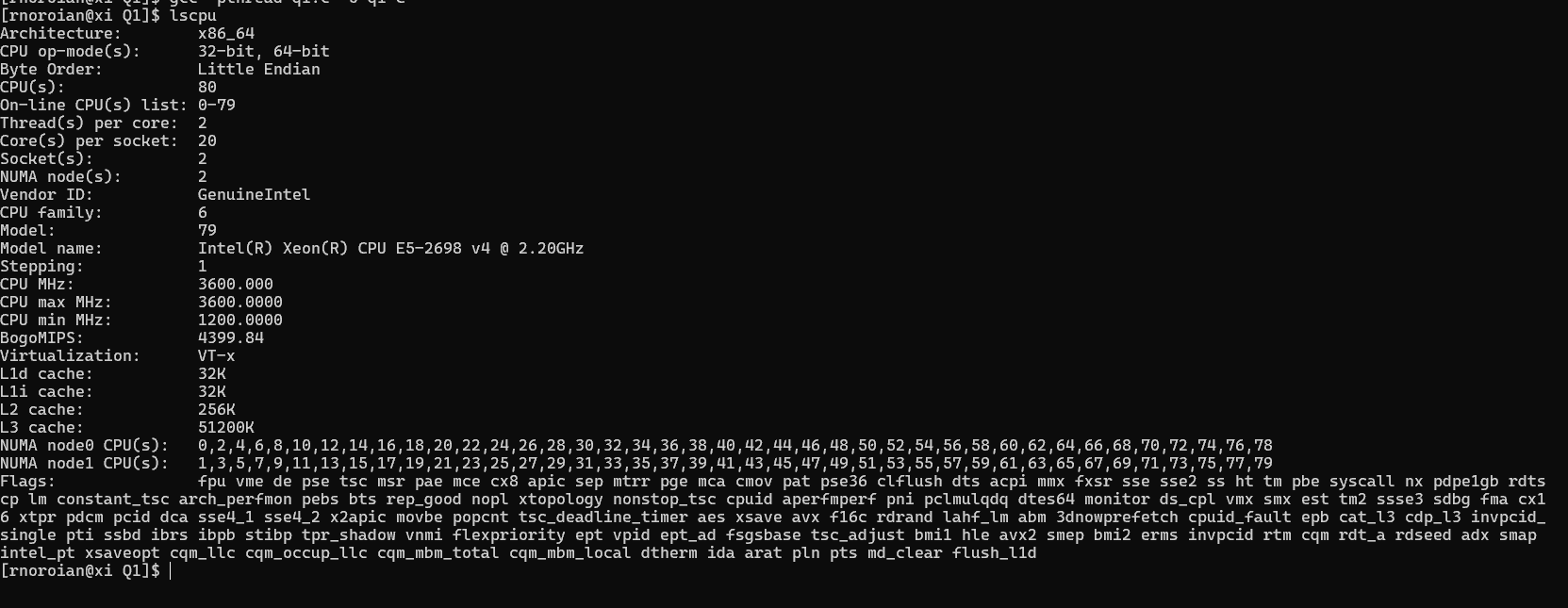
Question 1:

-Vector CPU Specifications

A) Pthread Implementation with Monte Carlo and Leibniz (Extra Credit)

Pthreads were used in both the Monte Carlo and Leibniz implementations of calculating Pi in an attempt to optimize speedup while maintaining accuracy.

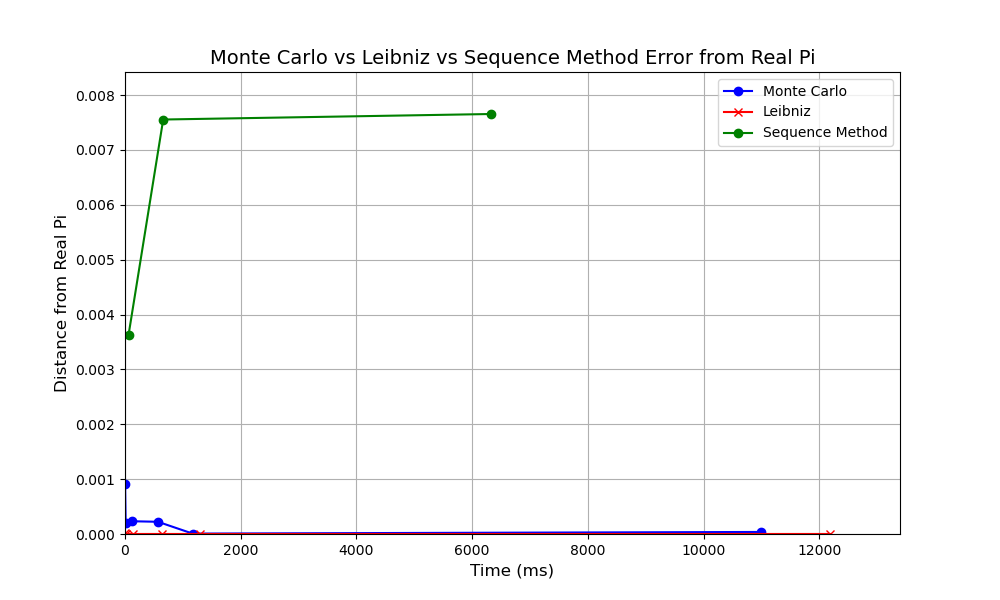
With Monte Carlo, the input number of darts was split evenly among the input number of threads. Each thread generated a random coordinate and measured its Euclidean distance to determine if it fell in the circle. The total number of darts that fell in the circle were summed per thread, then totally summed after threading concluded, and this was used to determine Pi. (Hits / Darts) \* 4

With Leibniz, the work in summing the infinite series is divided amongst threads, and then their totals are summed and multiplied by 4 to determine the estimation of Pi.

This table shows the average time for the Monte Carlo method and its average approximation at different dart amounts, the same for the Leibniz formula, and for the sequential code. For each number of points, 20 trials were run.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| #Threads = 20  Monte Carlo | T1 – 100,000 points | T2 – 1,000,000 | T3 – 10,000,000 | T4 – 50,000,000 | T4 – 100,000,000 | T-5  1,000,000,000 |
| Time (ms) | 1.762 | 11.65 | 117.112 | 569.06 | 1168.703 | 10983.683 |
| Approx | 3.1425 | 3.1414 | 3.14136 | 3.14137 | 3.1416 | 3.141630 |
| Leibniz |  |  |  |  |  |  |
| Time (ms) | 2.143 | 13.97 | 134.46 | 641.54 | 1298.01 | 12174.4 |
| Approx | 3.14158 | 3.141592 | 3.141593 | 3.141593 | 3.141593 | 3.141593 |
| Sequential |  |  |  |  |  |  |
| Time (ms) | 61.32 | 663.32 | 6321.11 | Too slow | Too slow | Too slow |
| Approx | 3.139 | 3.134 | 3.134 |  |  |  |

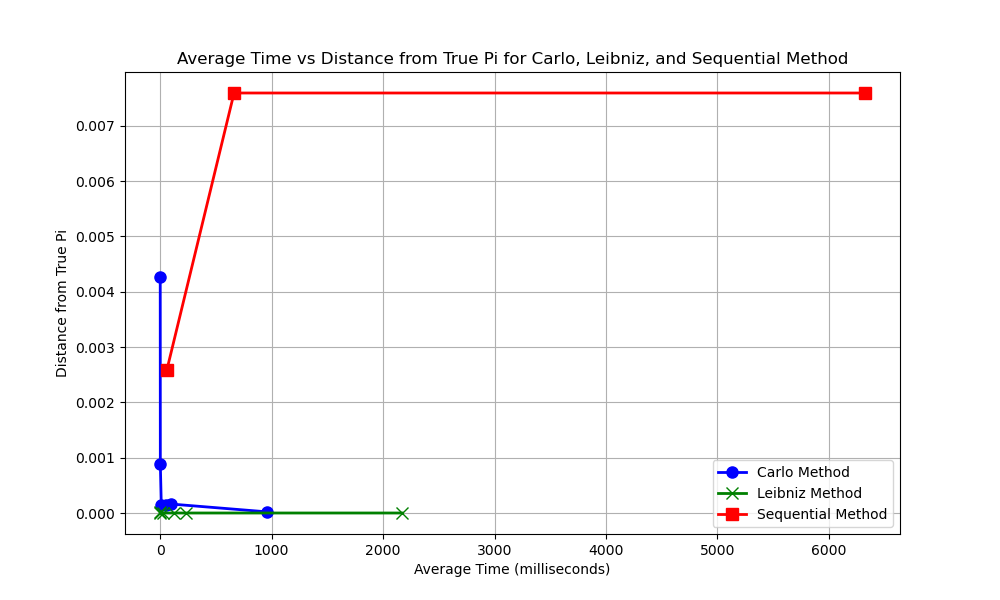
The graph below shows the Leibniz accuracy initially, and it demonstrates the slow convergence as Monte Carlo’s accuracy improves as the number of darts increases, while Leibniz accuracy remains the same. Both implementations are much faster than the sequential method.

Speedup with the Leibniz method was evaluated with the equation: speedup = *t*1 / *tN* , where t1 is the computational time for running with one processor and tN represents the computational time for N processors. The speedup is 28.6 for 100,000 darts, 47.5 for 1,000,000 darts, and 47.0 for 10,000,000 darts.

B) With OpenMP

The above evaluation was completed with OpenMP as well.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| #Threads = 20  Monte Carlo | T1 – 100,000 points | T2 – 1,000,000 | T3 – 10,000,000 | T4 – 50,000,000 | T4 – 100,000,000 | T-5  1,000,000,000 |
| Time (ms) | 1.036 | 1.908 | 10.433 | 52.03 | 96.408 | 957.336 |
| Approx | 3.1373 | 3.1424 | 3.1414 | 3.14144 | 3.14143 | 3.14161 |
| Leibniz |  |  |  |  |  |  |
| Time (ms) | 1.236 | 3.48 | 25.66 | 123.479 | 234.473 | 2171.476 |
| Approx | 3.141584 | 3.141592 | 3.141593 | 3.141593 | 3.141593 | 3.141593 |
| Sequential |  |  |  |  |  |  |
| Time (ms) | 61.32 | 663.32 | 6321.11 | Too slow | Too slow | Too slow |
| Approx | 3.139 | 3.134 | 3.134 |  |  |  |

A graph of the results is shown below for convergence: Speedup with the Leibniz method was evaluated with the equation: speedup = *t*1 / *tN* , where t1 is the computational time for running with one processor and tN represents the computational time for N processors. In this case, with 20 threads versus 1, the speedup increases as the number of darts increases. The speedup at 100,000 is 49.6, at 1,000,000 is 190.5, and at 10,000,000 is 246.7. This speedup is much greater than Pthread‘s, demonstrating the efficiency of OpenMP.

C) Strong + Weak Scaling Evaluation w/ Monte Carlo Simulation

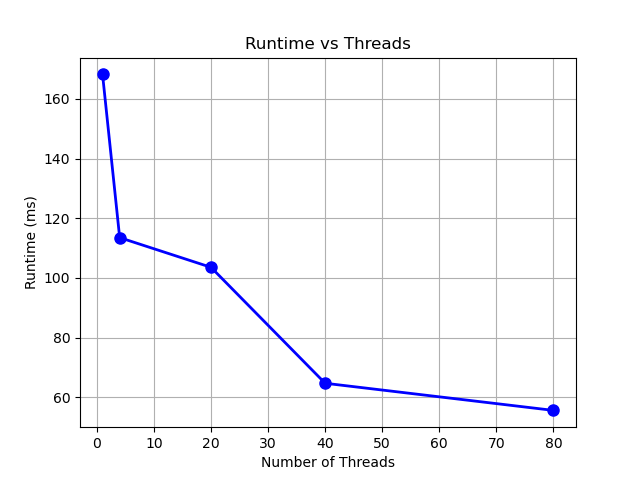
In part C, the Monte Carlo simulation implementation for Pthread and OpenMP was evaluated through each scaling.

Pthread:

Strong: Scaling the # of threads provided

To test strong scaling, the number of threads allocated to a specific workload volume was tested and analyzed. 10,000,000 darts was chosen. Threads were varied up to 80 since that was the maximum for the CPU specifications. Table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # Threads | 1 | 4 | 20 | 40 | 80 |
| Runtime (ms) | 168.218 | 113.524 | 103.571 | 64.721 | 55.625 |

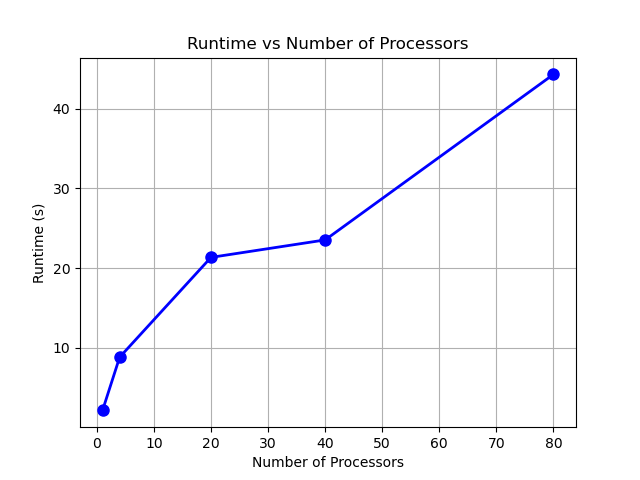


This demonstrates good strong scaling, as the graph shows better speedup/ decreased runtime with an increase in threads. As the scale of change in threads decreases, the speedup decreases as well, showing the leveling off that strong scaling exbibits.

Weak: Scaling the # of darts thrown

To test weak scaling, the number of threads was increased linearly with the number of darts thrown.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # Threads | 1 | 4 | 20 | 40 | 80 |
| # Darts | 100,000 | 400,000 | 2,000,000 | 4,000,000 | 8,000,000 |
| Runtime (ms) | 2.156 | 8.832 | 21.344 | 23.534 | 44.291 |

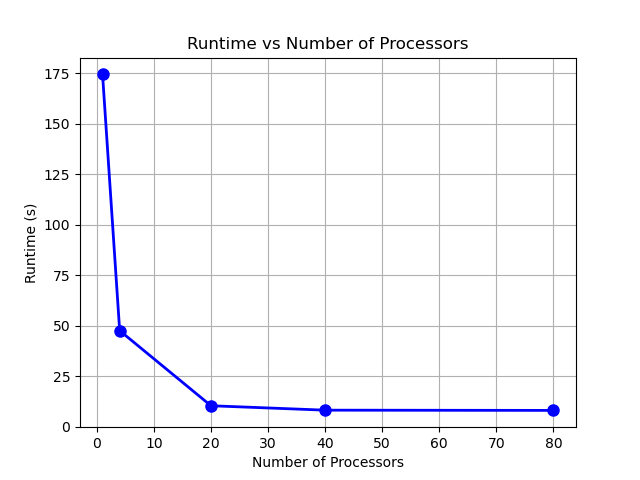
This graph shows that the program exhibits mediocre weak scaling, as the runtime should be about constant while the number of threads scales linearly with the workload. In this case, the workload almost doubles runtime for most changes in threads, meaning that speedup won’t be linear and will remain about the same for each number of threads.

OpenMP:

Strong: Scaling the # of threads provided

To test strong scaling, the number of threads allocated to a specific workload volume was tested and analyzed. A workload of 10,000,000 darts was again chosen.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # Threads | 1 | 4 | 20 | 40 | 80 |
| Runtime (ms) | 174.373 | 47.594 | 10.425 | 8.239 | 8.124 |

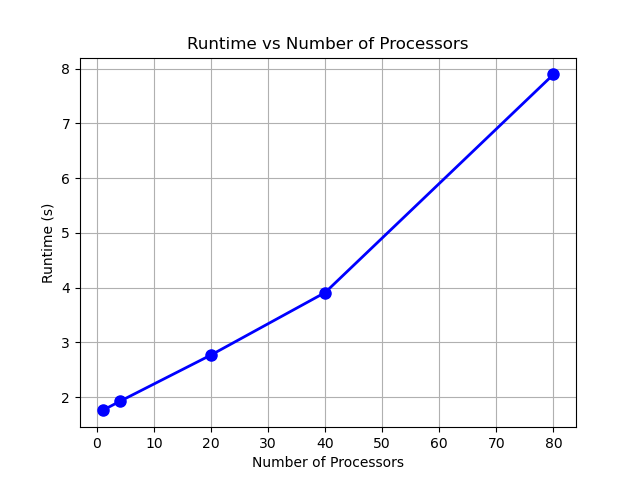


The OpenMP implementation shows very good strong scaling, as the greater the scale in difference of threads, the greater the speedup. As the number of processors doubles towards the end, and reaches 80 threads, the runtime begins to remain about constant, demonstrating the leveling off of the speedup at a certain point.

Weak: Scaling the # of darts thrown

To test weak scaling, the number of threads was increased linearly with the number of darts thrown.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # Threads | 1 | 4 | 20 | 40 | 80 |
| # Darts | 100,000 | 400,000 | 2,000,000 | 4,000,000 | 8,000,000 |
| Runtime (ms) | 1.757 | 1.921 | 2.769 | 3.910 | 7.894 |

Though this graph appears linear, the runtime itself for each scaled test differs very little compared to the Pthread implementation. In this case, runtime never exceeds 10ms, so I view differences in 1 to 2 ms as relatively insignificant, as overall speedup compared to runtime with 1 thread is going to increase as the over the number of processors. For this reason, I would consider this good weak scaling.