CS 531 HW 1

Hritvik JV

October 23, 2025

Abstract

This report details the performance and accuracy analysis of various parallel programming techniques applied to the calculation of π. We compare three synchronization mechanisms for numerical integration (Serial, OpenMP Critical, and OpenMP Atomic) against the Monte Carlo method across a varying number of threads (P) and problem sizes (N). Results demonstrate that parallel numerical integration achieves significant speedup. We also analyze the minor differences in overhead between the critical and atomic parallel directives.

1 Experimental Setup

The calculation of π was performed using two distinct methods:

1. **Numerical Integration:** Using the trapezoidal rule on the function $f(x) = 2\sqrt{1-x^2}$ over $[-1, 1]$.
2. **Monte Carlo Method:** Using random sampling within a unit square.

Parallelization was implemented using `OpenMP`. For the numerical integration, two different synchronization primitives were tested:

- **Critical Section (`#pragma omp critical`):** A block of code executed by only one thread at a time.
- **Atomic Operation (`#pragma omp atomic`):** A primitive that ensures the memory operation (addition to the sum) is performed atomically.

All experiments were run with 20 repetitions and averaged. The problem size was varied across three levels for comprehensive analysis: $N = 1 \times 10^8, 1 \times 10^9$, and 2×10^9. The thread count (P) was varied as $P = \{1, 4, 16, 128\}$.

2 Performance Analysis: Time and Speedup

2.1 Execution Time vs. Thread Count

The primary measure of parallel efficiency is the reduction in execution time as the number of threads increases.

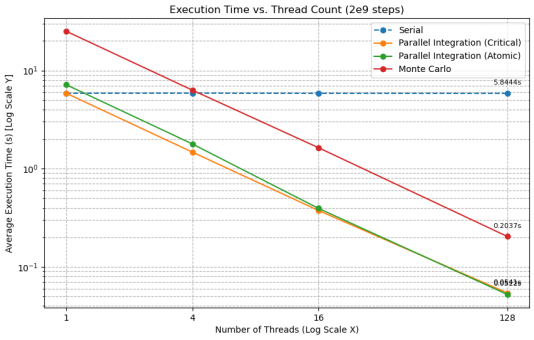

Figure 1: Average Execution Time versus Number of Threads (Log-Log Scale) for $N = 2 \times 10^9$. The Serial Integration time remains constant, serving as the performance ceiling.

The log-scale plot in Figure 1, 2 and 3 clearly demonstrates the substantial reduction in execution time for all parallel methods.

2.2 Comparison with Monte Carlo

The Monte Carlo method is inherently well-suited for parallelization due to its low data dependency. However, its overall execution time is significantly longer than the highly efficient parallel numerical integration, particularly for large N.

3 Accuracy Analysis: Error Convergence

The accuracy of the calculated π value is solely dependent on the total problem size (N), independent of the thread count (P). We analyze the convergence rate using the combined data across

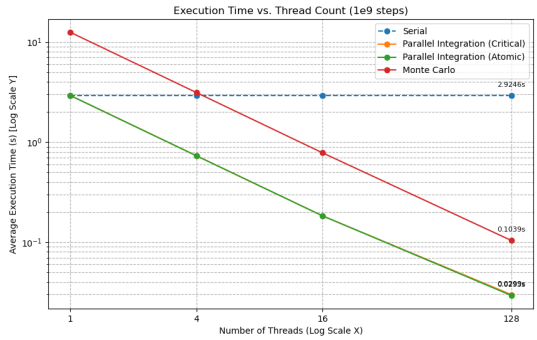

Figure 2: Average Execution Time versus Number of Threads (Log-Log Scale) for $N = 1 \times 10^9$. The Serial Integration time remains constant, serving as the performance ceiling.

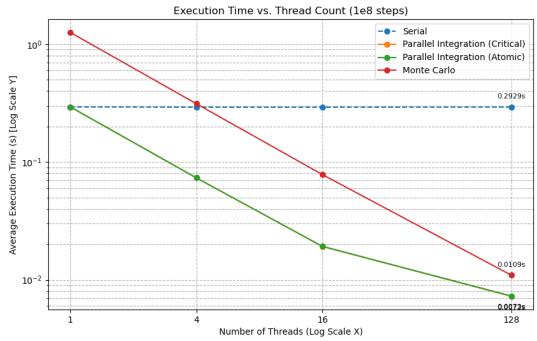

Figure 3: Average Execution Time versus Number of Threads (Log-Log Scale) for $N = 1 \times 10^8$. The Serial Integration time remains constant, serving as the performance ceiling.

all problem sizes.

3.1 Convergence Rates

- **Numerical Integration (All versions):** The error decreases rapidly resulting in a steep slope on the log-log plot.
- **Monte Carlo Method:** The error decreases slowly. This method requires a much larger increase in N to achieve the same level of accuracy as the numerical integration.

4 Conclusion

Parallel execution significantly reduced the computation time for both numerical integration and Monte Carlo methods. For the π calculation problem, the Numerical Integration method provides superior accuracy and performance, achieving high speedups (approaching $\times P$) and extremely low absolute error. The difference in overhead between OpenMP's `critical` and `atomic` primitives was negligible for this workload.

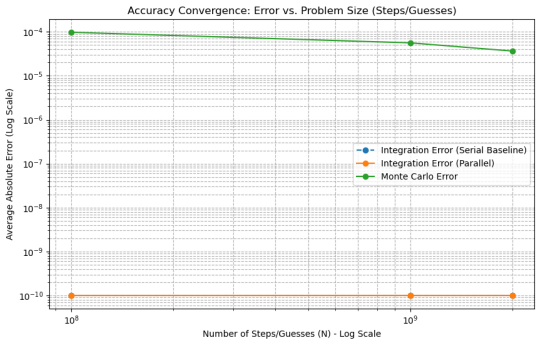

Figure 4: Average Absolute Error versus Problem Size (N) plotted on a log-log scale. This plot illustrates the theoretical convergence rates of the two methods.