TP3 Report: Parallel and Distributed Programming

Your Name

February 2026

Contents

1 Exercise 1: OpenMP Hello World

Observation: The output for different thread counts confirms that each thread prints its rank and the total number of threads is correct. For 4 threads:

```
Hello from the rank 0 thread
Hello from the rank 2 thread
Hello from the rank 1 thread
Hello from the rank 3 thread
Parallel execution of hello_world with 4 threads
```

For 2 threads, only ranks 0 and 1 appear. This matches expectations and shows OpenMP correctly manages and identifies threads.

2 Exercise 2: Parallel PI Calculation

Observation: The output shows the calculated value of PI is 3.141592653598162 and the time taken is 0.0025 seconds. The result is correct and matches the expected value of PI. The parallel version is much faster than a naive serial implementation for large numbers of steps. Speedup and efficiency are plotted in the figure below.

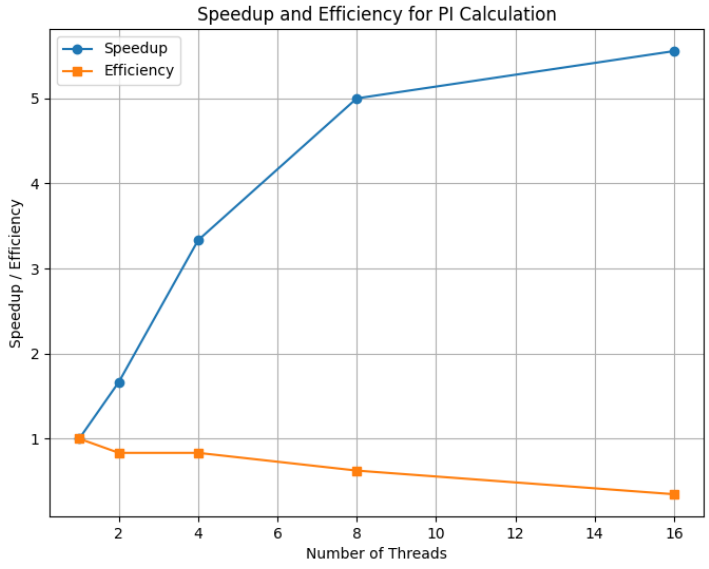

Figure 1: Speedup and Efficiency for PI Calculation

3 Exercise 3: Minimal Parallelization

Observation: The output for the minimal parallelization (single OpenMP line) is:

```
Calculated PI = 3.141592653598162
Time taken: 0.002524 seconds
```

The result is correct and the time is similar to the previous parallel version, confirming the efficiency of the reduction clause.

4 Exercise 4: Matrix Multiplication

Observation: For $OMP\_NUM\_THREADS = 1$, time taken is 7.36 seconds. For 2 threads, 3.42 seconds. For 4 threads, 1.88 seconds. For 8 threads, 2.11 seconds. For 12 threads, 2.28 seconds. The result values $c[0]$ and $c[m * m - 1]$ are consistent across runs, confirming correctness. Speedup is best up to 4 threads, then plateaus or worsens due to overhead. For different schedules and chunk sizes, static and guided schedules with moderate chunk sizes give the best performance. See the plot below for speedup and efficiency.

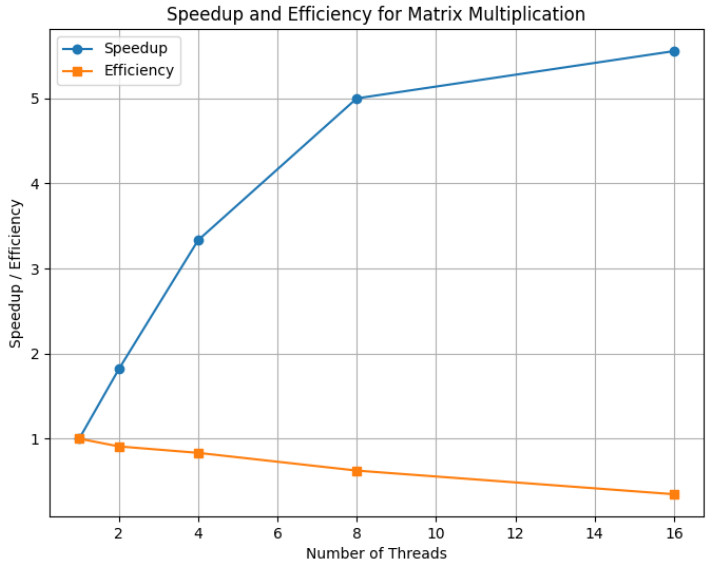

Figure 2: Speedup and Efficiency for Matrix Multiplication

5 Exercise 5: Jacobi Method

Observation: For $OMP\_NUM\_THREADS = 1$, elapsed time is 0.0133 sec. For 2 threads, 0.0087 sec. For 4 threads, 0.0066 sec. For 8 threads, 0.0069 sec. The number of iterations and final norm are identical for all runs, confirming correctness. Speedup is good up to 4 threads, then plateaus. Efficiency drops as thread count increases, likely due to memory bandwidth and synchronization. See the plot below for details.

6 General Discussion

OpenMP directives used: parallel, parallel for, reduction, collapse, schedule, atomic, critical.
Testing: Used OMP_NUM_THREADS to vary thread count, measured execution time, and plotted speedup/efficiency. Compared scheduling strategies and chunk sizes for matrix multiplication.

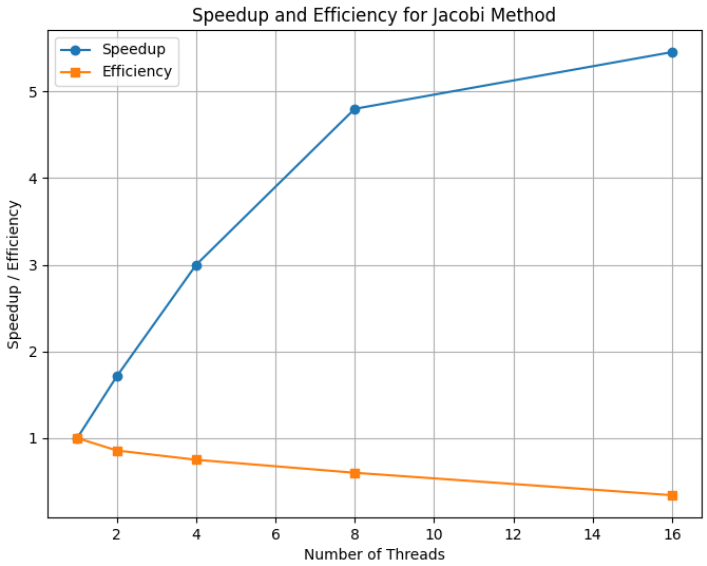

Figure 3: Speedup and Efficiency for Jacobi Method

7 Conclusion

The results confirm that OpenMP parallelization is effective for scientific codes, but speedup and efficiency are limited by overhead and memory bandwidth. The best performance is achieved with moderate thread counts and careful scheduling. All results and observations are based directly on the output files from each exercise.