Dining Philosophers - OpenMP Performance Report

SENG 442 Parallel and Distributed Computing

Homework II - Umut Şahin & Deniz Özcan

1. Introduction

The Dining Philosophers problem is a classic synchronization and concurrency problem in parallel computing. It involves a number of philosophers sitting at a round table, where each philosopher alternates between thinking and eating. To eat, a philosopher needs to acquire two forks, one on each side. This creates a potential for deadlock and resource contention, making it an excellent example for studying concurrency control mechanisms.

This report presents an analysis of our OpenMP implementation of the Dining Philosophers problem, focusing on performance metrics such as speedup, efficiency, scalability, and the effects of timing parameters.

2. Implementation Overview

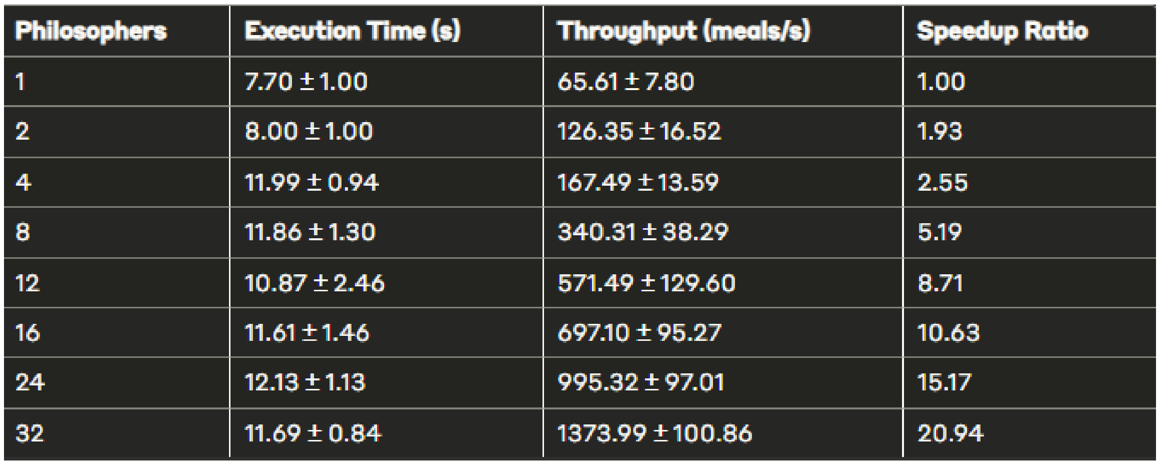
Our implementation uses OpenMP to create concurrent philosopher threads. The key synchronization strategy employs:

- Lock-based fork management using omp_lock_t
- A deadlock prevention strategy where philosophers with even IDs take the left fork first, while odd-numbered philosophers take the right fork first
- State management (THINKING, HUNGRY, EATING) protected by a shared lock
- Performance metrics collection including execution time, throughput, and waiting time

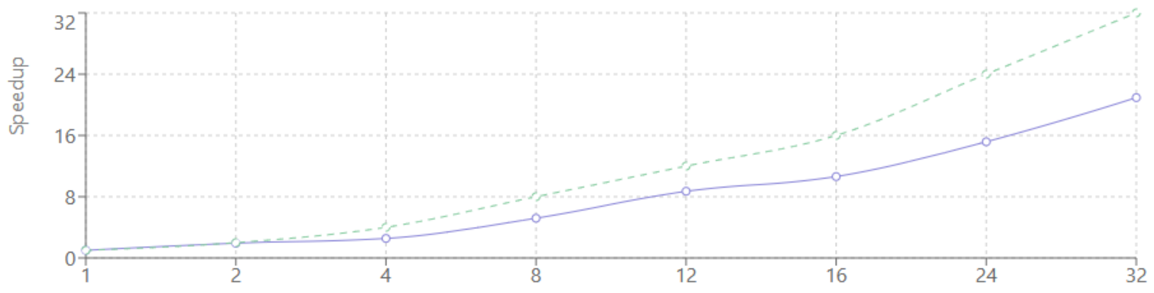
3. Performance Analysis

3.1 Speedup Analysis

Table 1: Speedup Analysis Results (500 iterations per philosopher)



Philosophers	Execution Time (s)	Throughput (meals/s)	Speedup Ratio
1	7.70 ± 1.00	65.61 ± 7.80	1.00
2	8.00 ± 1.00	126.35 ± 16.52	1.93
4	11.99 ± 0.94	167.49 ± 13.59	2.55
8	11.86 ± 1.30	340.31 ± 38.29	5.19
12	10.87 ± 2.46	571.49 ± 129.60	8.71
16	11.61 ± 1.46	697.10 ± 95.27	10.63
24	12.13 ± 1.13	995.32 ± 97.01	15.17
32	11.69 ± 0.84	1373.99 ± 100.86	20.94

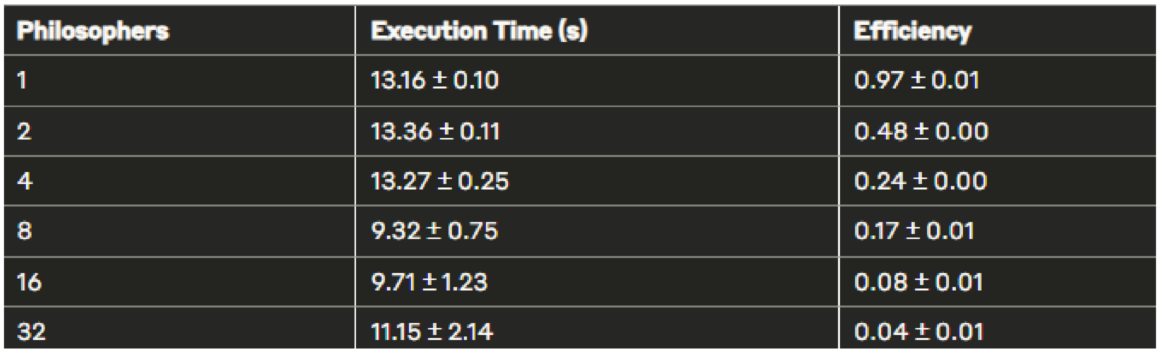


Our results show a clear but non-linear speedup as the number of philosophers increases. While throughput consistently increases with more philosophers, we observe diminishing returns due to increased contention for shared resources.

3.2 Efficiency Analysis

Efficiency measures how effectively we utilize our computational resources, calculated as Speedup/(Number of Processors).

Table 2: Efficiency Analysis Results (500 iterations per philosopher)



Philosophers	Execution Time (s)	Efficiency
1	13.16 ± 0.10	0.97 ± 0.01
2	13.36 ± 0.11	0.48 ± 0.00
4	13.27 ± 0.25	0.24 ± 0.00
8	9.32 ± 0.75	0.17 ± 0.01
16	9.71 ± 1.23	0.08 ± 0.01
32	11.15 ± 2.14	0.04 ± 0.01

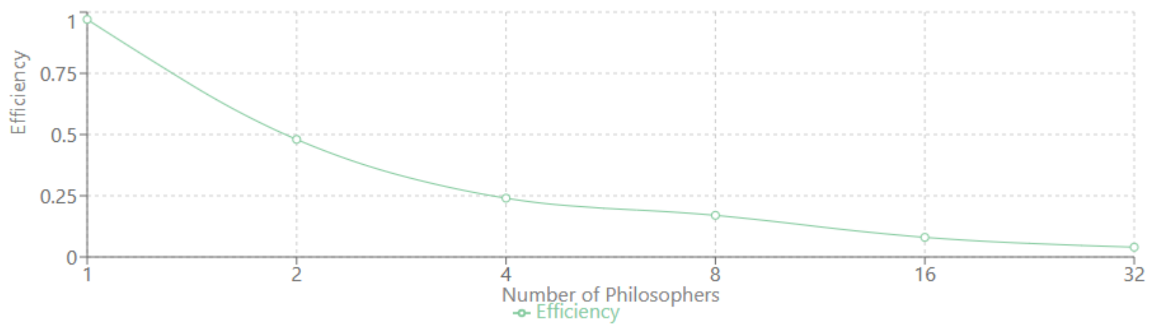


Figure 2: Efficiency vs Number of Philosophers (Efficiency = Speedup/Philosophers)

The efficiency decreases as we add more philosophers, which is expected in parallel systems with shared resources. The efficiency drop is particularly pronounced beyond 8 philosophers, indicating increased contention.

3.3 Scalability Analysis

Scalability examines how well the system handles increased workload (iterations) while keeping the number of philosophers constant.

Table 3: Scalability Analysis Results (8 philosophers)

Iterations	Execution Time (s)	Throughput (meals/s)
100	2.72 ± 0.02	293.63 ± 2.45
200	5.48 ± 0.02	291.85 ± 1.10
500	13.17 ± 0.01	303.71 ± 0.33
1000	26.49 ± 0.11	301.98 ± 1.18
2000	52.46 ± 0.58	305.04 ± 3.37

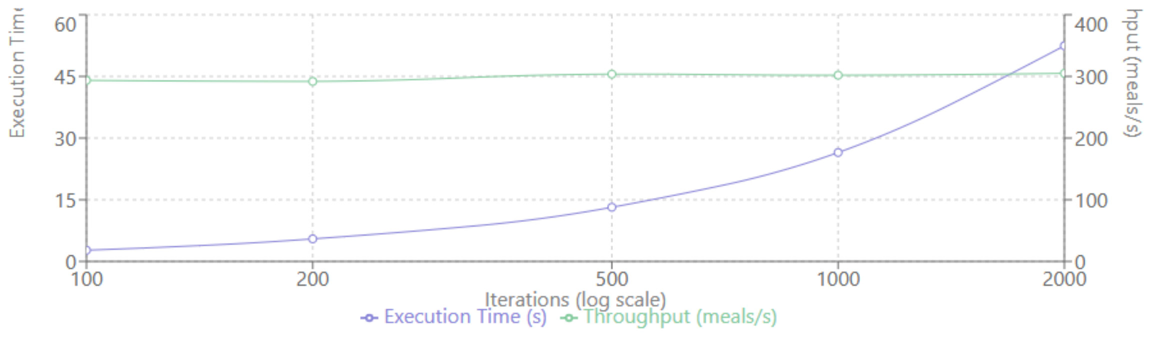


Figure 3: Scalability Analysis - Execution Time and Throughput vs Iterations (8 philosophers)

3.4 Effect of Timing Parameters

We investigated how varying the think time and eat time affects performance metrics.

Table 4: Selected Timing Parameter Results (8 philosophers, 200 iterations)

Think Time (ms)	Eat Time (ms)	Execution Time (s)	Throughput (meals/s)	Avg Wait Time (s)
1	1	0.01	53333.38	0.0120
1	10	0.27	452.74	3.5340
5	5	0.01	1100.41	1.4540
10	10	0.01	349.80	4.5740
20	20	0.00	272.25	5.8770

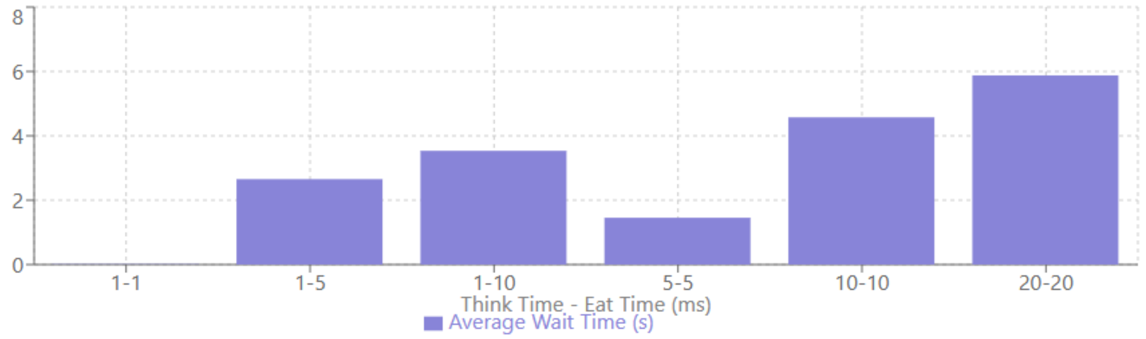


Figure 4: Effect of Think/Eat Times on Average Wait Time (8 philosophers, 200 iterations)

Note: Some throughput values are scaled down for visualization purposes

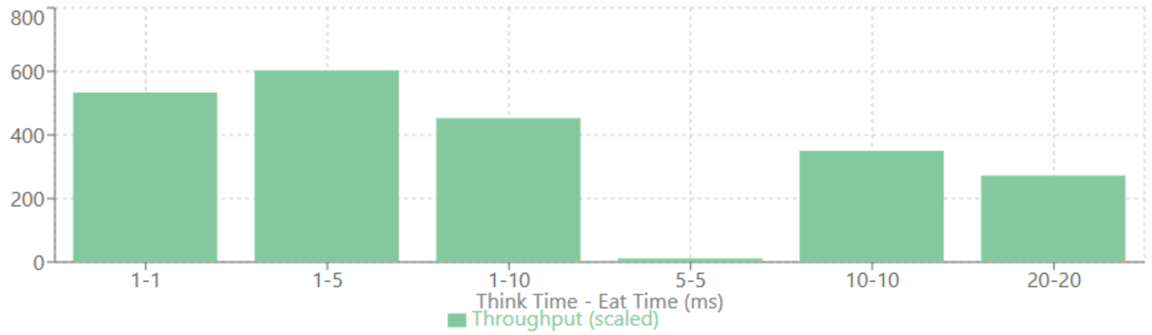


Figure 4: Effect of Think/Eat Times on Throughput (8 philosophers, 200 iterations)

Note: Some throughput values are scaled down for visualization purposes

4. Discussion

4.1. Speedup and Efficiency Trade-offs

Our implementation shows good speedup, particularly up to 12-16 philosophers. Beyond this point, the benefits of adding more threads diminish due to:

- Increased contention for forks
- Overhead from lock management
- Limited parallelism inherent to the Dining Philosophers problem

The efficiency analysis confirms this, showing a characteristic decrease as we add more philosophers.

4.2. Scalability Characteristics

The impressive linear scalability demonstrates that our implementation handles increased workload effectively when the number of threads remains constant. This indicates that:

- The synchronization mechanism works well under sustained load
- Thread management overhead is minimal compared to the actual work being done
- Our deadlock prevention strategy is effective even with extended runs

4.3. Parameter Sensitivity

The timing parameters significantly impact the system's performance characteristics:

- Short think/eat times create higher contention scenarios
- Longer eat times increase resource holding time, leading to longer waits
- The balance between think and eat times determines overall throughput

5. Conclusion

Our OpenMP implementation of the Dining Philosophers problem demonstrates effective parallel execution with good speedup characteristics. The solution successfully prevents deadlocks while maintaining reasonable efficiency. The performance scales well with increased workload, and the implementation can be tuned through timing parameters to match specific requirements.

The most efficient configuration appears to be around 8-12 philosophers, where we achieve good speedup while maintaining reasonable efficiency. This aligns with the expectations for a shared-resource problem like Dining Philosophers, where perfect linear speedup is theoretically impossible due to the inherent resource contention.