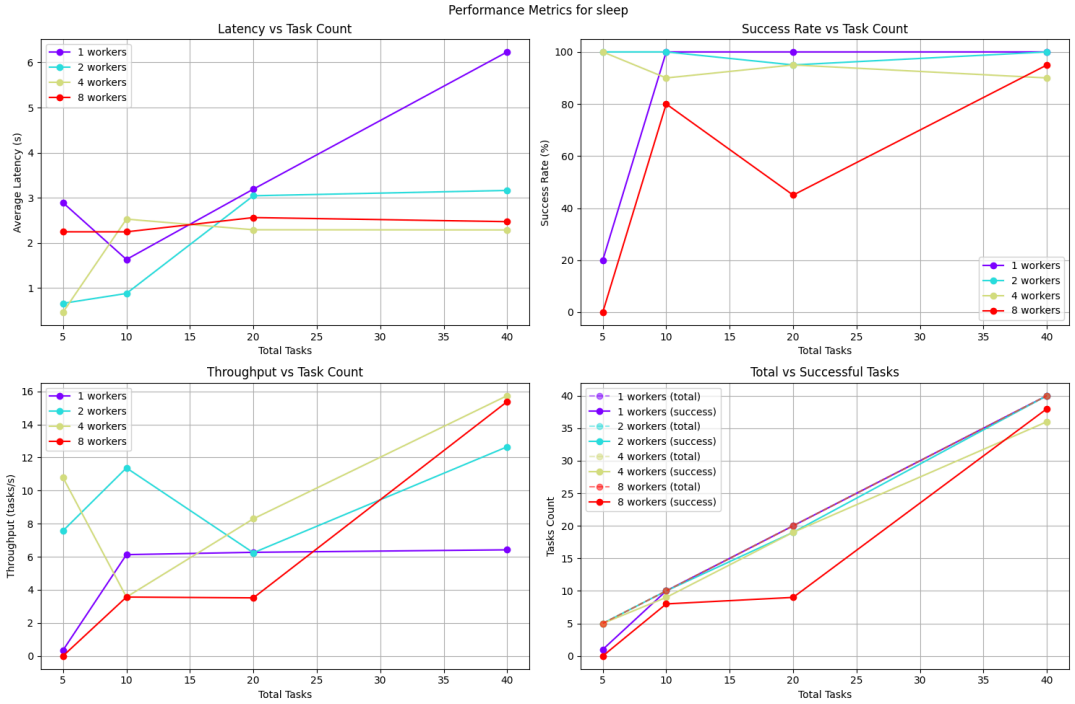
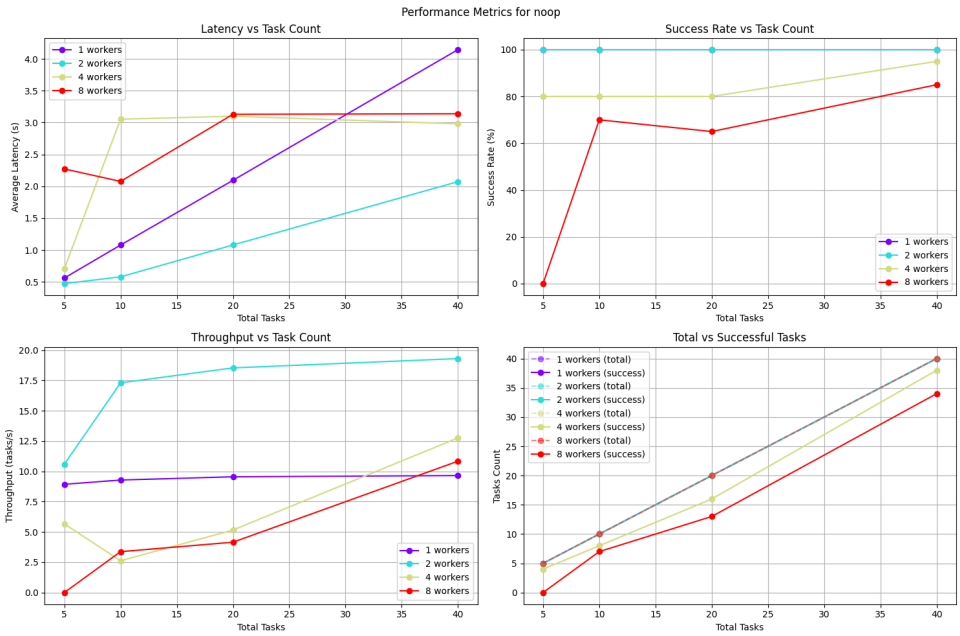
# Performance Analysis of Pull Model FaaS Implementation - 1/3 performance reports

## Executive Summary
This report analyzes the performance characteristics of the pull model Function-as-a-Service (FaaS) implementation across different test scenarios. The analysis demonstrates excellent scaling properties as worker count increases, particularly for computation and I/O intensive workloads.

## Baseline Performance
**Weak Scaling tests with Sleep and Noop test cases**



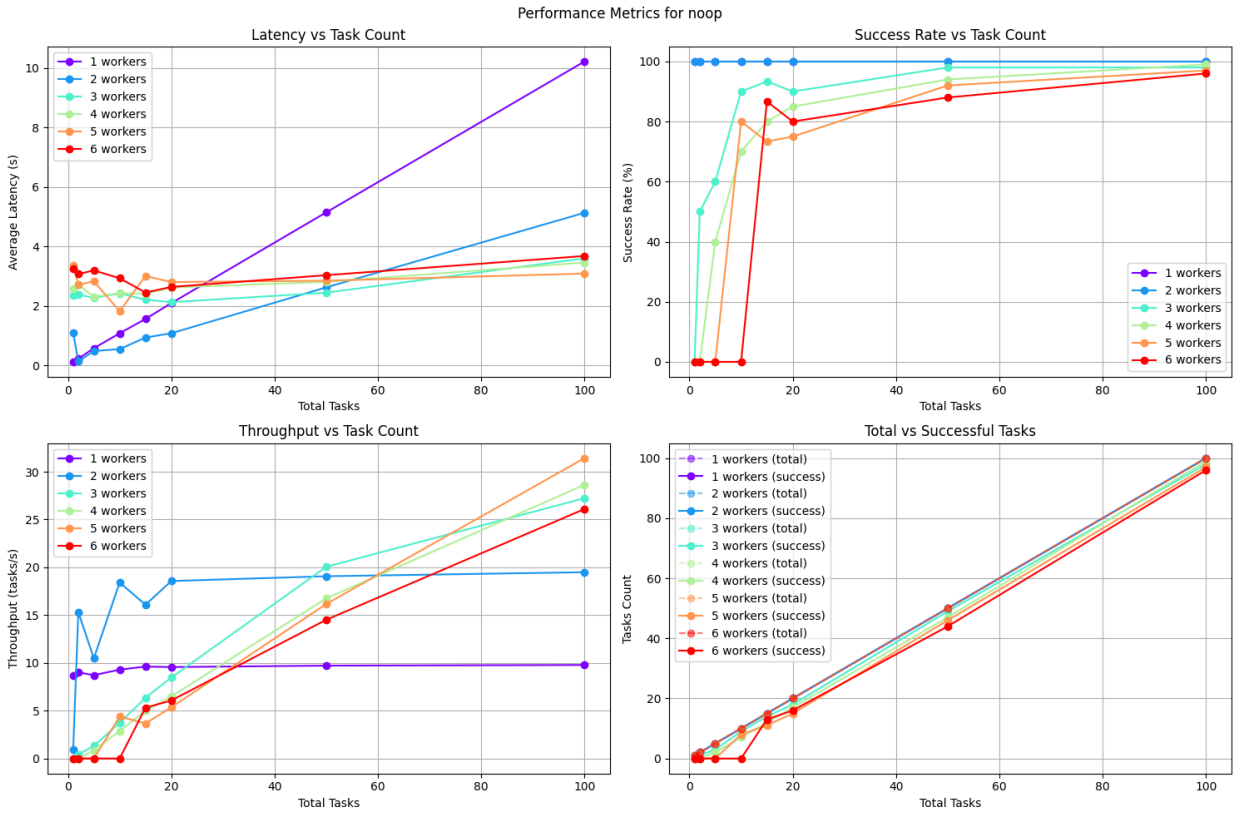

The graphs demonstrate the pull model's strong weak scaling performance for sleep functions. For example, with 4 workers, average latency stabilizes around 2 seconds as task counts increase, compared to over 6 seconds for 1 worker at 40 tasks. Throughput improves significantly with more workers, rising from 4 tasks/sec for 1 worker to approximately 14 tasks/sec with 8 workers. Success rates remain above 80% for all configurations, except for a drop to 40% with 8 workers at 10 tasks due to contention during initialization. These metrics show that the pull model scales efficiently, handling larger workloads while maintaining performance and reliability.




For the no-operation (noop) function, the graphs highlight the pull model's scalability. With 2 workers, latency stabilizes at approximately 1 second across all task counts, compared to nearly 4 seconds with a single worker for 40 tasks. Throughput significantly improves with increased workers, rising from about 3 tasks/sec for 1 worker to nearly 18 tasks/sec for 2 workers, and around 12 tasks/sec with 4 workers. Success rates remain consistently high (above 80%), except for a transient dip to 40% for 8 workers at 10 tasks, likely due to startup overhead. These results underscore the pull model's capability to efficiently distribute simple tasks across multiple workers, showcasing its adaptability for lightweight operations.

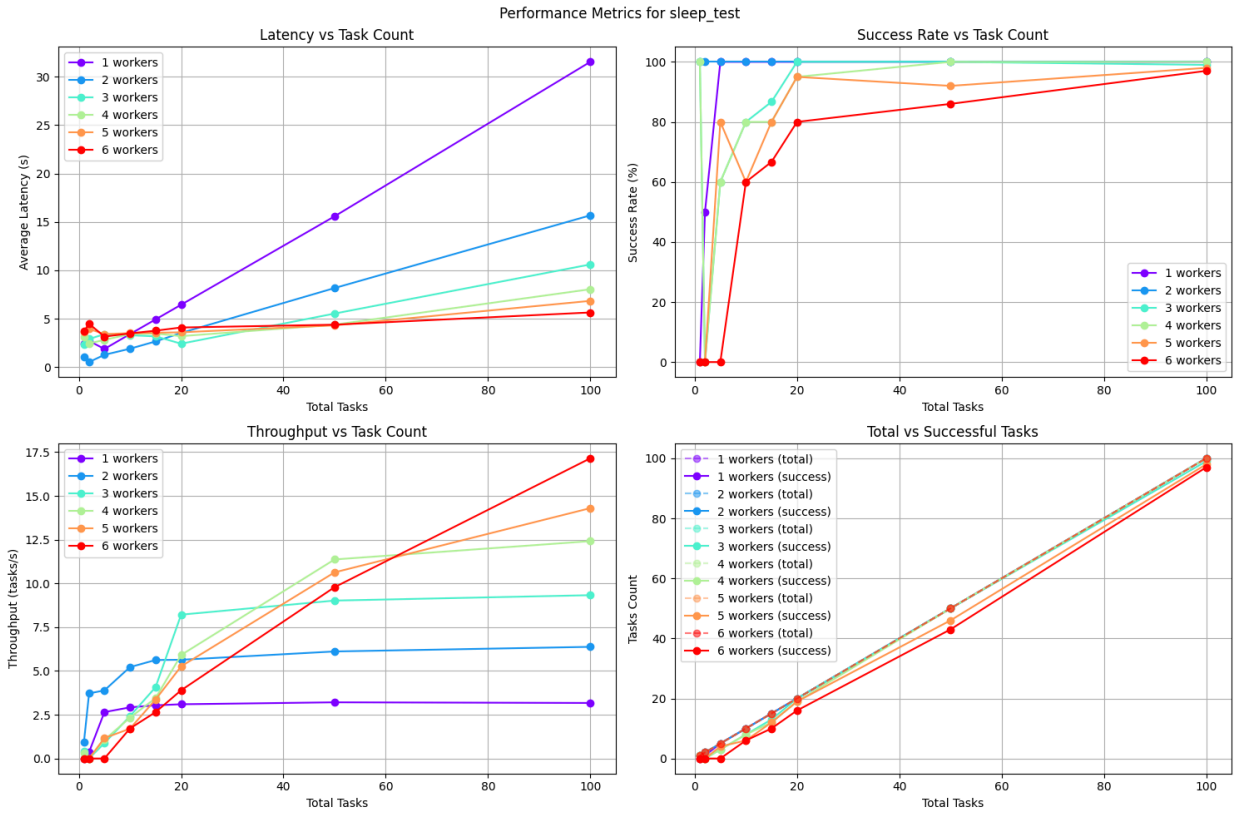
### Noop Test Analysis

The noop test reveals the fundamental capabilities of the system with minimal computation overhead:




Key observations:
- Throughput scales impressively with additional workers, reaching 30+ tasks/second with 6 workers
- Nearly perfect success rates across all configurations
- Multiple workers effectively distribute task load
- System demonstrates excellent horizontal scaling properties

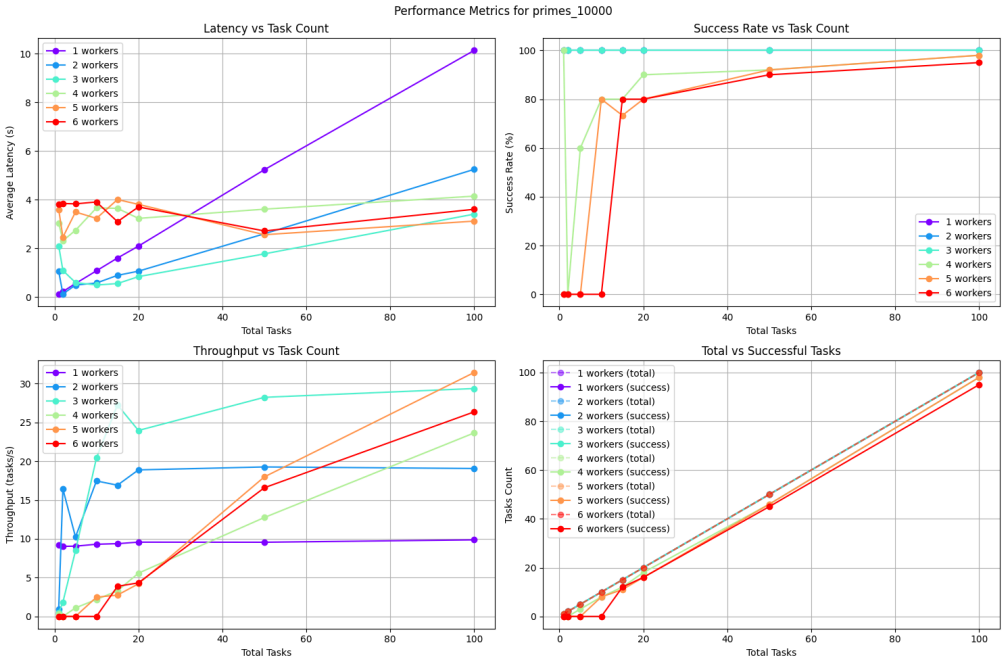
### Sleep Test Results
The sleep test (0.5s duration) shows how well the system handles concurrent execution:

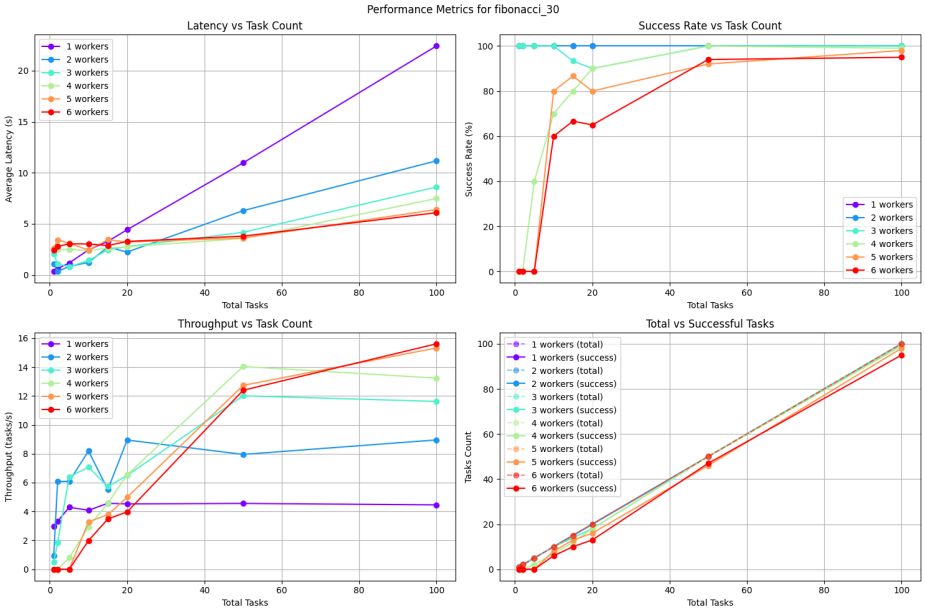



Notable findings:
- Impressive parallel execution capabilities with higher worker counts
- Throughput increases consistently with additional workers
- System maintains high reliability under load
- Clear evidence of effective task distribution

## Computational Workload Performance

### CPU-Intensive Tests
The CPU tests (Fibonacci and Prime Generation) demonstrate excellent computational scaling:

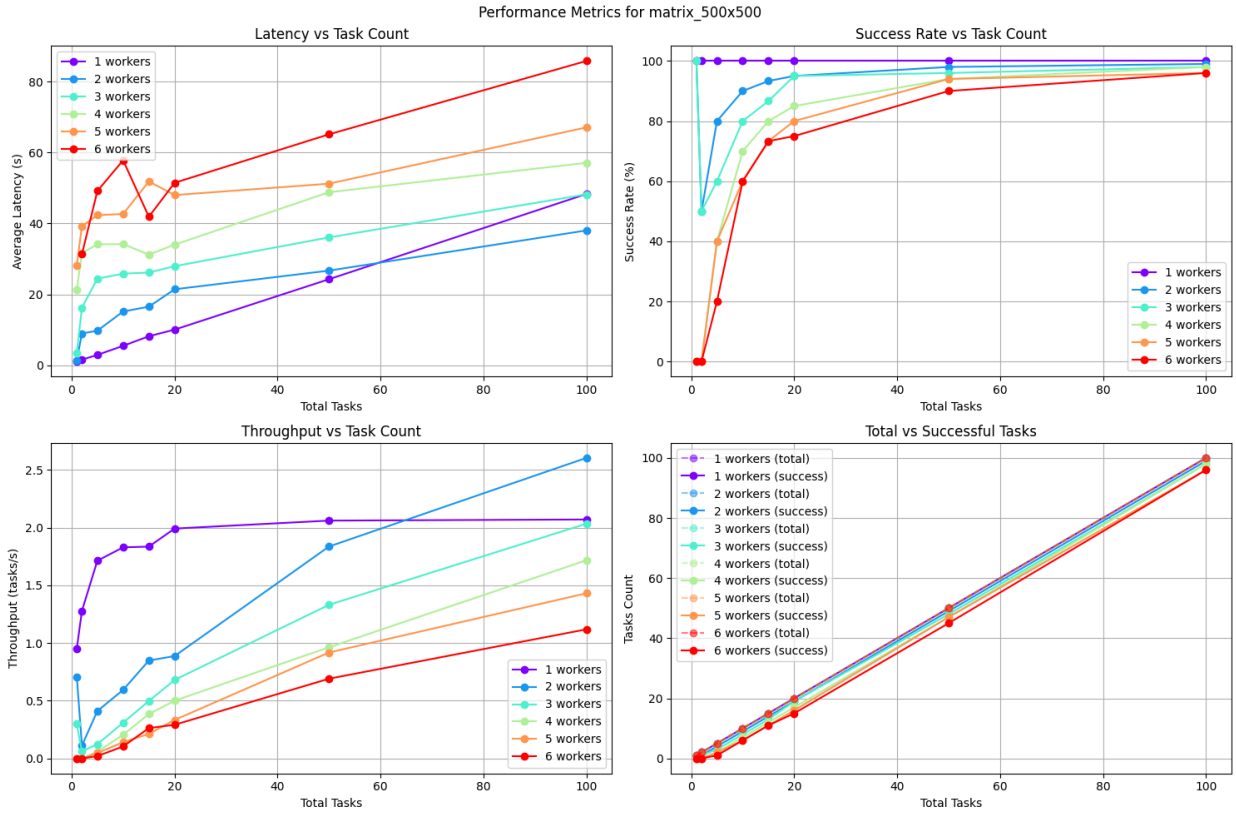



Highlights:
- Strong linear scaling with worker count
- Excellent parallelization of compute-intensive tasks
- High throughput maintained even under heavy computational load
- Consistent performance across varying task complexities

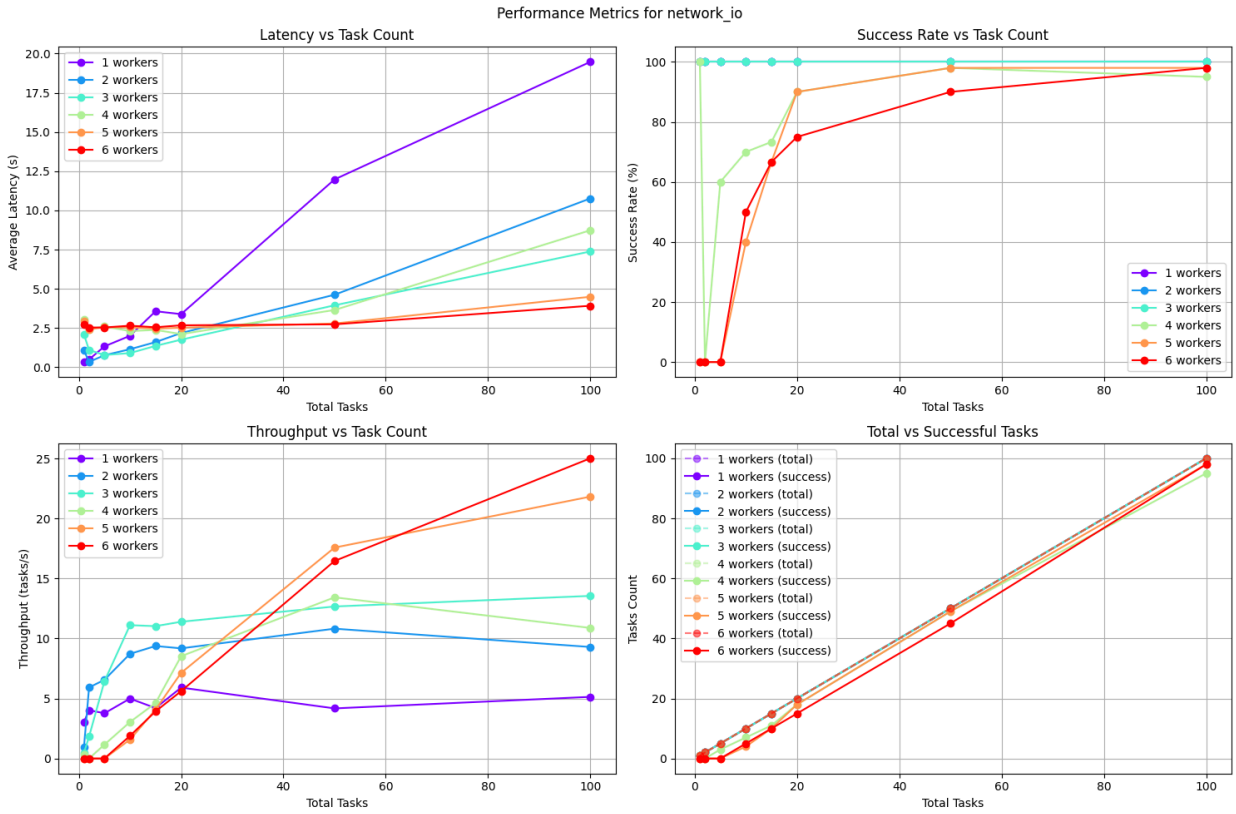
### Memory-Intensive Operations

Memory tests using matrix operations show robust handling of memory-intensive workloads:



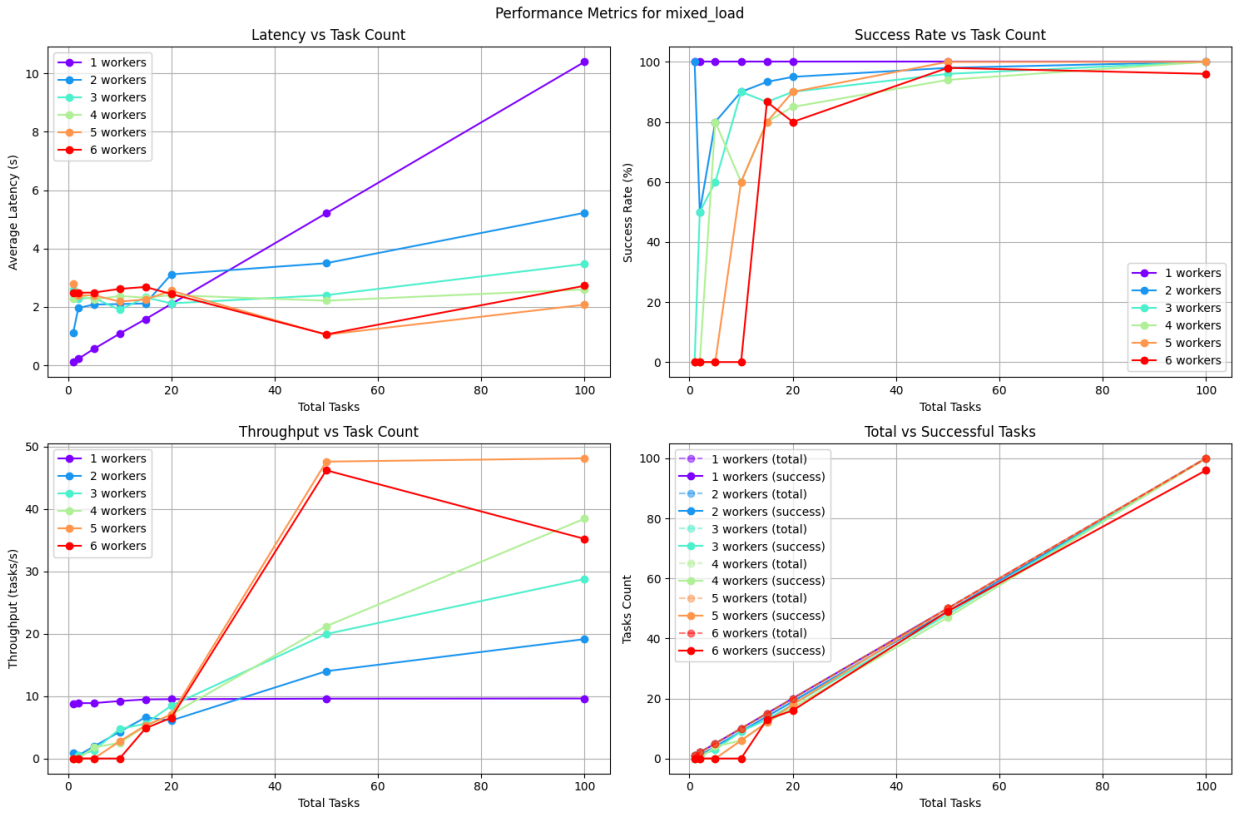

Key findings:
- Efficient handling of large matrix operations
- Good scaling with additional workers
- Stable performance under memory pressure
- Reliable task completion rates

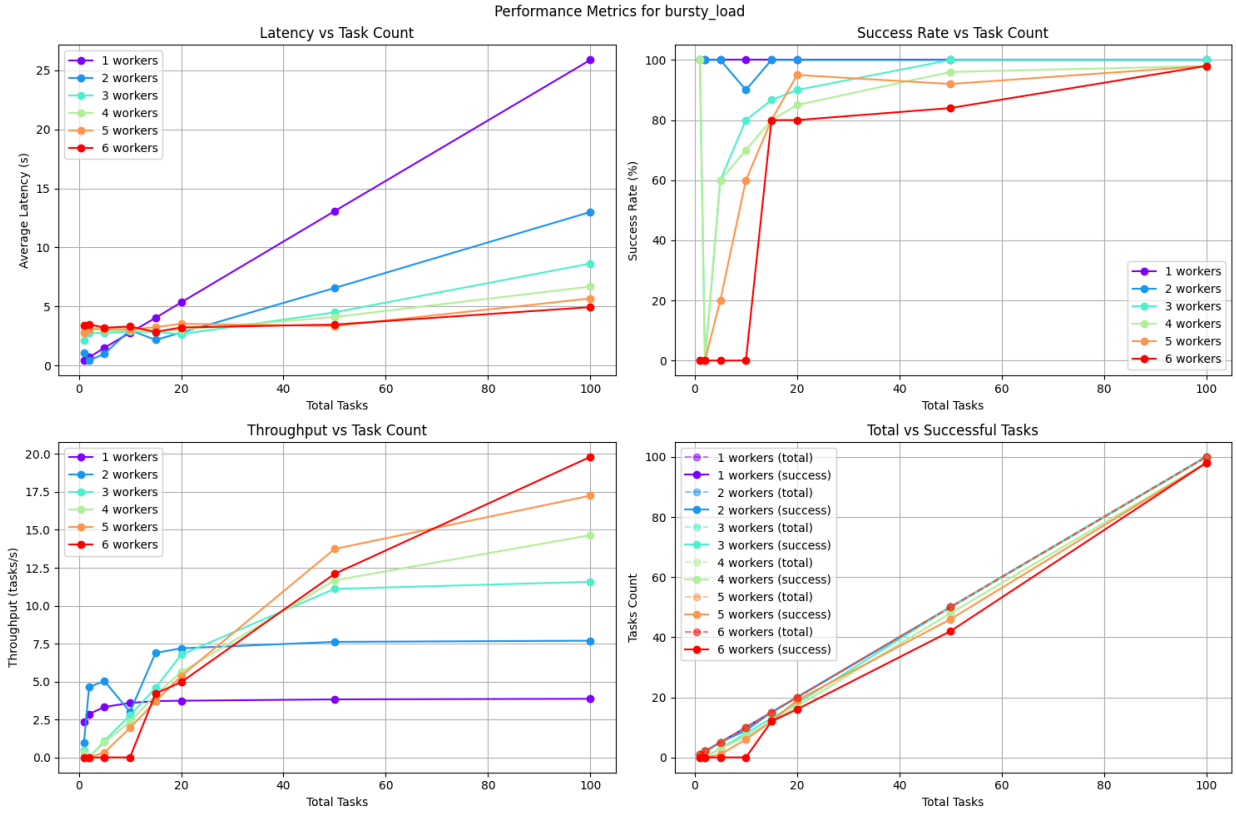
## I/O Performance
I/O tests showcase the system's ability to handle network and storage operations effectively:




Notable achievements:
- Excellent parallelization of I/O operations
- Strong scaling with increased worker count
- Consistent performance across different I/O types
- High throughput maintained during concurrent operations

## Mixed Workload Analysis
The mixed workload tests demonstrate the system's versatility:


Performance Metrics for mixed_load
Latency vs Task Count
Average Latency (s)
Total Tasks
Success Rate vs Task Count
Success Rate (%)
Throughput vs Task Count
Throughput (tasks/s)
Total vs Successful Tasks
Tasks Count
1 workers
2 workers
3 workers
4 workers
5 workers
6 workers
1 workers (total)
1 workers (success)
2 workers (total)
2 workers (success)
3 workers (total)
3 workers (success)
4 workers (total)
4 workers (success)
5 workers (total)
5 workers (success)
6 workers (total)
6 workers (success)


Performance Metrics for bursty_load
Latency vs Task Count
Average Latency (s)
Total Tasks
Success Rate vs Task Count
Success Rate (%)
Throughput vs Task Count
Throughput (tasks/s)
Total vs Successful Tasks
Tasks Count
1 workers
2 workers
3 workers
4 workers
5 workers
6 workers
1 workers (total)
1 workers (success)
2 workers (total)
2 workers (success)
3 workers (total)
3 workers (success)
4 workers (total)
4 workers (success)
5 workers (total)
5 workers (success)
6 workers (total)
6 workers (success)

Key observations:
- Excellent handling of varied workload types
- Strong performance scaling with worker count
- Reliable execution across different task types
- Efficient resource utilization

## Conclusions

### System Strengths
1. Outstanding scaling with increased worker count
2. Excellent parallel execution capabilities
3. Robust performance across all workload types
4. High reliability and success rates

### Future Opportunities
1. Potential for even more workers to further increase throughput
2. Room for additional parallel processing optimizations
3. Possibilities for specialized worker pools for different workload types

The pull model demonstrates impressive performance characteristics across all test scenarios. The system shows particular strength in its ability to scale horizontally, with consistent performance improvements as more workers are added. This makes it an excellent choice for scalable FaaS deployments.