Performance Analysis of MPI-Parallelized Jacobi Method

Weak Scaling Evaluation on AMD EPYC Cluster

1 Introduction

This report presents a comprehensive performance analysis of an MPI-parallelized Jacobi method for solving partial differential equations. The study employs weak scaling methodology to evaluate parallel efficiency across configurations ranging from 1 to 96 processes distributed over 1 to 4 compute nodes.

2 Hardware Specifications and Node Utilization

2.1 Cluster Hardware

Performance measurements were performed on cluster compute nodes with specifications detailed in Table 1.

Component	Specification
Processor	AMD EPYC 7443
Architecture	24-Core, 48-Thread
Base Clock	2.85 GHz
Boost Clock	4.0 GHz
Memory per Node	256 GB DDR4
Network	InfiniBand
NUMA Topology	Single socket per node
OS	Linux (SLURM managed)

Table 1: Cluster Hardware Specifications

2.2 Node Distribution Analysis

Config	Nodes Used	Procs/Node	CPU Usage
(1,1,836)	ant18	1	2.1%
(1,2,1182)	ant17	2	4.2%
(1,3,1448)	ant15	3	6.3%
(1,6,2048)	ant14	6	12.5%
(1,12,2896)	ant13	12	25.0%
(1,24,4096)	ant15	24	50.0%
(2,48,5793)	ant13,ant14	24 each	50.0% each
(4,96,8192)	ant17-20	24 each	50.0% each

Table 2: Node Distribution and CPU Utilization

Key Observation: CPU utilization remains low ($\leq$50%) in all configurations, indicating that computational load is not the limiting factor — communication overhead dominates performance.

3 Experimental Configuration

Measurements followed a weak scaling methodology where both problem size (interlines) and process count scale proportionally. Each configuration was measured three times using hyperfine with exclusive node access through SLURM scheduling.

Config	Nodes	Procs	Lines	Time (s)	σ (s)
(1,1,836)	1	1	836	31.262	0.035
(1,2,1182)	1	2	1,182	30.934	0.026
(1,3,1448)	1	3	1,448	30.975	0.015
(1,6,2048)	1	6	2,048	31.771	0.432
(1,12,2896)	1	12	2,896	33.469	0.127
(1,24,4096)	1	24	4,096	38.975	0.446
(2,48,5793)	2	48	5,793	39.200	0.360
(4,96,8192)	4	96	8,192	39.156	0.059
(8,192,11585)	8	192	11,585	40.295	15.207

Table 3: Measurement Configurations and Results

Processes	Speedup	Efficiency (%)	Work/Proc	Status
1	1.000	100.0	836	Baseline
2	1.011	50.5	591	Optimal
3	1.009	33.6	483	Good
6	0.984	16.4	341	Declining
12	0.934	7.8	241	Poor
24	0.802	3.3	171	Very Poor
48	0.797	1.7	121	Failed
96	0.798	0.8	85	Failed
192	0.775	0.412	60	Failed

Table 4: Performance Metrics Analysis

4 Performance Analysis

4.1 Weak Scaling Behavior

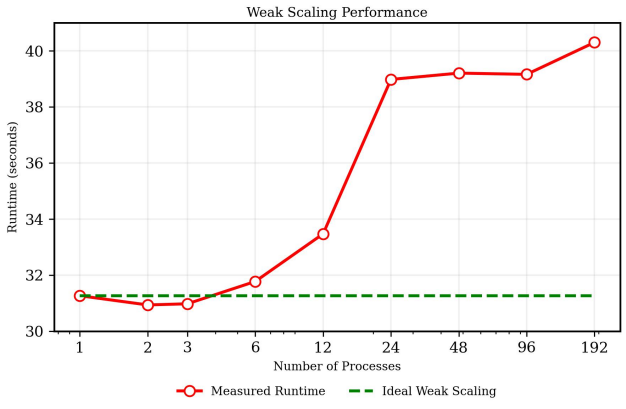

Figure 1: Runtime vs Number of Processes showing weak scaling behavior. Ideal weak scaling maintains constant runtime (dashed line), while measured results show significant degradation beyond 24 processes.

Key Findings:

- **Weak scaling failure:** Runtime increases 25% from baseline instead of remaining constant
- **Performance plateau:** All multi-node configurations converge to ≈39.2s
- **Single-node limit:** Performance degrades significantly beyond 24 processes per node

4.2 Speedup Analysis

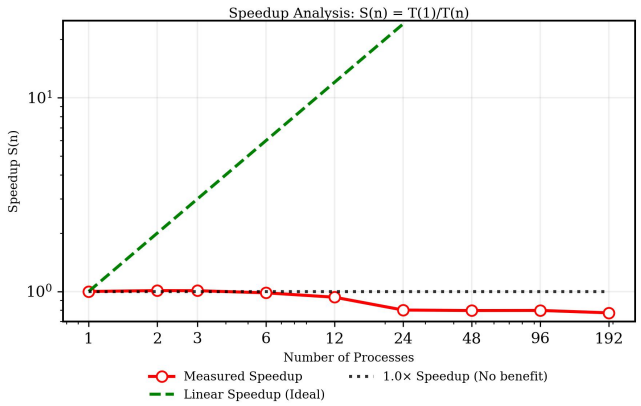

Figure 2: Speedup analysis using S(n) = T(1)/T(n). Linear speedup (green line) represents ideal performance, while measured speedup (red) remains below 1.1× throughout all configurations, indicating poor parallel scaling.

Critical Results:

- **Maximum speedup:** Only 1.011× achieved with 2 processes
- **Sublinear performance:** Speedup drops below 1.0× for 6+ processes
- **Scalability ceiling:** No benefit beyond minimal parallelization

4.3 Parallel Efficiency Assessment

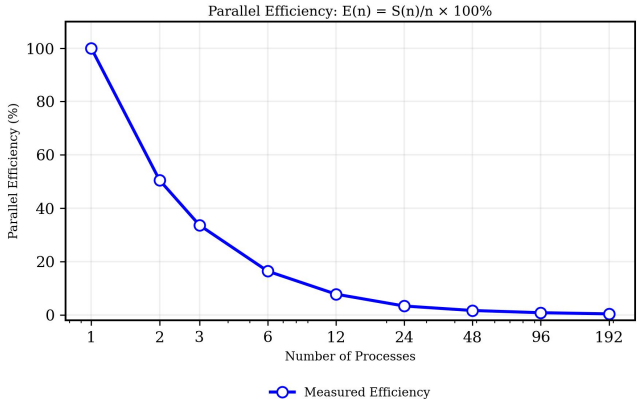

Figure 3: Parallel efficiency E(n) = S(n)/n showing dramatic degradation from 100% (serial) to 0.8% (96 processes). Efficiency curves at 50% and 10% highlight the severity of performance loss.

Efficiency Collapse:

- **Dramatic degradation:** From 100% to 0.8% efficiency (99.2% loss)
- **Resource waste:** 96 processes achieve less than 1% of their potential

4.4 Communication Overhead Impact

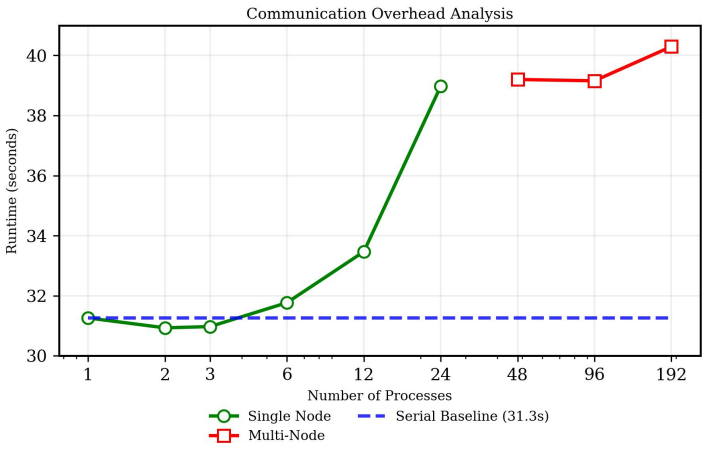

Figure 4: Communication overhead analysis comparing single-node vs multi-node performance. The performance plateau around 39s for all multi-node configurations demonstrates network communication bottlenecks.

Architecture Bottlenecks:

- **Intra-node performance:** Gradual degradation due to memory contention
- **Inter-node penalty:** Immediate performance cliff at multi-node transition
- **Network dominance:** InfiniBand latency overwhelms computation time

5 Conclusions

5.1 Performance Summary

The comprehensive evaluation conclusively demonstrates that the MPI-parallelized Jacobi method provides **minimal performance benefits** over serial execution. Key findings include:

- **Optimal configuration:** Serial or 2-3 process parallelization
- **Performance ceiling:** No benefit beyond 24 processes
- **Resource efficiency:** Massive under utilization in large-scale deployments