Homework 5 Report

KAI HOGAN

This report compares the performance of the SPMV COO algorithm using MPI communication using different configurations of compute nodes, MPI tasks per node, and CPU's per task.

CCS Concepts: • **Computing methodologies** → **Parallel algorithms**; *Shared memory algorithms*; • **Theory of computation** → *Design and analysis of algorithms.*

ACM Reference Format:
Kai Hogan. 2025. Homework 5 Report. *ACM Trans. Graph.* 1, 1, Article 1 (October 2025), 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Findings

1.1 Communication Times

1.1.1 Vector Broadcasting. Configurations 1 and 4 were the least efficient at broadcasting the vector to all processes, taking approximately 0.001 seconds each. The remaining three configurations performed about 10 percent faster, completing the operation in roughly 0.0009 seconds. Configuration 5 achieved the fastest vector broadcasting time at 0.000803 seconds. The relatively small variation across configurations suggests that broadcasting a single vector is not a significant bottleneck.

1.1.2 Vector Broadcasting. All configurations performed similarly for the matrix scatter operation, with times ranging between 0.015 and 0.030 seconds. Configuration 5 achieved the fastest time at 0.017 seconds, while Configuration 4 was slightly slower at 0.030 seconds. Despite the range appearing modest, Configuration 5's scatter time was around 40 percent faster than Configuration 4's, indicating that having fewer MPI ranks (4 vs 8) reduces the overhead of distributing the sparse matrix data across processes.

1.1.3 Result Vector Reduction. The result vector reduction showed the most dramatic variation in performance across configurations. Configurations 2 and 3 took approximately 0.014-0.018 seconds, while Configurations 1 and 4 were faster at just under 0.009 seconds. Configuration 5 was by far the most efficient, completing the reduction in only 0.000742 seconds, more than 10× faster than Configurations 2 and 3. This pattern clearly demonstrates that reduction time scales with the number of MPI ranks: fewer ranks mean less communication overhead during the collective reduction operation, making it a critical factor in multi-node performance.

Author's Contact Information: Kai Hogan, khogan@uoregon.edu.

ACM 1557-7368/2025/10-ART1
https://doi.org/XXXXXXX.XXXXXXX

Communication Times (seconds)

Config	Nodes	Total Cores	Vec Bcast	Mat Scatter	Res Reduce	Comm Total
1	1	8	0.001048	0.025891	0.008634	**0.035573**
2	2	16	0.000906	0.024747	0.013954	**0.039607**
3	4	32	0.000860	0.026293	0.018368	**0.045521**
4	1	32	0.001296	0.029553	0.007938	**0.038787**
5	4	112	0.000803	0.017163	0.000742	**0.018708**

Fig. 1. Communication Times

1.2 SpMV Computation Times

The first three configurations all exhibited similar SpMV COO performance, taking slightly under 0.3 seconds each: Configuration 1 completed in 0.225 seconds, Configuration 2 in 0.258 seconds, and Configuration 3 in 0.291 seconds. Configurations 4 and 5 were significantly more performant, taking 0.080 and 0.041 seconds respectively. Configuration 5's superior SpMV performance is attributable to its higher core count at 112 total cores; it had 3.5× more computational resources than the 32-core configurations. Since the SpMV COO code was parallelized using OpenMP, Configuration 5 could fully exploit this parallelism by assigning more threads to each MPI task. The most striking finding is that Configuration 4 was approximately 3.6× faster than Configuration 3 despite both having identical core counts (32 cores). This substantial performance discrepancy stems from Configuration 4 utilizing a single node while Configuration 3 distributed work across four separate nodes. When all processes share the same memory system on a single node, they benefit from better cache coherency and lower memory access latency.

Computation Times (seconds)

Config	Nodes	Total Cores	Lock Init	COO SpMV	Compute Total
1	1	8	0.000324	0.225072	**0.225396**
2	2	16	0.000327	0.258336	**0.258663**
3	4	32	0.000353	0.290528	**0.290881**
4	1	32	0.000294	0.079922	**0.080216**
5	4	112	0.000325	0.041558	**0.041883**

Fig. 2. SPMV and Lock initialization times

1.3 Total Execution Time

Configurations 1 through 3 demonstrated nearly identical overall performance despite their different node configurations. Configuration 1 took 0.261 seconds to complete the entire operation, Configuration 2 was slightly slower at 0.298 seconds, and Configuration 3 was the slowest at 0.336 seconds. Configurations 4 and 5 significantly outperformed the first three: Configuration 4 completed in 0.119 seconds, while Configuration 5 was nearly twice as fast at 0.061 seconds. This progression reveals that simply adding more nodes without adjusting the MPI-to-OpenMP balance provides no performance benefit and can actually degrade performance due to

increased communication overhead. SpMV COO was clearly the dominant bottleneck, consuming between 68 percent and 87 percent of total execution time across all configurations. Matrix scattering emerged as a secondary bottleneck, accounting for 9 percent to 29 percent of the total runtime. The fact that Configurations 4 and 5 dramatically reduced the SpMV computation time while maintaining relatively stable communication times explains their superior overall performance—they addressed the primary bottleneck through better parallelization strategy.

1.4 Key Conclusions

1. Configuration 5 was the fastest by a significant margin, nearly twice as fast as the second-best configuration. While its computational advantage (112 vs 32 cores) played a major role, its performance gains also stemmed from an optimal balance: fewer MPI ranks minimized communication overhead while maximizing OpenMP threads per rank fully exploited shared-memory parallelism.

2. Scaling nodes without scaling CPUs per task led to declines in performance. Configurations 1 through 3 progressively slowed down as more nodes were added while keeping CPUs per task constant at 1-4. This counterintuitive result reveals that the algorithm's communication overhead and inter-node synchronization costs outweigh any potential benefits from distributing work across more nodes. For this particular COO SpMV implementation, shared-memory parallelism (OpenMP) is far more effective than distributed-memory parallelism (MPI).

3. Result vector reduction time more than doubled from 1 to 4 nodes (0.009s → 0.018s), clearly showing that communication costs grow significantly with more MPI ranks. However, since result reduction represented only 1-6 percent of total execution time, this slowdown had an insignificant impact on overall performance.

Speedup Analysis (Relative to Config 1)

Config	Total Cores	Total Time	Speedup	Efficiency
1	8	0.260969	1.00×	100%
2	16	0.298270	0.87×	44%
3	32	0.336402	0.78×	24%
4	32	0.119003	2.19×	55%
5	112	0.060591	4.31×	31%

Fig. 3. Total Times