Performance Analysis of Inline Compression on Sparse Matrix Vector Multiplication

Precious Eyabi
peyabi@clemson.edu
Clemson University
Clemson, South Carolina, USA

Jon Calhoun (Advisor)
Clemson University
Clemson, South Carolina, USA
jonccal@clemson.edu

Abstract

Sparse matrix vector multiplication (SPMV) plays a critical role in scientific computing. From computational fluid dynamics, to artificial intelligence, to economics problems, the use cases are vast. Additionally, the data required for high performance computing (HPC) jobs is quickly outgrowing available memory storage. In-line compression aims to mitigate this memory bottleneck by keeping data compressed in storage and only decompressing as needed. We analyze the effects of performing this in-line compression with SPMV in an effort to minimize the overhead it causes while maximizing the overall memory footprint improvements.

CCS Concepts

• **Software and its engineering** → *Main memory.*

Keywords

Data Compression, Sparse Matrix Vector Multiplication

ACM Reference Format:
Precious Eyabi and Jon Calhoun (Advisor). 2026. Performance Analysis of Inline Compression on Sparse Matrix Vector Multiplication. In *Proceedings of The International Conference for High Performance Computing, Networking, Storage, and Analysis ('SC26).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction and Background

Sparse matrix-vector multiplication (SpMV) is a fundamental kernel in scientific computing. It has applications in tasks such as graph analytics, machine learning, and scientific simulations. SpMV computes $y = Ax$, where y and x are dense vectors and A is a sparse matrix. Because most of the elements in a sparse matrix are zeros, they are stored using specialized formats that record only the nonzero values. In this work, we evaluate three such formats: coordinate (COO), compressed sparse row (CSR), and compressed sparse column (CSC). Although these formats reduce storage requirements, their memory savings are often insufficient.

High-performance computing (HPC) applications generate increasingly large sparse datasets. For example, graph neural networks (GNNs) repeatedly perform SpMV and graphs with millions of parameters may exceed available memory. This limitation motivates additional memory reduction techniques such as inline compression.

Inline compression stores data in compressed form and decompresses only the portions required for computation. Previous work has shown that this approach can substantially reduce memory usage [3, 4], but compression and decompression introduce additional computational overhead. In this work, we evaluate the performance trade-offs of applying inline compression to COO, CSR, and CSC SpMV kernels to identify configurations that maximize memory savings while minimizing performance loss.

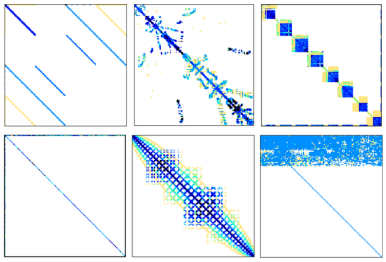


Figure 1: Images of the sparse matrices under evaluation

2 Methodology

To begin, we selected six sparse matrices from the SuiteSparse Matrix Collection [1]. The matrices used in our evaluation are shown in Figure 1 and represent the following application domains:

- Case39 - Power Network Optimization
- Bcsstk13 - Computational Fluid Dynamics
- Nxp1 - Circuit Simulation
- BundleAdj - Computer Vision
- Ga3As3H12 - Theoretical/Quantum Chemistry
- Poli4 - Economics Problem

These matrices were chosen because they represent a diverse set of real-world problems that require SpMV. Their varying sparsity patterns, structures, and numerical properties enable the analysis of how matrix characteristics influence performance.

We first measure the performance of the COO, CSR and CSC kernels without compression to establish a baseline. We then evaluate the effects of compression by first measuring the performance of full compression and decompression before examining inline compression. For the inline compression experiments, we evaluate implementations both with and without caching of previously decompressed blocks. For compression, we use the SZ compression algorithm [2] with a 0.001 error bound.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
'SC26, Chicago, IL
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

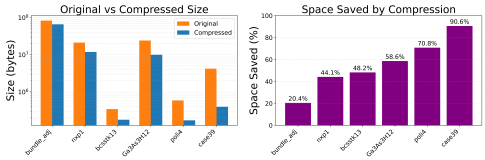


Figure 2: Amount of memory saved by compressing the nonzero values of the matrices

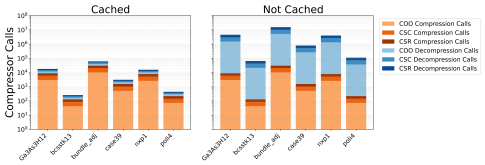


Figure 3: Number of compression and decompression calls with and without caching

Performance is evaluated using metrics including execution time and memory savings. For the compression experiments, we additionally measure the number of compression and decompression calls, the compression ratio, and the compression and decompression bandwidths to quantify the overhead and effectiveness of the compression techniques.

3 Results

The memory savings obtained through compression ranged from 20% to 90%. Depending on the application, these reductions can translate into substantial decreases in memory usage.

As shown in Figure 2, the effectiveness of compression is not determined solely by matrix size but rather by the matrix's structure and numerical characteristics. Among the matrices evaluated, *Case39* achieved the greatest memory savings. We hypothesize that this is because the matrix contains many similar values, allowing the SZ compressor to exploit its high spatial locality more effectively. Additional experiments using other lossy compression algorithms are needed to validate this hypothesis.

The performance cost of inline compression in SpMV is consistent with a broader pattern observed in high-performance computing, where application performance degrades as workloads become memory and latency bound rather than compute-bound. For example, on LinShine, the current #1 Top500 system [5], the dense, regular, compute-bound HPL benchmark achieves near-peak performance, whereas the sparse, irregular, memory-bound HPCG benchmark attains only about 1% of HPL performance on the same hardware. We observe a similar trend when introducing inline compression into SpMV. Across the COO, CSC, and CSR formats, the ratio of baseline throughput the compressed kernels achieved ranged from 0.8% to 1.2%. While this represents a significant performance penalty, it is consistent with the overheads commonly observed in memory-bound HPC workloads. Future work will investigate techniques to reduce this overhead while preserving the substantial memory savings provided by inline compression.

Regarding compression performance, we found that introducing a cache into the inline compression manager significantly reduced the number of decompression calls required during execution, as shown in Figure 3. By reusing previously decompressed blocks, the cache substantially reduced the number of decompression calls across all evaluated matrices, in several cases by more than an order of magnitude. Fewer calls decrease decompression overhead and improve the overall efficiency of the inline compression approach.

4 Conclusion and Future Work

This work is part of an ongoing research project. Our preliminary results show that inline compression can provide substantial memory savings, although these gains depend strongly on the characteristics of the sparse matrix and come at the cost of additional execution overhead. We also found that caching previously decompressed blocks reduces decompression overhead and improves overall performance.

Future work will investigate additional factors affecting performance, including block size, memory access patterns, compression error bounds, and the choice of compression algorithm, to better balance performance and memory savings. We also plan to identify the matrix characteristics that most strongly influence compression effectiveness, evaluate lossless compression of index structures, and study the energy efficiency of inline compression in HPC systems.

Acknowledgments

Clemson University is acknowledged for generous allotment of compute time on the Palmetto cluster.

References

[1] Timothy A. Davis and Yifan Hu. 2011. The University of Florida Sparse Matrix Collection. *ACM Trans. Math. Software* 38, 1 (2011), 1:1–1:25. doi:10.1145/2049662.2049663

[2] Sheng Di and Franck Cappello. 2016. Fast error-bounded lossy HPC data compression with SZ. In *2016 ieee international parallel and distributed processing symposium (ipdps)*. IEEE, 730–739.

[3] Emily Lattanzio, Sansriti Ranjan, Robert Underwood, Thomas Saupe, Robert Speck, and Jon C Calhoun. 2025. Performance Analysis of Inline Compression in pySDC. In *2025 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 1–8.

[4] Steven Platt and Jon C Calhoun. 2024. Analyzing an In-line Compression on the Matrix Matrix Multiplication Kernel. In *2024 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 1–11.

[5] TOP500.org. 2026. HPCG - June 2026. https://top500.org/lists/hpcg/2026/06/. Accessed: 2026-08-06.