POSTER: Parallel Algorithms for Masked Sparse Matrix-Matrix Products

Srđan Milaković
Rice University
Houston, TX, USA
sm108@rice.edu

Oguz Selvitopi
Lawrence Berkeley Nat. Laboratory
Berkeley, CA, USA
roselvitopi@lbl.gov

Israt Nisa
AWS AI
Palo Alto, CA, USA
nisisrat@amazon.com

Zoran Budimlić
Rice University
Houston, TX, USA
zoran@rice.edu

Aydın Buluç
Lawrence Berkeley Nat. Laboratory
Berkeley, CA, USA
abuluc@lbl.gov

Abstract

Computing the product of two sparse matrices (SpGEMM) is a fundamental operation in various combinatorial and graph algorithms as well as various bioinformatics and data analytics applications for computing inner-product similarities. For an important class of algorithms, only a subset of the output entries are needed, and the resulting operation is known as Masked SpGEMM since a subset of the output entries is considered to be "masked out". In this work, we investigate various novel algorithms and data structures for this rather challenging and important computation, and provide guidelines on how to design a fast Masked-SpGEMM for shared-memory architectures.

CCS Concepts: • **Software and its engineering → Software performance**; • **Mathematics of computing → Computations on matrices**; *Graph algorithms.*

Keywords: Masked-SpGEMM, Sparse Matrix, GraphBLAS

1 Introduction

Masked sparse-sparse matrix multiplication (*Masked SpGEMM*) is the problem of computing the product of two sparse matrices only for the set of entries given by the nonzero structure of the mask. The mask can be thought as a sparse matrix whose pattern determines which elements should exist in the output matrix. While the first use of this primitive was in the context of triangle counting, its applications include any multi-source graph traversal where the mask serves as a filter to avoid rediscovery of previously discovered vertices.

The existence of a mask in the multiplication introduces new optimization opportunities as well as challenges. A simple way to perform Masked SpGEMM is to compute the multiplication as if the mask does not exist and then apply the mask to the output matrix, which causes unnecessary computation if the overlap between the output matrix and the mask is low. The mask needs to be considered as part of the multiplication to attain good performance, which is the focus of this work. Our code implementing our algorithms and data structures is available at https://github.com/PASSIONLab/MaskedSpGEMM.

2 Algorithms

We describe four novel algorithms for Masked SpGEMM: Hash, Masked Sparse Accumulator (MSA), Mask Compressed Accumulator (MCA), and Heap. Three of these algorithms –Hash, MSA, and Heap– are novel improvements to the SpGEMM algorithms described in [1, 3, 5], whereas MCA, to the best of our knowledge, is a completely new algorithm specifically developed for Masked SpGEMM. All algorithms belong to the category of row-by-row push-based algorithms. The computational flow of row-by-row Masked SpGEMM algorithms is illustrated Figure 1. Each row $\mathbf{C}_{i*}$ is calculated as element-wise multiplication of $\mathbf{M}_{i*}$ and linear combination of each row $\mathbf{B}_{k*}$ for which $\mathbf{A}_{ik} \neq 0$, i.e., $\mathbf{C}_{i*} = \mathbf{M}_{i*} \odot \sum_{\mathbf{A}_{ik}\neq 0} \mathbf{A}_{ik}\mathbf{B}_{k*}$.

2.1 Accumulator

A key component in all the Masked SpGEMM algorithms is the *accumulator*, which is basically a data structure to merge scaled rows and can be considered as a set union operation. The design and the implementation of the accumulator has a significant impact on memory hierarchy behavior and therefore on the performance of Masked SpGEMM, and is the key differentiating feature between our proposed algorithms. In the following subsections we give a brief description of each accumulator. A more detailed description of different accumulators and our algorithms can be found in [4].

PPoPP '22, April 2–6, 2022, Seoul, Republic of Korea
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9204-4/22/04.
https://doi.org/10.1145/3503221.3508430

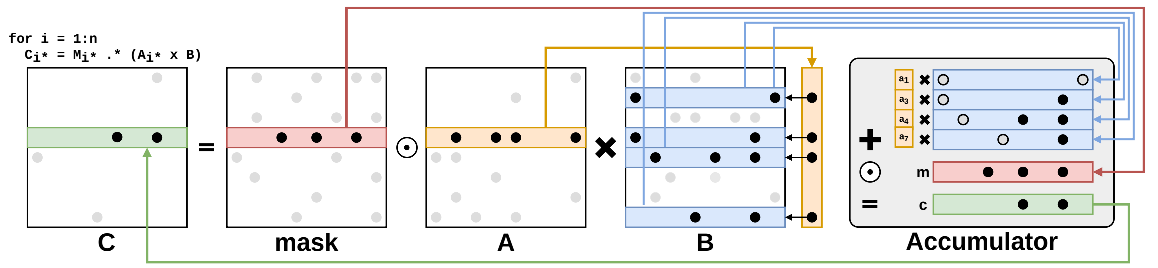

Figure 1. Row-wise Masked SpGEMM using an accumulator to compute output row $\mathbf{C}_{i*}$. The rows corresponding to the column indices of entries in row $\mathbf{A}_{i*}$ are merged and filtered through the respective mask entries to compute $\mathbf{C}_{i*}$. This merging and filtering process can be performed in a number of ways.

2.2 Masked Sparse Accumulator (MSA)

Internally, MSA uses two dense arrays, *values* and *states*, each with $ncols(\mathbf{B})$ length. The algorithm that uses MSA as accumulator has three main steps. First, the algorithm marks the values in the MSA that should not be discarded based on mask. Second, the algorithm finds all products $\mathbf{A}_{ik}\mathbf{B}_{kj} \neq 0$ and inserts them into MSA. Finally, the algorithm gathers all values from MSA, and resets the states in MSA.

2.3 Hash Accumulator

In practice, the arrays in the MSA accumulator are too large to fit in L1 cache, even though they usually have only a few nonzeros, so indexing an element of these arrays usually incurs a cache miss in the MSA algorithm. To overcome this issue, we utilize a hash map instead of dense arrays for storing values and states, reducing cache misses but increasing access overhead.

2.4 Mask Compressed Accumulator (MCA)

Mask Compressed Accumulator (MCA) algorithm is based on the observation that the number of elements in the accumulator cannot be greater than the number of nonzeros in the row $\mathbf{M}_{i*}$. The MCA accumulator uses size $nnz(\mathbf{M}_{i*})$ for the *values* and *states* arrays. To indexing MCA, we first find intersection between each row of $\mathbf{B}$ and current row of $\mathbf{M}$.

2.5 Masked Heap SpGEMM Algorithm

Masked SpGEMM algorithm is based on column-column Heap algorithm developed by Buluç and Gilbert [1]. Like the base algorithm, our heap algorithm requires that the indices in mask $\mathbf{M}$ and column indices in matrix $\mathbf{B}$ are sorted. Since the column indices are sorted, we can merge $\mathbf{A}_{ik}\mathbf{B}_{kj}$ product using the algorithm similar to multi-way merge and then intersect the result with elements from the mask.

3 Experimental Setup and Results

The experiments were conducted on a computer node with two Haswell with Intel Xeon E5-2698 processors. We compare our algorithms with SuiteSparse:GraphBLAS [2] using k-truss benchmark and 26 real-world matrices.

Figure 2 compares performance profiles of our three best performing algorithms and SS:GB. Our MSA implementation

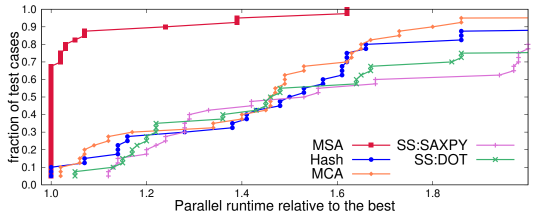

Figure 2. K-Truss: our algorithms vs. SS:GB.

significantly outperforms other implementations including SS:GB implementations.

4 Conclusion

In this paper, we presented four novel algorithms for performing parallel masked sparse-sparse matrix multiplication. We investigated Masked SpGEMM operation from various design and optimization standpoints to evaluate whether the challenges posed for plain SpGEMM still hold. We shown that our methods significantly outperform the SS:GB [2] library, which is, to the best of our knowledge, the fastest Masked SpGEMM implementation in existence to-date.

Acknowledgments

This work is supported by the Office of Science of the DOE under contract number DE-AC02-05CH11231. We used resources of the NERSC supported by the Office of Science of the DOE under Contract No. DE-AC02-05CH11231.

References

[1] Aydin Buluc and John R Gilbert. On the representation and multiplication of hypersparse matrices. In *IEEE International Symposium on Parallel and Distributed Processing*, pages 1–11. IEEE, 2008.
[2] Timothy A Davis. Algorithm 10xx: Suitesparse:graphblas: parallel graph algorithms in the language of sparse linear algebra. draft manuscript.
[3] Mehmet Deveci, Christian Trott, and Sivasankaran Rajamanickam. Multithreaded sparse matrix-matrix multiplication for many-core and gpu architectures. *Parallel Computing*, 78:33–46, 2018.
[4] Srdan Milakovic, Oguz Selvitopi, Israt Nisa, Zoran Budimlic, and Aydin Buluç. Parallel algorithms for masked sparse matrix-matrix products. *CoRR*, abs/2111.09947, 2021.
[5] Yusuke Nagasaka, Satoshi Matsuoka, Ariful Azad, and Aydın Buluç. Performance optimization, modeling and analysis of sparse matrix-matrix products on multi-core and many-core processors. *Parallel Computing*, 90:102545, 2019.