Fast Sparse General Matrix-Matrix Multiplication on GPU with Low Memory Usage

[Extended Abstract]

Yusuke Nagasaka *, Akira Nukada, and Satoshi Matsuoka
Tokyo Institute of Technology
Meguro, Tokyo, Japan
* nagasaka.y.aa@m.titech.ac.jp

I. Sparse General Matrix-Matrix Multiplication

Sparse general matrix-matrix multiplication (SpGEMM) is one of the key kernel of preconditioner such as algebraic multigrid (AMG) method or graph algorithms. Even though SpGEMM is matrix-matrix multiplication, the performance of SpGEMM is quite low since the memory access to both input matrix and output matrix is random. Since non-zero pattern of output matrix is unknown before execution, the execution should be done in 2-pass; first counts the number of non-zero elements of output matrix, then allocates memory and calculates values and column indices of output matrix. Existing work executes in 1-pass and accelerates SpGEMM on GPU with large memory usage. ESC (Expansion, Sorting and Contraction) algorithm [1] makes a list of intermediate products and sorts them by row and column index. It outputs the matrix by contracting the products with same row and column indices. BHSPARSE [2] focuses on load-balancing for irregular matrix data which has some dense rows and many of rows with few non-zero elements. BHSPARSE groups the rows by the number of intermediate products and applies appropriate merge algorithm with shared memory for each group. Anh et al. proposed BalancedHash algorithm [3], which improves imbalanced workloads by making the work list in global memory and inefficient random global memory access by devising hash table on shared memory. Hash table on shared memory works well, but this requires additional global memory usage for hash collision. Although existing work successfully accelerates SpGEMM on GPU by use large amount of memory, this property limits applicable matrix data. None of existing SpGEMM algorithm on GPU achieves high performance with low memory usage.

II. Proposal

We propose the state of the art algorithm which accelerates SpGEMM on GPU and reduces memory usage by utilizing shared memory and appropriate case analysis. We assume input matrices and output matrix are stored as CSR format. Our SpGEMM algorithm is executed in 2-pass and the flow of our approach is below.

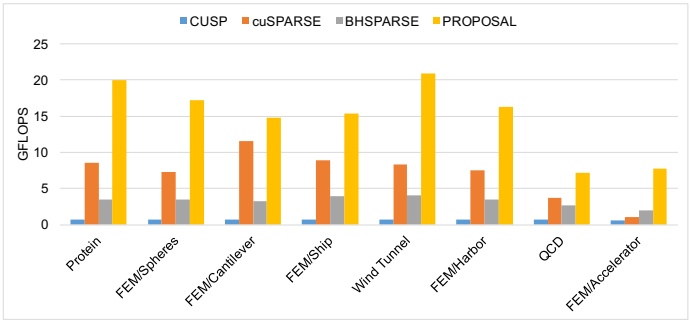
1) Count the number of intermediate product of each row
2) Group rows by the number of intermediate product
3) Count the number of non-zero elements of each row of output matrix
4) Set row pointer of output matrix to store as CSR format by scan operation
5) Group rows by the number of non-zero elements
6) Calculate values and column indices of output matrix
    a) Calculate values and column indices on hash table
    b) Shrink table to hold only non-zero elements
    c) Sort by column index in ascending order

First, the algorithm forms seven groups of rows by the number of intermediate products or non-zero elements and changes the thread assignment and other parameters to improve the load-balance. We construct efficient hash table at (3) and (6-a). The table size is set based on group and hash table is on shared memory, except a row has large number of non-zero elements. We adopt linear probing algorithm and the operation about hash table can be done on only shared memory. To achieve perfect coalesced memory access to input matrices, we devise two-way thread assignment and memory access. Not only memory access efficiency but also load-balancing can be improved by switching two ways based on the group, that is, the number of intermediate products or non-zero elements.

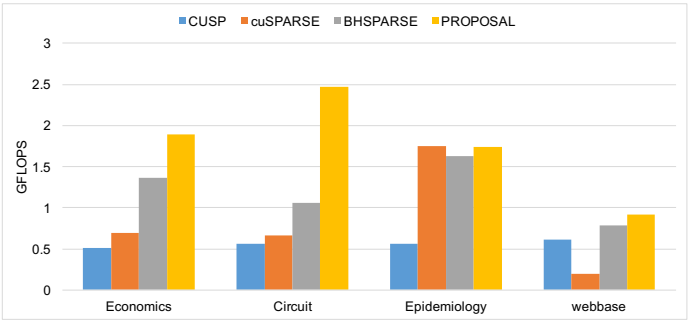
III. Evaluation

We evaluate the performance of our SpGEMM computations. For our evaluations, we select various 12 matrices from the University of Florida Sparse Matrix Collection [4]. Our evaluations have been done on NVIDIA's Quadro M6000 GPU. The GPU has 12 GBytes device memory and its peak memory bandwidth is 317 GByte/sec. There is also 96 KB shared memory on each SMM. GPU codes are implemented in CUDA 8.0RC.

Fig. 1 and 2 show the performance of SpGEMM computations in single and double precision, respectively. We compared our SpGEMM algorithm to the existing SpGEMM libraries; cuSPARSE, CUSP [5] and BHSPARSE. The performance evaluation shows that our approach overcomes existing libraries for all matrix data and achieves significant speedups of x28.7, x7.7 and x5.7 on maximum, and x15.3, x2.8 and x3.5 on average in single precision, respectively. The evaluation shows that our approach works well in double precision and it achieves speedups of x23.7, x6.1 and x4.9 on maximum, and x11.7, x2.5 and x3.0 on average, respectively. Our approach shows speedups of up to x4.0 in single precision and x3.3 in double precision compared to best libraries.

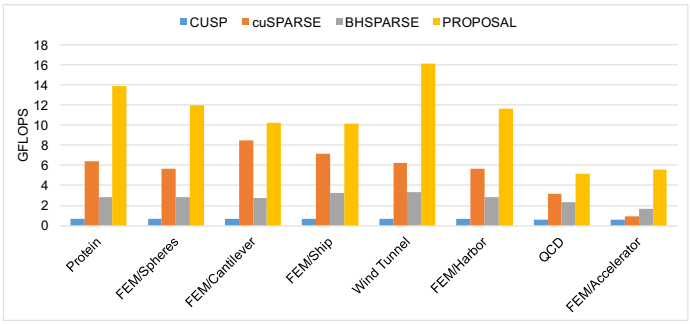


(a) High-Throughput matrices

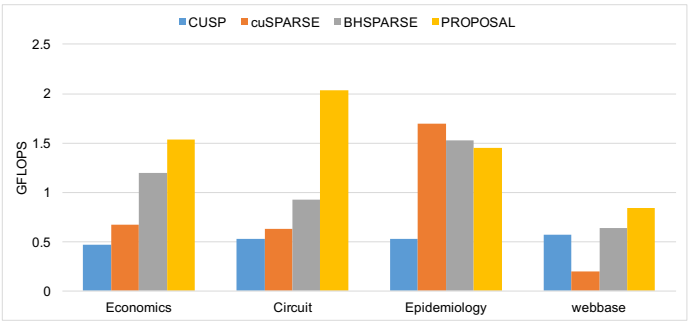


(b) Low-Throughput matrices

Fig. 1: Performance on SpGEMM computation in single precision



(a) High-Throughput matrices



(b) Low-Throughput matrices

Fig. 2: Performance on SpGEMM computation in double precision

IV. Conclusions

We propose the novel SpGEMM algorithm which is designed for reducing both memory overhead and execution time on GPU. Our algorithm achieves speedups of up to x4.0 in single precision and x3.0 in double precision compared to existing fast SpGEMM libraries. For future work, we will apply our technique to the preconditioner such as AMG method and real-world applications.

Acknowledgment

This work was partially supported by JSPS KAKENHI Grant Number 23220003, JST-CREST (Research Area: Advanced Core Technologies for Big Data Integration), and NVIDIA GPU Center of Excellence.

References

[1] S. Dalton, L. Olson, and N. Bell, "Optimizing sparse matrix—matrix multiplication for the gpu," *ACM Trans. Math. Softw.*, vol. 41, no. 4, pp. 25:1–25:20, Oct. 2015. [Online]. Available: http://doi.acm.org/10.1145/2699470

[2] W. Liu and B. Vinter, "An efficient gpu general sparse matrix-matrix multiplication for irregular data," in *2014 IEEE 28th International Parallel and Distributed Processing Symposium*, May 2014, pp. 370–381.

[3] P. N. Q. Anh, R. Fan, and Y. Wen, "Balanced hashing and efficient gpu sparse general matrix-matrix multiplication," in *Proceedings of the 2016 International Conference on Supercomputing*, ser. ICS '16. New York, NY, USA: ACM, 2016, pp. 36:1–36:12. [Online]. Available: http://doi.acm.org/10.1145/2925426.2926273

[4] T. Davis, "The University of Florida Sparse Matrix Collection." [Online]. Available: http://www.cise.ufl.edu/research/sparse/matrices

[5] S. Dalton, N. Bell, L. Olson, and M. Garland, "Cusp: Generic parallel algorithms for sparse matrix and graph computations," 2014, version 0.5.1. [Online]. Available: http://cusplibrary.github.io/