Spla: Generalized Sparse Linear Algebra Library with Vendor-Agnostic GPUs Acceleration

1st Egor Orachev
St. Petersburg State University
St. Petersburg, Russia
egor.orachev@gmail.com
0000-0002-0424-4059

2nd Semyon Grigorev
St. Petersburg State University
St. Petersburg, Russia
s.v.grigoriev@spbu.ru
0000-0002-7966-0698

***Abstract*—Scalable high-performance graph analysis is an actual nontrivial challenge. Usage of sparse linear algebra operations as building blocks for graph analysis algorithms, which is a core idea of GraphBLAS standard, is a promising way to attack it. While it is known that sparse linear algebra operations can be efficiently implemented on GPU, full GraphBLAS implementation on GPU is a nontrivial task that is almost solved by GraphBLAST project. Though it is shown that utilization of GPUs for GraphBLAS implementation significantly improves performance, portability problem is not solved yet: GraphBLAST uses Nvidia Cuda stack. In this work we propose Spla library which aims to solve this problem using OpenCL API for vendor-agnostic GPUs computations. Evaluation shows that the proposed solution demonstrates performance comparable with GraphBLAST, outperforming it up to 36 times in some cases, remaining portable across different GPUs vendors. However there is still a space for further optimizations.**

***Index Terms*—graphs, algorithms, graph analysis, sparse linear algebra, GraphBLAS, GPGPU, OpenCL**

I. Introduction

Scalable high-performance graph analysis is an actual challenge. CuSha [1], Gunrock [2] show that utilization of GPUs can improve the performance of graph analysis. But low flexibility and high complexity of API are problems of these solutions. The promising thing which provides a user-friendly API is a linear algebra based graph programmin model, formalized in GraphBLAS [3]. While reference CPU-based implementation of this API, SuiteSparse [4], demonstrates good performance in real-world tasks, GPU-based implementation is challenging due to required operations generalization, data sparsity and hardware programming complexity. Well-known libraries as cuSPARSE, clSPARSE, CUSP cannot be reused, since they are almost all specified for operations over floats. GraphBLAST [5], GBTL [6] show promising GPU performance of GraphBLAS-based graph analysis solutions. But these solutions are not portable because they are based on Nvidia Cuda. In order to address this problem we developed the GraphBLAS-inspired *Spla* library[1], which features portable OpenCL-based acceleration and shows performance comparable with GraphBLAST, achieving up to 36 times speedup in some cases.

[1]Source code of Spla library available at: https://anonymous.link

II. Proposed Solution

The proposed Spla library implemented in C++17 language, using vendor-agnostic OpenCL 1.2 for GPU specific computations. The OpenCL is chosen over Cuda and SyCL technologies. While Cuda is specific only for Nvidia devices, SyCL is a promising API, but it is too high-level and its support is still debatable. The implementation of linear algebra operations is based on Yang et al. [5], [7], [8] works. Common graph algorithms, such as BFS, SSSP, PR and TC implemented with respect to a GraphBLAST. Spla features a number of optimizations, such as push-pull, masking, early exit, custom small memory allocation, and sparse-dense storage switch.

III. Evaluation

For performance analysis of the proposed solution, we evaluated a few most common graph algorithms using real-world sparse matrix data. As a baseline for comparison we chose LAGraph [9] in connection with SuiteSparse [4] as a multi-core CPU tool, Gunrock [2] and GraphBLAST [5] as a Nvidia GPU tools. Also, we tested algorithms on several devices with distinct OpenCL vendors in order to validate the portability of the proposed solution.

A. Evaluation Setup

We use a PC with Ubuntu 20.04 installed, which has 3.40Hz Intel Core i7-6700 4-core CPU, DDR4 64Gb RAM, either Nvidia GeForce GTX 1070 8Gb VRAM, Intel Arc A770 flux 8GB VRAM, or AMD Radeon Vega Frontier Edition, 16GB VRAM. Programs were compiled with GCC v9.4. Programs using CUDA were compiled with GCC v8.4 and Nvidia NVCC v10.1. Data loading time, preparation, format transformations, and host-device initial communications are excluded from time measurements. All tests are averaged across 10 runs. The deviation of measurements does not exceed the threshold of 10 percent. Additional warm-up run is excluded from measurements. The 1st graph vertex is initial node in traversal algorithms.

Thirteen matrices with graph data were selected from the Sparse Matrix Collection at University of Florida [10]. Information is summarized in Table I. The dataset is converted to undirected graphs. Self-loops and duplicated edges are

TABLE I
DATASET DESCRIPTION.

Graph	Vertices	Edges	Out Degree		
			Avg	Sd	Max
coAuthorsCit	227.3K	1.6M	7.2	10.6	1.4K
coPapersDBLP	540.5K	30.5M	56.4	66.2	3.3K
amazon2008	735.3K	7.0M	9.6	7.6	1.1K
hollywood2009	1.1M	112.8M	98.9	271.9	11.5K
comOrkut	3.1M	234.4M	76.3	154.8	33.3K
citPatents	3.8M	33.0M	8.8	10.5	793.0
socLiveJournal	4.8M	85.7M	17.7	52.0	20.3K
indochina2004	7.4M	302.0M	40.7	329.6	256.4K
belgiumosm	1.4M	3.1M	2.2	0.5	10.0
roadNetCA	2.0M	5.5M	2.8	1.0	12.0
rggn222s0	4.2M	60.7M	14.5	3.8	36.0
rggn223s0	8.4M	127.0M	15.1	3.9	40.0
roadcentral	14.1M	33.9M	2.4	0.9	8.0

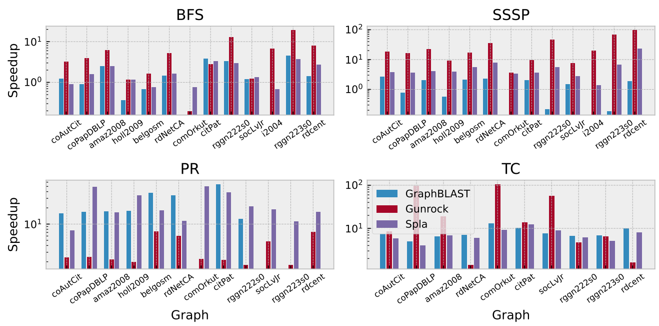

Fig. 1. Performance of Spla library and GPU tools on the same device, shown as a speedup relative to LaGraph. Logarithmic scale is used.

removed. Weights generated using uniform distribution $[0, 1]$ of floating-point values.

B. Results Summary

RQ1. *What is the performance of the proposed solution relative to existing tools for GPU analysis?* Taking a look at Fig. 1, Spla shows very acceptable performance in all algorithms, running with comparable speed to its nearest competitor, GraphBLAST. Also proposed library does not suffer from memory issues on some large graphs. Spla is consistently several times faster than LaGraph, overcoming it up to 25× in some cases. Gunrock is the fastest GPU framework for analysis. It dominates the overall performance and only suffers in a PR algorithm.

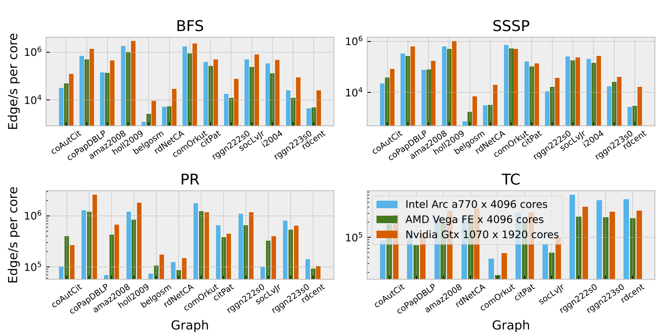

Fig. 2. Performance of Spla library on different devices, shwon as edge/s througput per device core. Logarithmic scale is used.

RQ2. *What is the performance of the proposed solution on various devices vendors and OpenCL runtimes?* Spla successfully launches and workes on the GPU of distinct vendors, including Intel, AMD and Nvidia. It shows promising performance and demonstrated scalability in relation to the number of computing cores. Fig. 2 depicts the edge/s throughput per a GPU core for all devices. This metric is quite predictable for the same graphs. This can be seen if one takes into account the overall shape of the figures for BFS, SSSP and PR as a whole.

IV. Conclusion

We presented Spla, generalized sparse linear algebra library with vendor-agnostic GPUs accelerated computations. The evaluation of the proposed solutions for some real-world graph data in four different algorithms shows, that OpenCL-based solution has a promising performance, comparable to analogs, has acceptable scalability on devices of different GPU vendors. There is still a plenty of research questions and directions for improvement, such as further workload balance, performance tuning [11], graph streaming [1] and multi-GPU support [2].

References

[1] F. Khorasani, K. Vora, R. Gupta, and L. N. Bhuyan, "Cusha: Vertex-centric graph processing on gpus," in *Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 239–252. [Online]. Available: https://doi.org/10.1145/2600212.2600227

[2] Y. Pan, Y. Wang, Y. Wu, C. Yang, and J. D. Owens, "Multi-gpu graph analytics," in *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2017, pp. 479–490.

[3] J. Kepner, P. Aaltonen, D. Bader, A. Buluç, F. Franchetti, J. Gilbert, D. Hutchison, M. Kumar, A. Lumsdaine, H. Meyerhenke, S. McMillan, C. Yang, J. D. Owens, M. Zalewski, T. Mattson, and J. Moreira, "Mathematical foundations of the graphblas," in *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, 2016, pp. 1–9.

[4] T. A. Davis, "Algorithm 1000: Suitesparse:graphblas: Graph algorithms in the language of sparse linear algebra," *ACM Trans. Math. Softw.*, vol. 45, no. 4, Dec. 2019. [Online]. Available: https://doi.org/10.1145/3322125

[5] C. Yang, A. Buluc, and J. D. Owens, "Graphblast: A high-performance linear algebra-based graph framework on the gpu," 2019.

[6] P. Zhang, M. Zalewski, A. Lumsdaine, S. Misurda, and S. McMillan, "Gbtl-cuda: Graph algorithms and primitives for gpus," in *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2016, pp. 912–920.

[7] C. Yang, Y. Wang, and J. D. Owens, "Fast sparse matrix and sparse vector multiplication algorithm on the gpu," in *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, 2015, pp. 841–847.

[8] C. Yang, A. Buluc, and J. D. Owens, "Implementing push-pull efficiently in graphblas," 2018. [Online]. Available: https://arxiv.org/abs/1804.03327

[9] G. Szárnyas, D. A. Bader, T. A. Davis, J. Kitchen, T. G. Mattson, S. McMillan, and E. Welch, "Lagraph: Linear algebra, network analysis libraries, and the study of graph algorithms," 2021.

[10] T. A. Davis and Y. Hu, "The university of florida sparse matrix collection," *ACM Trans. Math. Softw.*, vol. 38, no. 1, dec 2011. [Online]. Available: https://doi.org/10.1145/2049662.2049663

[11] Y. Nagasaka, A. Nukada, and S. Matsuoka, "High-performance and memory-saving sparse general matrix-matrix multiplication for nvidia pascal gpu," in *2017 46th International Conference on Parallel Processing (ICPP)*, 2017, pp. 101–110.