Linear Algebra Based Graph Analysis on RISC-V GPGPU Vortex

Shaehmitov Shamil
Software Engineering Chair
Saint Petersburg State University
Saint Petersburg, Russia
samilsaehmitov@gmail.com

Vladimir Kutuev
Software Engineering Chair
Saint Petersburg State University
Saint Petersburg, Russia
https://orcid.org/0000-0001-7749-4940

Semyon Grigorev
Software Engineering Chair
Saint Petersburg State University
Saint Petersburg, Russia
https://orcid.org/0000-0002-7966-0698

***Abstract*—In this work we evaluate Spla—sparse linear algebra based library for graph analysis—on RISC-V ISA based open source GPGPU Vortex. We show that !!!**

***Index Terms*—GraphBLAS, Sparse Linear Algebra, Graph Analysis, GPGPU, RISC-V**

I. Introduction

Sparse linear algebra has emerged as a powerful paradigm for high-performance graph analysis. A wide range of problems—from graph traversing to clustering—can be reduced to efficient algebraic operations over matrices and vectors. GraphBLAS API [1] follows this idea and defines a standardized set of building blocks: sparse matrices and vectors, algebraic structures like monoids and semirings, and fundamental operations such as matrix-matrix multiplication. GraphBLAS is specifically designed to serve as a foundational layer for the development of scalable, linear-algebra-based graph algorithms.

While highly tuned CPU implementations of GraphBLAS—most notably SuiteSparse:GraphBLAS[1] [2]—deliver strong performance on multi-core systems, implementing the GraphBLAS API efficiently on general-purpose graphics processing units (GPGPUs) remains a significant challenge. While GPGPUs is a promising platform for linear algebra based computations, they introduce well-known obstacles for sparse workloads, including irregular memory access patterns and load imbalance. Additionally, creating generalized kernels capable of operating not only on primitive data types like floats or integers but also on user-defined custom types presents a nontrivial engineering task.

Despite these challenges, several efforts have been made to create GPU-accelerated libraries for linear-algebra-based graph analysis, such as GraphBLAST[2] [3] which uses CUDA and the portable Spla[3] [4] library which uses OpenCL.

In parallel, the rise of open instruction set architectures (ISAs), most notably RISC-V, is expanding the hardware landscape. Recent work has explored the potential of RISC-V-based CPUs for graph analysis [5]–[7], including designs leveraging vector extensions [8]. Beyond CPUs, specialized accelerators—including RISC-V-based GPGPUs—are now emerging. One actively developed example is the Vortex platform, a RISC-V-based GPGPU that has been evaluated not only for graphics but also for scientific computing [10] and graph analysis [9], [11]. However, its applicability to linear-algebra-based graph analysis remains unexamined.

In this paper, we evaluate the suitability of the Vortex architecture for linear-algebra-based graph analysis. Specifically, we examine the performance scaling of the Spla library on this platform. Our evaluation using a cycle-approximate simulator shows that [Results to be inserted here].

II. Spla Graph Analysis Library

Spla [4] is a GPGPU-accelerated, GraphBLAS-inspired library for graph analysis. It is based on sparse linear algebra and uses OpenCL to offload linear algebra kernels to appropriate devices, including GPGPUs. Using OpenCL makes the library vendor-agnostic: it has been shown in [4] that Spla performs and scales well across GPUs from different vendors including AMD, Intel, and Nvida.

The library implements several classical graph analysis algorithms, including canonical single-source level BFS, triangle counting (TC), single-source shortest path (SSSP), and PageRank.

III. RISC-V GPGPU Vortex

Vortex[4] [12] is an open-source RISC-V-based GPGPU. It supports OpenCL programming via the POCL compiler[5] [13]. Additionally, it is designed for FPGAs equipped with high-bandwidth memory (HBM), which is advantageous for graph processing.

The high-level architecture of the Vortex processor[6] is shown in Fig. 1. The processor consists of *clusters*, which may share an optional L_3 cache. Each *cluster* contains multiple *sockets*, which may share an optional L_2 cache. *Sockets* consist of cores with shared L_1 cache, and each core hosts multiple

Identify applicable funding agency here. If none, delete this.

[1]Source code of SuiteSparse:GraphBLAS on GitHub: https://github.com/DrTimothyAldenDavis/GraphBLAS

[2]GraphBLAST project page: https://github.com/gunrock/graphblast

[3]Spla project page: https://github.com/SparseLinearAlgebra/spla

[4]https://github.com/vortexgpgpu/vortex

[5]Portable Computing Language project: https://portablecl.org/

[6]Detailed architectural information is available at https://github.com/vortexgpgpu/vortex/blob/master/docs/microarchitecture.md.

threads. Threads share local memory and are logically grouped into warps.

The design is flexibly configurable: the numbers of clusters, cores, threads, and warps in the target processor can be specified, and the L_3 and L_2 caches can be independently enabled or disabled. Number of sockets calculated automatically such that socket size is a minimum of 4 and number of cores.

The Vortex design is distributed with SimX, a cycle-level functional simulator. A cycle-accurate RTL simulation is also available. Although the A extension (atomics instructions)[7] is declared, atomic operations are currently supported only in the SimX simulator and not in the RTL implementation.

IV. Evaluation

The goal of this evaluation is to assess the performance scaling of Spla on Vortex. Due to limitations in atomic operation support within the RTL implementation, all experiments were performed using the SimX functional simulator.

A. Environment

Initial testing revealed issues with floating-point operations, which produced incorrect results for some hardware configurations. Consequently, we limited subsequent experiments to Breadth-First Search (BFS) and Triangle Counting (TC), excluding Single-Source Shortest Path (SSSP) and PageRank. To keep simulation times manageable, we used a single graph from the SuiteSparse matrix collection[8]: soc-Epinions1, with 75 888 vertices and 508 837 edges.

We conducted two series of experiments. The first varies the number of warps and threads per warp while keeping the number of clusters and cores fixed (at 2 and 4, respectively), with the goal of selecting the best core configuration while preserving multi-core execution to account for cache effects. The second series, using the best configuration identified in the first step, varies the number of clusters and cores per cluster to assess scaling at the core and cluster levels. Cache sizes were set to their default values: 16 KB for L_1, 1 MB for L_2, and 2 MB for L_3.

We use the number of cycles reported by SimX as a performance metric. For multi-core configurations, we report the maximum cycle count across all cores. During the experiments, we encountered unexpected behavior in SimX that led to out-of-memory exceptions. Therefore, some data points are missing from the graphs below.

B. Results

In figures 2 and 3

Best configuration for BFS is 2 warps, 8 threads per warp (16 threads total). Best configuration for TC is 4 warps, 16 threads per warp (64 threads total).

Edges per core on cycle. Compare with Spla on other GPUs.

[7]Supported RISC-V profiles are RV32IMAF and RV64IMAFD (https://github.com/vortexgpgpu/vortex?tab=readme-ov-file#specifications)

[8]A diverse collection of sparse matrices from various domains: http://sparse.tamu.edu/

C. Scaling limitations analysis

To analyze the reasons for limited scaling as the number of threads increases, we measured the average utilization of the ALU and LSU, in terms of stall cycles, for the best BFS configuration. The results are presented in Fig. 5 and Fig. 6, respectively. The data indicate that the LSU is the performance bottleneck within the core.

The same bottleneck was observed in the scaling analysis across clusters and cores. Whether increasing cache sizes can alleviate this problem remains a question for future research. We anticipate that careful cache size tuning may help identify a more efficient configuration.

V. Conclusion

In this work we evaluated Spla—linear-algebra-based graph analysis library—on RISC-V IAS based GPGPU Vortex. We show that Spla is portable enough to be run on Vortex. Vortex ready to run. Scaling. !!!!!

Several directions remain for future work. First, it is necessary to resolve issues with floating-point operations and to conduct experiments with additional algorithms such as PageRank and single-source shortest path (SSSP). Furthermore, evaluating Spla on Vortex across a broader set of diverse graphs would yield more robust scaling insights.

After simulation-based evaluation in SimX, the next step is to estimate the FPGA resources required for the most performant configuration and to perform actual evaluations on different FPGA platforms, when atomics support will be finished.

Another promising direction is analyzing the applicability of SparseWeaver to sparse-linear-algebra-based graph analysis. Prior work has shown that SparseWeaver improves the performance of manually crafted graph algorithms and has suggested its potential for optimizing sparse linear algebra kernels [11].

We also plan to evaluate Spla on Ventus [9] [14], another RISC-V-based GPGPU, and to compare its performance with Vortex.

Applicability of open-source GPUs such as Vortex or Ventus as a foundation for specialized processors dedicated to sparse-linear-algebra-based graph analysis is a goal for future research.

References

[1] B. Brock, A. Buluç, T. G. Mattson, S. McMillan, and J. E. Moreira, "Introduction to graphblas 2.0," in *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2021, pp. 253–262.

[2] T. A. Davis, "Algorithm 1037: Suitesparse:graphblas: Parallel graph algorithms in the language of sparse linear algebra," *ACM Trans. Math. Softw.*, vol. 49, no. 3, Sep. 2023. [Online]. Available: https://doi.org/10.1145/3577195

[3] C. Yang, A. Buluç, and J. D. Owens, "Graphblast: A high-performance linear algebra-based graph framework on the gpu," *ACM Trans. Math. Softw.*, vol. 48, no. 1, Feb. 2022. [Online]. Available: https://doi.org/10.1145/3466795

[9]Ventus GPU project page: https://github.com/THU-DSP-LAB/ventus-gpgpu

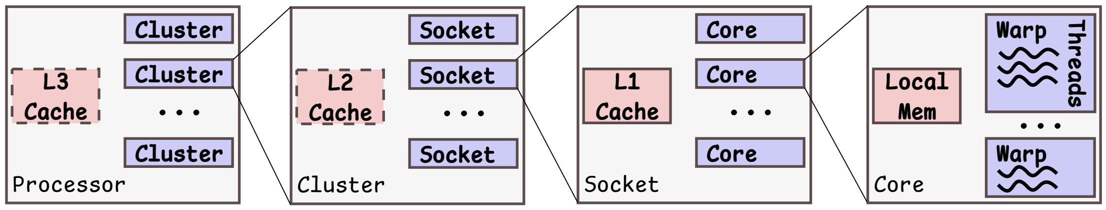

Fig. 1. Vortex architecture

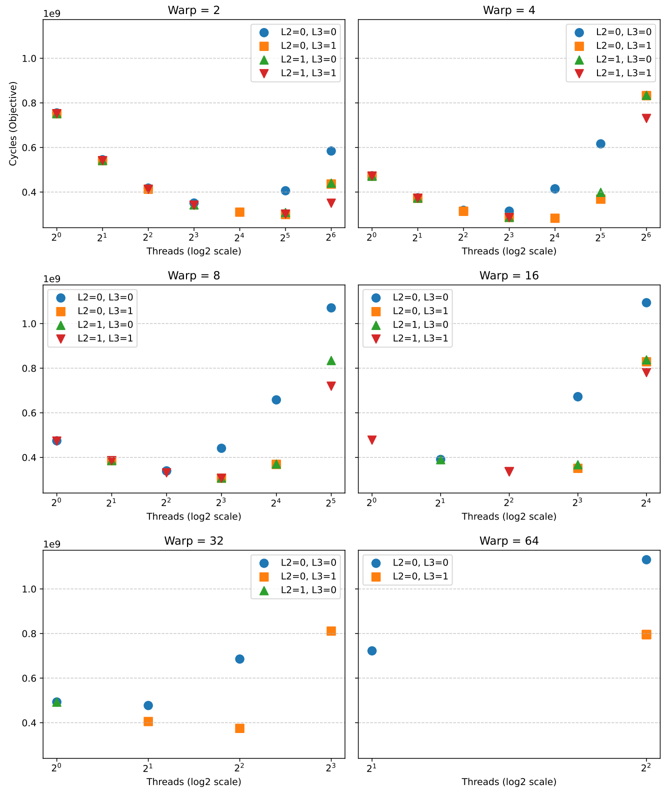

Fig. 2. Scaling analysis of triangle counting for varying numbers of warps and threads per warp

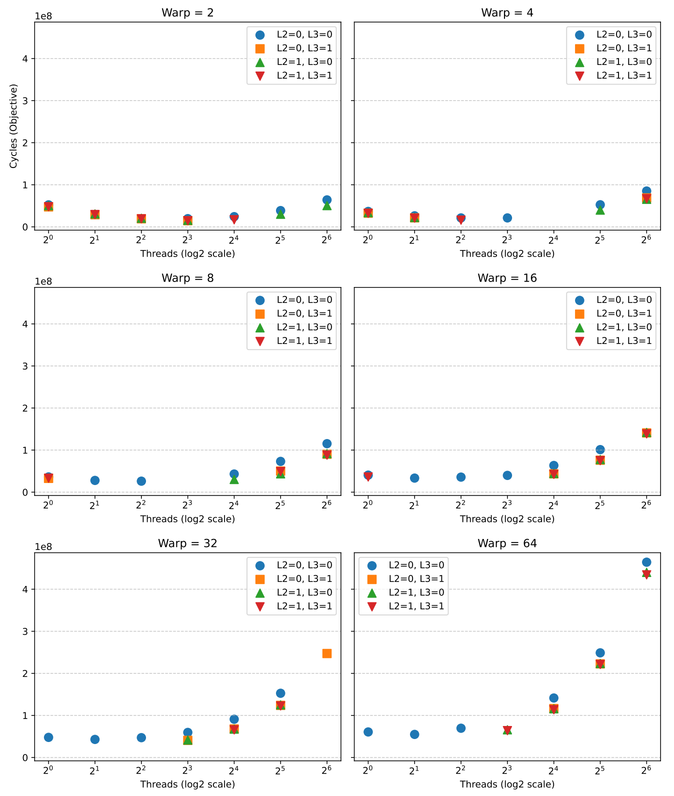

Fig. 3. Scaling analysis of BFS for varying numbers of warps and threads per warp

[4] E. Orachev, "Generalized sparse linear algebra library with vendor-agnostic gpus acceleration," Master's thesis, Saint Petersburg State University, 2023.

[5] A. M. Ravikumar, A. Vinay, K. K. Nagar, and M. Purnaprajna, "Parallel graph algorithms on a RISCV-based many-core," *Int. J. Reconfigurable Embed. Syst. (IJRES)*, vol. 14, no. 3, p. 843, Nov. 2025.

[6] K. Zhou, J. Deng, and Y. Zeng, "Design and memory access optimization of graph processing processor design based on risc-v," in *Proceedings of the 2023 6th International Conference on Artificial Intelligence and Pattern Recognition*, ser. AIPR '23. New York, NY, USA: Association for Computing Machinery, 2024, p. 576–583. [Online]. Available: https://doi.org/10.1145/3641584.3641670

[7] M. S. Yenimol, "Hardware/software co-design of domain-specific risc-v processor for graph applications," 2022. [Online]. Available: http://hdl.handle.net/11693/80659

[8] P. Vizcaino, J. Labarta, and F. Mantovani, "Graph computing on long vector architectures (yes, it works!)," in *2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2024, pp. 986–995.

[9] N. Shah and M. Verhelst, "Graph analytics on risc-v gpu: Where are

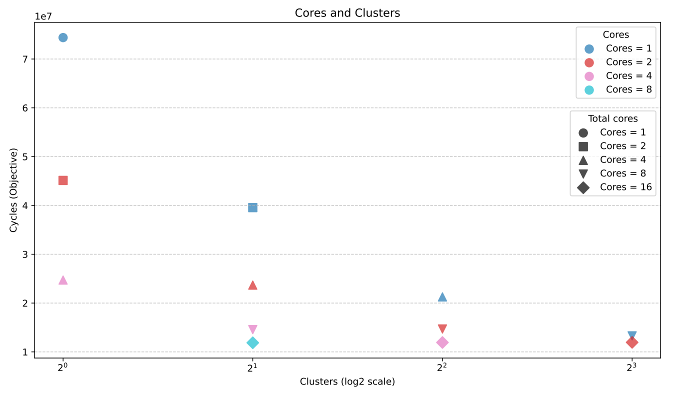

Fig. 4. Scaling analysis of BFS for varying numbers of clusters and cores per cluster

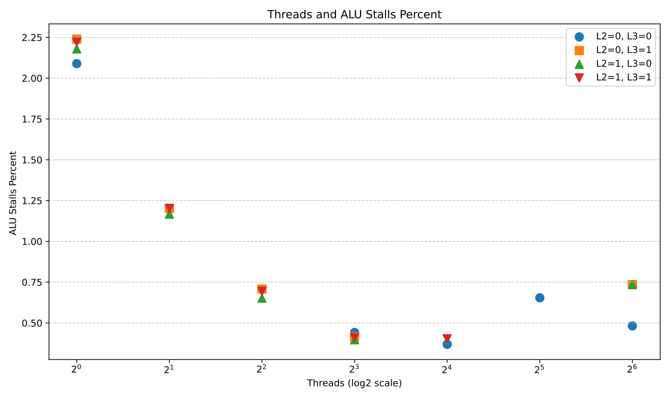

Fig. 5. ALU stalls on BFS for the best configuration

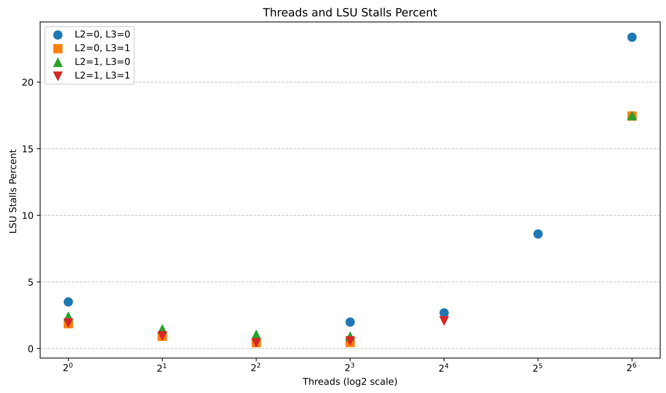

Fig. 6. LSU stalls on BFS for the best configuration

the bottlenecks?" in *Spring 2022 RISC-V Week, Location: Paris, France*, 2022.

[10] E. Guthmuller, J. Fereyre, and D. Herrera-Marti, "GPGPUs on FPGAs: A competitive approach for scientific computing ?" in *DATE 2025 - Design, Automation and Test in Europe Conference*, Lyon, France, Mar. 2025. [Online]. Available: https://cea.hal.science/cea-05043041

[11] S. Jeong, L. P. Cooper, J. M. Lee, H. Choi, N. Parnenzini, C. Ahn, Y. Lee, H. Kim, and H. Kim, "Sparseweaver: Converting sparse operations as dense operations on gpus for graph workloads," in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2025, pp. 1437–1451.

[12] B. Tine, K. P. Yalamarthy, F. Elsabbagh, and K. Hyesoon, "Vortex: Extending the risc-v isa for gpgpu and 3d-graphics," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 754–766. [Online]. Available: https://doi.org/10.1145/3466752.3480128

[13] P. Jääskeläinen, C. S. Lama, E. Schnetter, K. Raiskila, J. Takala, and H. Berg, "pocl: A performance-portable opencl implementation," *Int. J. Parallel Program.*, vol. 43, no. 5, p. 752–785, Oct. 2015. [Online]. Available: https://doi.org/10.1007/s10766-014-0320-y

[14] J. Li, K. Yang, C. Jin, X. Liu, Z. Yang, F. Yu, Y. Shi, M. Ma, L. Kong, J. Zhou, H. Wu, and H. He, "Ventus: A high-performance open-source gpgpu based on risc-v and its vector extension," in *2024 IEEE 42nd International Conference on Computer Design (ICCD)*, 2024, pp. 276–279.