APPLIED RESEARCH LABORATORY FOR
INTELLIGENCE AND SECURITY

Technical Report

Benchmarking Hardware for Accelerating Intelligence and Security Graph Problems

Kent O'Sullivan, University of Maryland, osullik@umd.edu
Amir Ghaemi, Applied Research Laboratory for Intelligence and Security, aghaemi@arlis.umd.edu
Nandini Ramachandran, University of Maryland, nandinir@umd.edu
Vladimir Rife, Applied Research Laboratory for Intelligence and Security, vrife@umd.edu
William Regli, University of Maryland, regli@umd.edu

Project Github: https://github.com/UMD-ARLIS/Graph-Benchmarking-Project
ReadTheDocs: https://graph-benchmarking.readthedocs.io/en/latest/

December 18, 2023

Abstract

1 Introduction

Purpose. The purpose of this technical report is to describe the design and implementation of the *ARLIS Graph Hardware Acceleration Benchmark* (AGHAB).

Motivation. A recent analysis of the computing requirements of the United States (US) intelligence and Security (I&S) community shows that many of the core analytic problems are formulated as graph processing problems [1]. Formally, a Graph $\mathcal{G}$ is a data structure with a set of nodes (vertexes) V and a set of connections between those nodes (edges) E such that $\mathcal{G} = \{V, E\}$. A common problem that the I&S community may use graphs to model is a social network graph, where each vertex is a person, and each node represents a social relationship between the two people (nodes) it connects. Intelligence graph datasets like social network graphs quickly grow to billions of edges. The worst-case performance of many graph processing algorithms is quadratic in the number of edges, meaning that the larger a graph becomes, the slower it is to process it.

In an environment where collection streams are constant, a *streaming graph* is continuously updated with new information. Additionally, to maintain compliance with legislative requirements, collected information may need to be purged after some time has passed. With new data changing the shape of a graph, and the requirement to delete data, the graph analysis becomes temporally bound, so timely graph processing of streaming graphs is critical.

The traditional approach to reduce the time required to compute large datasets is to parallelize them. Graph algorithms are difficult to parallelize, and in particular, are poorly suited for the distributed parallelism favoured for processing very large datasets because of the difficulty of selecting partition points, and the lack of spatial locality in graph data structures [2]. Shared memory parallelism approaches have been shown as viable on very large graphs for a wide collection of algorithms [3], but still perform suboptimally compared to other data structures of similar size because of their irregular structure, which manifests as memory-boundedness. That is, operations on graphs themselves are computationally simple, but because the information being processed is not necessarily proximal in memory traditional optimization approaches like caching and branch prediction are ineffective [4]. If caching is problematic, it follows that a processor that makes limited use of caching like a Graphic Processing Unit (GPU) should be a better choice for graph processing. However, GPUs are poorly suited for many graph problems because of the *sparsity* of graphs. *Sparsity* here means that when the graph is stored in memory (particularly as an adjacency matrix) there is a lot of 'empty space'. GPUs in particular are optimized for dense-matrix multiplication operations, so processing these graphs with a lot of 'empty space' means the matrices that represent them are sparse, not dense and so have a lot of wasted computation. The GUNROCK approach in particular is notable in GPU-based graph processing [5] but still suffers from resource under-utilization compared to operations on natively dense data structures, like images.

If the representation of graphs is the issue degrading performance, changing that representation to a more efficient one appears wise. The Graph Basic Linear Algebra Subprogram (GraphBLAS) project is an effort to

evolve the representation of Graphs and implementations of graph processing algorithms to leverage linear algebraic representations and primitive operations to improve the consistency, effectiveness and efficiency of graph processing [6]–[10]. Benchmarking of GraphBLAS against other approaches has shown it to execute slower than other approaches, likely due to the computational cost of the abstraction layer that makes it such an attractive option [11].

Related work comparing the original BLAS performance across different hardware systems suggests that Field Programmable Gate Arrays (FPGA) and Application Specific Integrated Circuits (ASIC) can significantly accelerate linear algebraic operations [12], [13].

Exploring how to accelerate the processing of these large streaming graphs is one of the core goals of the Defense Advanced Research Projects Agency (DARPA) Hierarchical Identify Verify Exploit (HIVE) program, and relates to work undertaken in the DARPA Software Defined Hardware (SDH) program [14]. Many emerging hardware architectures optimized for graph processing are emerging, like EMU [15], Graphicianado [16], the Lincoln Labs architecture [17] and PiUMA [4] among others [18] will be similarly limited.

The proliferation of new hardware options presents an imperative for those acquiring and integrating graph hardware accelerators to select the best processor for the task at hand. There is currently no benchmark designed to enable the comparison of relevant I&S algorithms across hardware architectures, on graph problems that reflect the real-world problems that they will be expected to process. The AGHAB aims to fill the gap.

The remainder of this report outlines the architecture of the AGHAB (section 2), describes the benchmarks, reference implementations and telemetry in sections 3, 4 and 5 before providing an example usage of the benchmark (section 6) and summarizing contribution guidelines in section 7.

2 Benchmark Architecture

3 Benchmark Data

4 Reference Implementations

4.1 Requirements

4.2 Problem Domains

4.2.1 Community Detection

4.2.2 Subgraph Matching

4.3 Reference Implementations

4.3.1 Louvain Algorithm

4.3.2 VF3

5 Telemetry

6 Benchmarking Example

7 Contributing

8 References

[1] W. Regli, P. Loats, L. Miller-Sims, *et al.*, "Operationalizing emerging hardware for ai applications: A survey of transition opportunities and datasets," 2022.

[2] S. Beamer, K. Asanovic, and D. Patterson, "Locality exists in graph processing: Workload characterization on an ivy bridge server," in *2015 IEEE International Symposium on Workload Characterization*, IEEE, 2015, pp. 56–65.

[3] L. Dhulipala, G. E. Blelloch, and J. Shun, "Theoretically efficient parallel graph algorithms can be fast and scalable," *ACM Transactions on Parallel Computing (TOPC)*, vol. 8, no. 1, pp. 1–70, 2021.

[4] S. Aananthakrishnan, N. K. Ahmed, V. Cave, *et al.*, "Piuma: Programmable integrated unified memory architecture," *arXiv preprint arXiv:2010.06277*, 2020. [Online]. Available: `https://arxiv.org/pdf/2010.06277`.

[5] Y. Wang, A. Davidson, Y. Pan, Y. Wu, A. Riffel, and J. D. Owens, "Gunrock: A high-performance graph processing library on the gpu," in *Proceedings of the 21st ACM SIGPLAN symposium on principles and practice of parallel programming*, 2016, pp. 1–12.

[6] T. Mattson, D. Bader, J. Berry, *et al.*, "Standards for graph algorithm primitives," in *2013 IEEE High Performance Extreme Computing Conference (HPEC)*, IEEE, 2013, pp. 1–2.

[7] T. Mattson, T. A. Davis, M. Kumar, *et al.*, "Lagraph: A community effort to collect graph algorithms built on top of the graphblas," in *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, IEEE, 2019, pp. 276–284.

[8] J. Kepner, D. Bader, A. Buluç, J. Gilbert, T. Mattson, and H. Meyerhenke, "Graphs, matrices, and the graphblas: Seven good reasons," *Procedia Computer Science*, vol. 51, pp. 2453–2462, 2015.

[9] J. Kepner, P. Aaltonen, D. Bader, *et al.*, "Mathematical foundations of the graphblas," in *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, IEEE, 2016, pp. 1–9.

[10] 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)*, IEEE, 2021, pp. 253–262.

[11] A. Azad, M. M. Aznaveh, S. Beamer, *et al.*, "Evaluation of graph analytics frameworks using the gap benchmark suite," in *2020 IEEE International Symposium on Workload Characterization (IISWC)*, IEEE, 2020, pp. 216–227.

[12] S. Kestur, J. D. Davis, and O. Williams, "Blas comparison on fpga, cpu and gpu," in *2010 IEEE computer society annual symposium on VLSI*, IEEE, 2010, pp. 288–293.

[13] C. Xiong and N. Xu, "Performance comparison of blas on cpu, gpu and fpga," in *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, IEEE, vol. 9, 2020, pp. 193–197.

[14] W. Regli, B. Johnson, L. Miller-Sims, A. Keshavarzi, and M. Teli, "Software-defined hardware:a study of emerging hardware for ai applications," 2022.

[15] T. Dysart, P. Kogge, M. Deneroff, *et al.*, "Highly scalable near memory processing with migrating threads on the emu system architecture," in *2016 6th Workshop on Irregular Applications: Architecture and Algorithms (IA3)*, IEEE, 2016, pp. 2–9.

[16] T. J. Ham, L. Wu, N. Sundaram, N. Satish, and M. Martonosi, "Graphicionado: A high-performance and energy-efficient accelerator for graph analytics," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, IEEE, 2016, pp. 1–13.

[17] W. S. Song, V. Gleyzer, A. Lomakin, and J. Kepner, "Novel graph processor architecture, prototype system, and results," in *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, IEEE, 2016, pp. 1–7.

[18] B. Peccerillo, M. Mannino, A. Mondelli, and S. Bartolini, "A survey on hardware accelerators: Taxonomy, trends, challenges, and perspectives," *Journal of Systems Architecture*, vol. 129, p. 102 561, 2022.

9 Glossary

ASIC Application Specific Integrated Circuits
ARLIS Applied Research Laboratory for Intelligence and Security.
BFS Breadth-First Search
CPU Central Processing Unit
DARPA Defense Advanced Research Projects Agency
FPGA Field Programmable Gate Array
GPU Graphics Processing Unit
HIVE Hierarchical Identify Verify Exploit
KGA Knowledge Graph Analytics
PIUMA Programmable Integrated Unified Memory Architecture
SGM Sub Graph Matching
TEPS Traversed Edges Per Second
TEPS/W Traversed Edges Per Second Per Watt