LAGraph: A Graph Algorithm and Network Analysis Library for GraphBLAS

Timothy A. Davis*, Timothy G. Mattson†, James Kitchen‡, Scott McMillan§, Gábor Szárnyas¶, Erik Welch‡
*Texas A&M University †Intel Corporation ‡Anaconda, Inc. §Carnegie Mellon University ¶CWI Amsterdam

***Abstract*—design decisions, easy mode/expert mode, GAP algorithms, ...**

***Index Terms*—Graph Processing, Graph Algorithms, Graph Analytics, Linear Algebra, GraphBLAS**

I. Introduction

LAGraph is a library of Graph Algorithms based on the GraphBLAS

Key contributions:

- document design decisions for LAGraph
- present a concise notation for GraphBLAS algorihms
- algorithms of the GAP benchmark suite [3] used in the IISWC benchmark paper [2]
- improve data ingestion performance, e.g. using SIMD techniques [10]

Recently, numerous graph-specific have targeted GPUs such as Gunrock [15] and GraphBLAST [17], and FPGAs [4].

However, in the near future we expect even more heterogeneous hardware architectures including graph-specific hardware based on the Programmable Integrated Unified Memory Architecture (PIUMA) [1]. Additionally, graph processing workloads can be offloaded to machine learning accelerators, e.g., Tensor Processing Units (TPUs) [9], systolic arrays using reconfigurable dataflow architecture [7], sparse linear algebra-based deep learning accelerators [11].

Previous GraphBLAS design papers: theory [12], C API [14], C++ API [6], distributed API [5], LAGraph [13]

```
int main() {
    return 0; // return zero
}
```

Listing 1: Example

[1]

II. Design Decisions

Jim

We investigate the following design questions:

- easy/expert mode or standard/advanced more
- multi-threadable
- data structure for representing a graph/matrix
- error handling
- opacity of LAGraph

[1]A non peer-reviewed comparison of 6 popular graph algorithms libraries is available at https://www.timlrx.com/blog/benchmark-of-popular-graph-network-packages-v2.

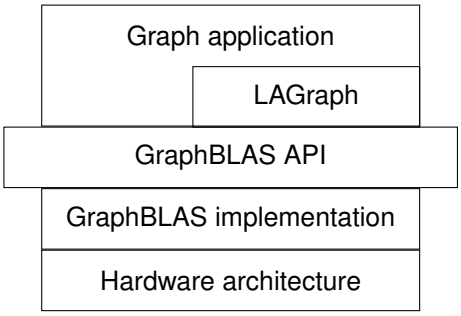

Fig. 1: Separation of concerns using the GraphBLAS API.

- how to work around typecasting being expensive in GraphBLAS
- terminology: properties (parameters? features?)

III. Notation

IV. Algorithms

GAP algorithms: BFS, SSSP, TC, BC, PR. Not sure whether CC should be included.

A. BFS

push/pull [16]

B. Betweenness Centrality

also does push/pull

C. PageRank

D. SSSP

E. Triangle Count

(Not too interesting from an API design point of view.)

F. Connected Components

Needs a few GxB extensions.

V. Evaluation

Tim D

A. SuiteSparse Extensions

SuiteSparse performance figures: lazy sort, bitmap, ...

B. Performance Results

Benchmark environment (DGX), 2x6x5

VI. Utility Fuctions

sort, diag/invdiag (?), equal

VII. Conclusion

Future work - ideas:

- Create a Python wrapper for LAGraph
- Implement the LDBC Graphalytics benchmark [8]
- Improve data ingestion performance using e.g., SIMD instructions [10]

Acknowledgements

add acks

G. Szárnyas was partially supported by the SQIREL-GRAPHS NWO project.

REFERENCES

[1] S. Aananthakrishnan *et al.*, "PIUMA: Programmable Integrated Unified Memory Architecture," *CoRR*, vol. abs/2010.06277, 2020. [Online]. Available: https://arxiv.org/abs/2010.06277

[2] A. Azad *et al.*, "Evaluation of graph analytics frameworks using the GAP Benchmark Suite," in *IEEE*. IEEE, 2020, pp. 216–227. [Online]. Available: https://doi.org/10.1109/IISWC50251.2020.00029

[3] S. Beamer *et al.*, "The GAP Benchmark Suite," *CoRR*, vol. abs/1508.03619, 2015. [Online]. Available: http://arxiv.org/abs/1508.03619

[4] M. Besta, D. Stanojevic, J. de Fine Licht, T. Ben-Nun, and T. Hoefler, "Graph processing on FPGAs: Taxonomy, survey, challenges," *CoRR*, vol. abs/1903.06697, 2019. [Online]. Available: http://arxiv.org/abs/1903.06697

[5] B. Brock *et al.*, "Considerations for a distributed GraphBLAS API," in *GrAPL at IPDPS*. IEEE, 2020, pp. 215–218. [Online]. Available: https://doi.org/10.1109/IPDPSW50202.2020.00048

[6] ——, "A roadmap for the GraphBLAS C++ API," in *GrAPL at IPDPS*. IEEE, 2020, pp. 219–222. [Online]. Available: https://doi.org/10.1109/IPDPSW50202.2020.00049

[7] G. F. Grohoski, S. J. Luttrell, R. Prabhakar, R. Sivaramakrishnan, and M. K. Shah, "Virtualization of a reconfigurable data processor," U.S. Patent 20 200 257 643A1, Aug. 13, 2020.

[8] A. Iosup *et al.*, "LDBC Graphalytics: A benchmark for large-scale graph analysis on parallel and distributed platforms," *VLDB*, 2016. [Online]. Available: http://www.vldb.org/pvldb/vol9/p1317-iosup.pdf

[9] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *ISCA*. ACM, 2017. [Online]. Available: https://doi.org/10.1145/3079856.3080246

[10] G. Langdale and D. Lemire, "Parsing gigabytes of JSON per second," *VLDB J.*, vol. 28, no. 6, pp. 941–960, 2019. [Online]. Available: https://doi.org/10.1007/s00778-019-00578-5

[11] S. Lie, G. R. Lauterbach, M. E. James, M. Morrison, and S. Arekapudi, "Dataflow triggered tasks for accelerated deep learning," U.S. Patent 10 614 357B2, Apr. 7, 2020.

[12] T. Mattson *et al.*, "Standards for graph algorithm primitives," in *HPEC*. IEEE, 2013. [Online]. Available: https://doi.org/10.1109/HPEC.2013.6670338

[13] ——, "LAGraph: A community effort to collect graph algorithms built on top of the GraphBLAS," in *GrAPL at IPDPS*, 2019. [Online]. Available: https://doi.org/10.1109/IPDPSW.2019.00053

[14] T. G. Mattson *et al.*, "GraphBLAS C API: Ideas for future versions of the specification," in *HPEC*. IEEE, 2017. [Online]. Available: https://doi.org/10.1109/HPEC.2017.8091095

[15] Y. Wang *et al.*, "Gunrock: GPU graph analytics," *ACM Trans. Parallel Comput.*, vol. 4, no. 1, pp. 3:1–3:49, 2017. [Online]. Available: https://doi.org/10.1145/3108140

[16] C. Yang *et al.*, "Implementing push-pull efficiently in GraphBLAS," in *ICPP*. ACM, 2018, pp. 89:1–89:11. [Online]. Available: https://doi.org/10.1145/3225058.3225122

[17] ——, "GraphBLAST: A high-performance linear algebra-based graph framework on the GPU," *CoRR*, vol. abs/1908.01407, 2019. [Online]. Available: http://arxiv.org/abs/1908.01407