IEEE HPEC 2023

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

Egor Orachev[1] Semyon Grigorev[1]

egor.orachev@gmail.com s.v.grigoriev@spbu.ru,

[1] *St. Petersburg University, Russia*

Problem Statement

Scalable high-performance graph analysis is a challenge. GraphBLAS standard attempts to solve this challenge using sparse linear algebra operations. The full GPU-based implementation of this standard is still missing. Existing works are focused on Nvidia Cuda platform only, what limits their potability.

Is it possible to implement portable GPU-based library with generalized sparse linear algebra operations for graph analysis?

Results

Spla, the generalized sparse linear algebra library with vendor-agnostic GPUs accelerated computations.

- Proven **portability** of the OpenCL-based solution.
- Demonstrated **competitive** performance.
- Had acceptable **scalability** on devices of different GPUs vendors.
- Published **package** for python users.

Future Research

- Adaptive workload balance for better GPU's waves utilization.
- Performance tuning of auxiliary utilities, such as sorting, reduction, etc.
- New operations variations, such as SpGEMM, SpMM, etc.
- Graph data from RAM to VRAM streaming.
- Sclalability to multiple GPUs.

Motivating Example

```
from pyspla import *

def bfs(s: int, A: Matrix):
    v = Vector(A.n_rows, INT)
    front = Vector.from_lists([s], [1], A.n_rows, INT)
    front_size = 1
    depth = Scalar(INT, 0)

    while front_size > 0:
        depth += 1
        v.assign(front, depth, op_assign=INT.SECOND, op_select=INT.NQZERO)
        front = front.vxm(v, A, op_mult=INT.LAND, op_add=INT.LOR, op_select=INT.EQZERO)
        front_size = front.reduce(op_reduce=INT.PLUS).get()

    return v, depth.get()
```

Push-only BFS implementation with out of the box **GPUs acceleration** using pyspla package API available at PyPI.

Implementation

Spla implemented using the C++17 language and vendor-agnostic OpenCL 1.2 for GPU specific computations. General optimizations.

- **K-way merge-based** *vxm* operation [1].
- **Push-pull** and **early-out** [2].
- **Masking** of reached vertices [3].
- **Sparse-dense** storage switch [3].

OpenCL Specifics

Auxiliary functionality and optimizations to reduce mostly OpenCL CPU-side driver overhead.

- Run-time compiled **kernels cache** with a robin hood hash map for fast look-ups.
- In-place **unique kernel key** string constriction.
- Kernel code compilation with **user-defined** functions using simple text preprocessing.
- **Custom linear allocator** based on sub-buffer mechanism for temporary small ($\leq 1MiB$) allocations.
- **Custom auxiliary** operations, such as *sort*, *reduce*, *scan*.

Contact Us

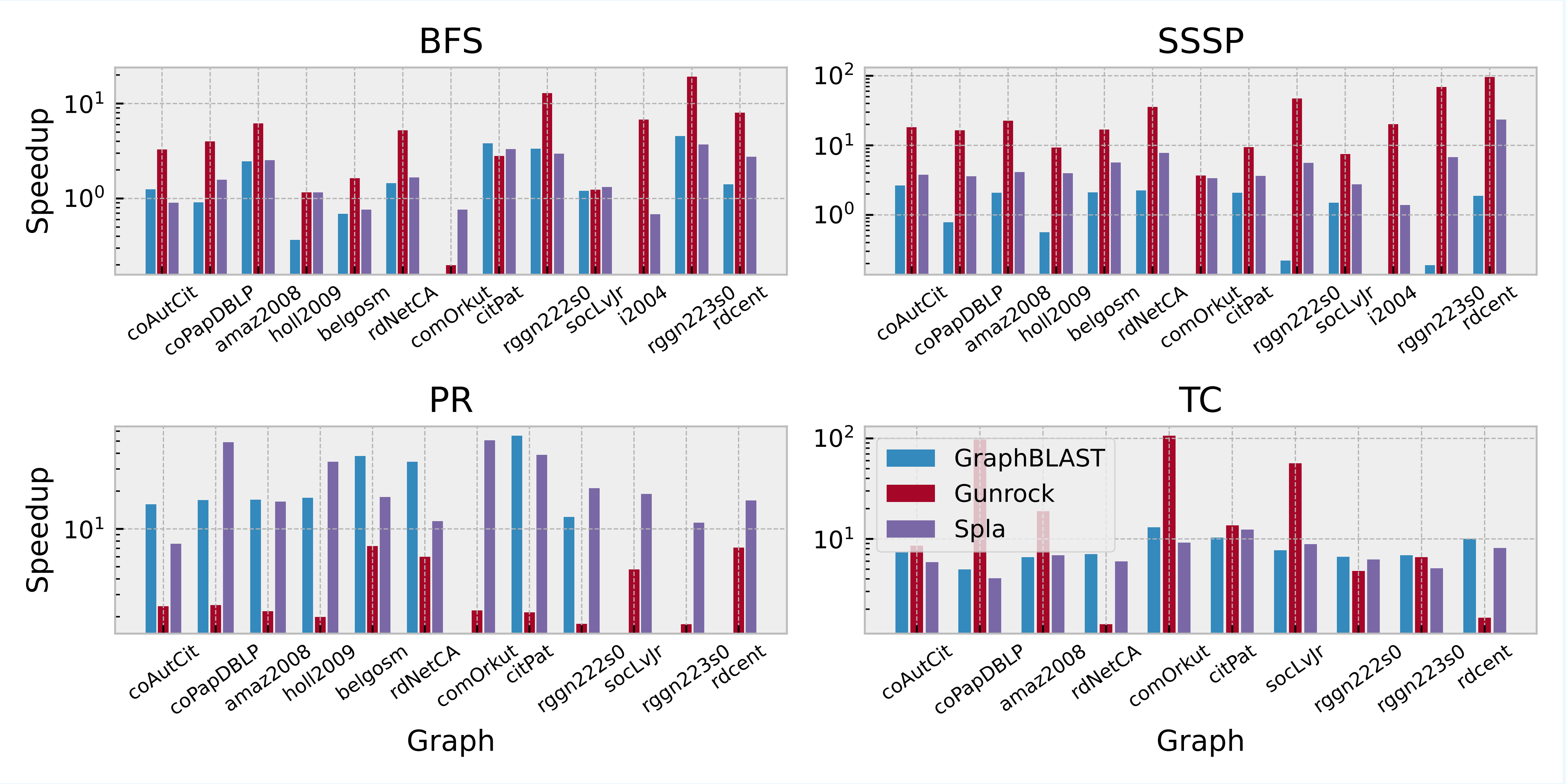
Our team:

- Semyon Grigorev: s.v.grigoriev@spbu.ru
- Egor Orachev: egor.orachev@gmail.com

Acknowledgments

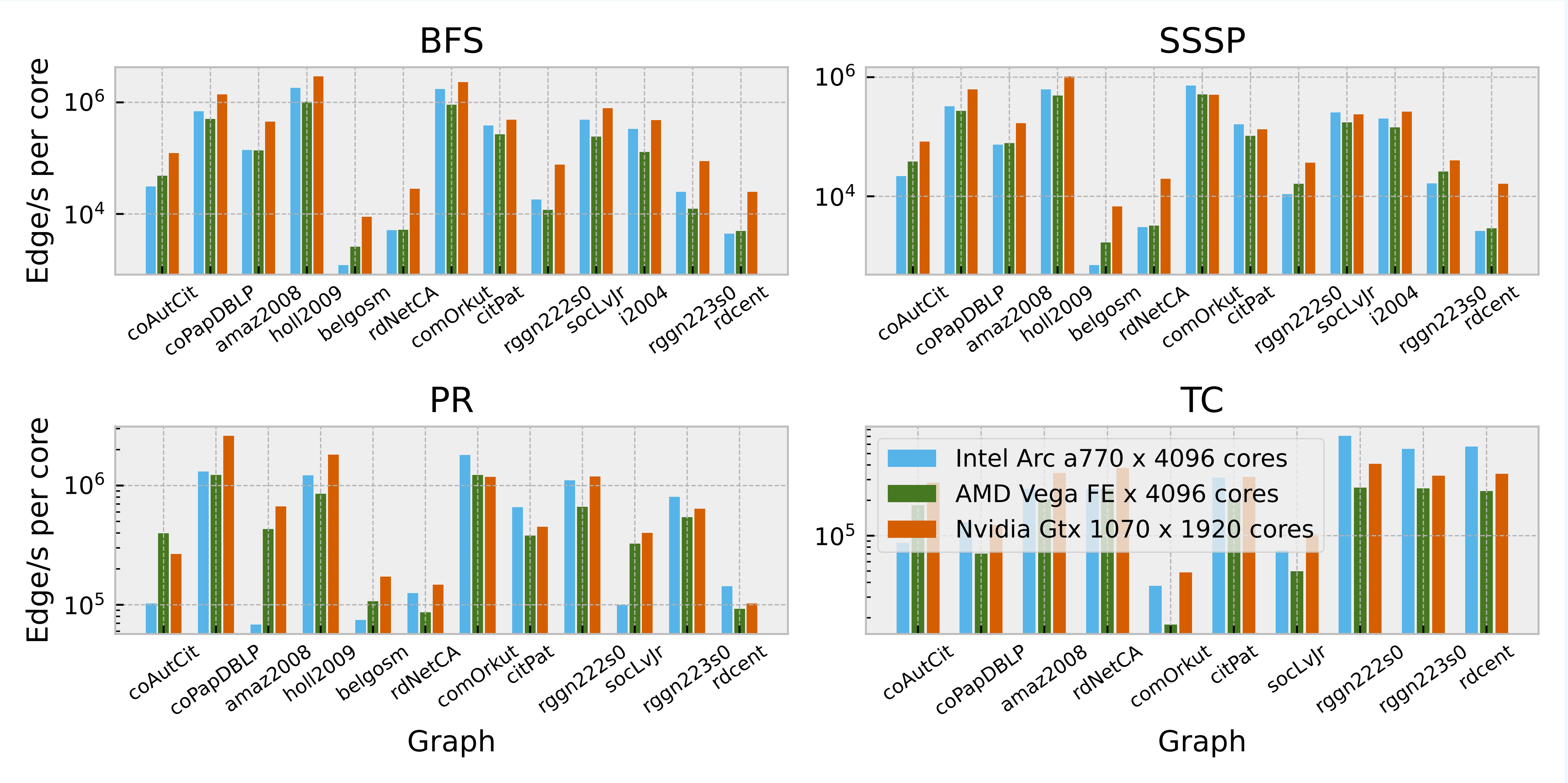
We would like to thank Huawei Technologies Co., Ltd for supporting this research. We are also grateful to our team and reviewers.

Evaluation: Performance on a Nvidia GPU



Tools: Gunrock [4], GraphBLAST [3], Spla and LaGraph [5] as a baseline.
Configuration: Ubuntu 20.04, 3.40Hz Intel Core i7-6700 4-core CPU, DDR4 64Gb RAM ♦ Nvidia GeForce GTX 1070 GPU with 8Gb VRAM.

Evaluation: Scalability on different GPUs



Configuration: Ubuntu 20.04, 3.40Hz Intel Core i7-6700 4-core CPU, DDR4 64Gb RAM ♦ Nvidia GeForce GTX 1070 GPU 8Gb VRAM ♦ Intel Arc A770 flux GPU 8GB VRAM ♦ AMD Radeon Vega Frontier Edition GPU 16GB VRAM.

References

[1] Carl Yang, Yangzihao Wang, and John D. Owens. Fast sparse matrix and sparse vector multiplication algorithm on the gpu. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, 2015.

[2] Carl Yang, Aydin Buluc, and John D. Owens. Implementing push-pull efficiently in graphblas, 2018.

[3] Carl Yang, Aydin Buluc, and John D. Owens. Graphblast: A high-performance linear algebra-based graph framework on the gpu, 2019.

[4] Yuechao Pan, Yangzihao Wang, Yuduo Wu, Carl Yang, and John D. Owens. Multi-gpu graph analytics. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2017.

[5] Gábor Szárnyas, David A. Bader, Timothy A. Davis, James Kitchen, Timothy G. Mattson, Scott McMillan, and Erik Welch. Lagraph: Linear algebra, network analysis libraries, and the study of graph algorithms, 2021.