Adapting Arkouda for Enabling Large Scale Graph Algorithms

Oliver Alvarado Rodriguez
New Jersey Institute of Technology

Introduction

Arkouda is built on a combination of Chapel and Python to enable high performance data analytics. Specifically, it can be adapted to enable large scale graph algorithms for data scientists who need to perform graph analytics. Development of algorithms can occur at both the front-end and/or back-end. When developing algorithms in the Chapel back-end it is required to always generate a front-end method in Python for users to call the methods. Communication between the front and back-ends are enabled through ZeroMQ, an asynchronous messaging library dedicated for distributed and concurrent applications. Since Arkouda was initially built to serve as a distributed array replacement to NumPy, there is also the option to utilize their array abstractions, pdarrays, to develop algorithms solely in the front-end. We have worked primarily on the development of algorithms in the back-end to provide another option in Python for data scientists who need to perform graph-centric computations.

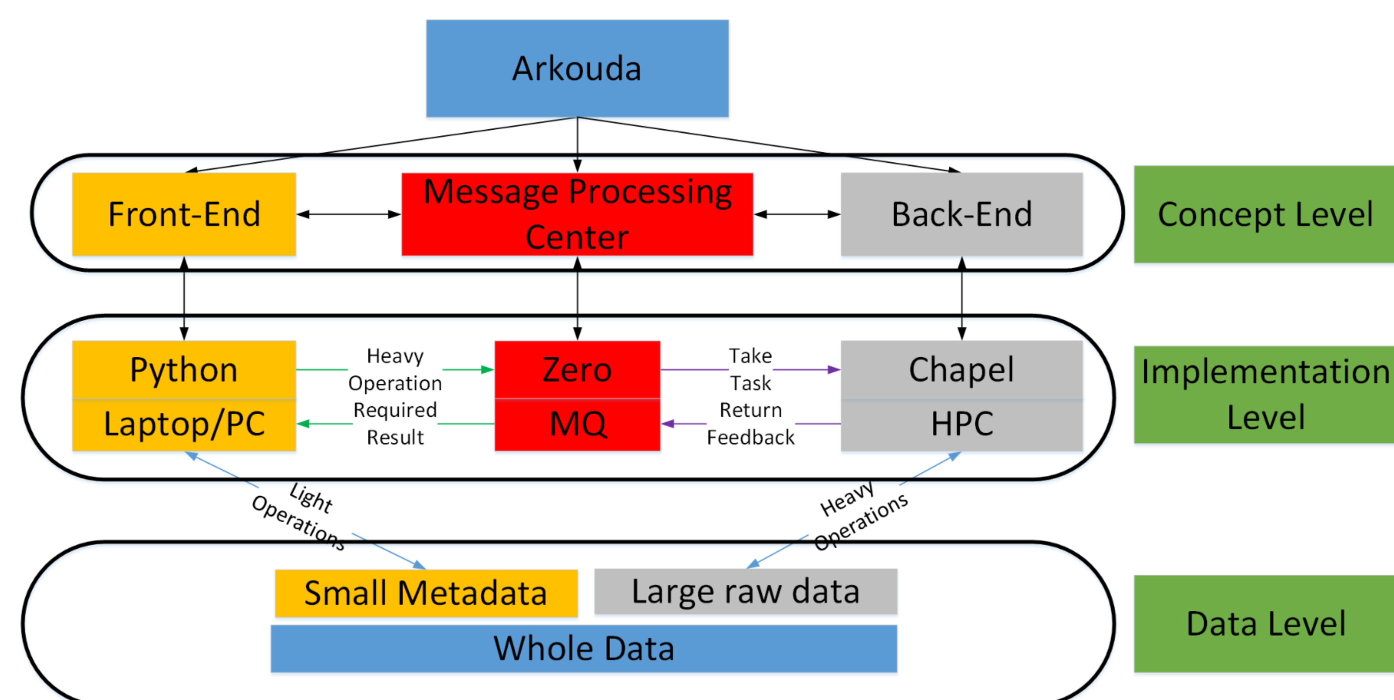

Fig. 1 – Diagram of the Arkouda developmental ecosystem [1].

In Fig. 1 we note three levels of the Arkouda ecosystem. First, is the concept level where the front-end and back-end work together through a message processing center. Then, we see the implementation level. We do not need to explicitly code the message processing center since it is handled by ZeroMQ. Lastly, the data level shows that the Chapel back-end is there to handle heavy data workloads and the Python front-end is only a way to "view" the large data.

Methodology

Thus far we have implemented four methods in our Arkouda graph framework which can be found in the following GitHub repo: [https://github.com/Bears-R-Us/arkouda-njit]. Two of these implementations have led to two publications [1-2]. The rest are work-in-progress experimental implementations. The full list, with a brief algorithmic description, is as follows:

1. Breadth-first search – neighbor vertices of each BFS step visited in parallel if they are on the same compute node [2].
2. Triangle counting – triangles are counted on a sketch of a graph on each node, summed up, & estimated at the end [1].
3. Connected components – components are searched in parallel in each partition using a BFS-like approach.
4. Betweenness centrality – initial workings of a parallel, not distributed approach with atomic arrays and variables.

Experimental Setup

Experiments were conducted on a 32-node cluster with an FDR InfiniBand between the nodes of the cluster. Each node has two 10-core Intel Xeon E5-2650 v3 @ 2.30GHz processors and 512GB DDR4 memory [1-3].

Results

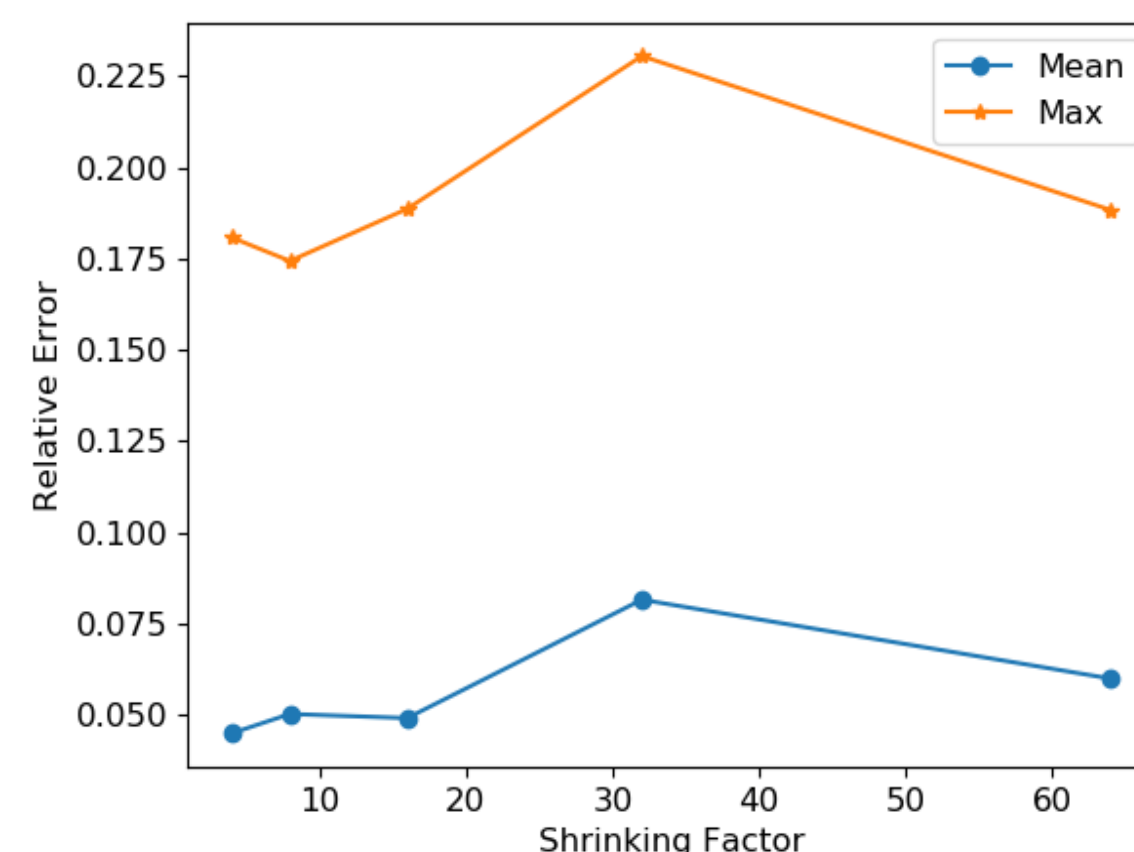

Fig. 2 – Low relative error for triangle counting on power-law graph sketches as shrink factor increases [2].

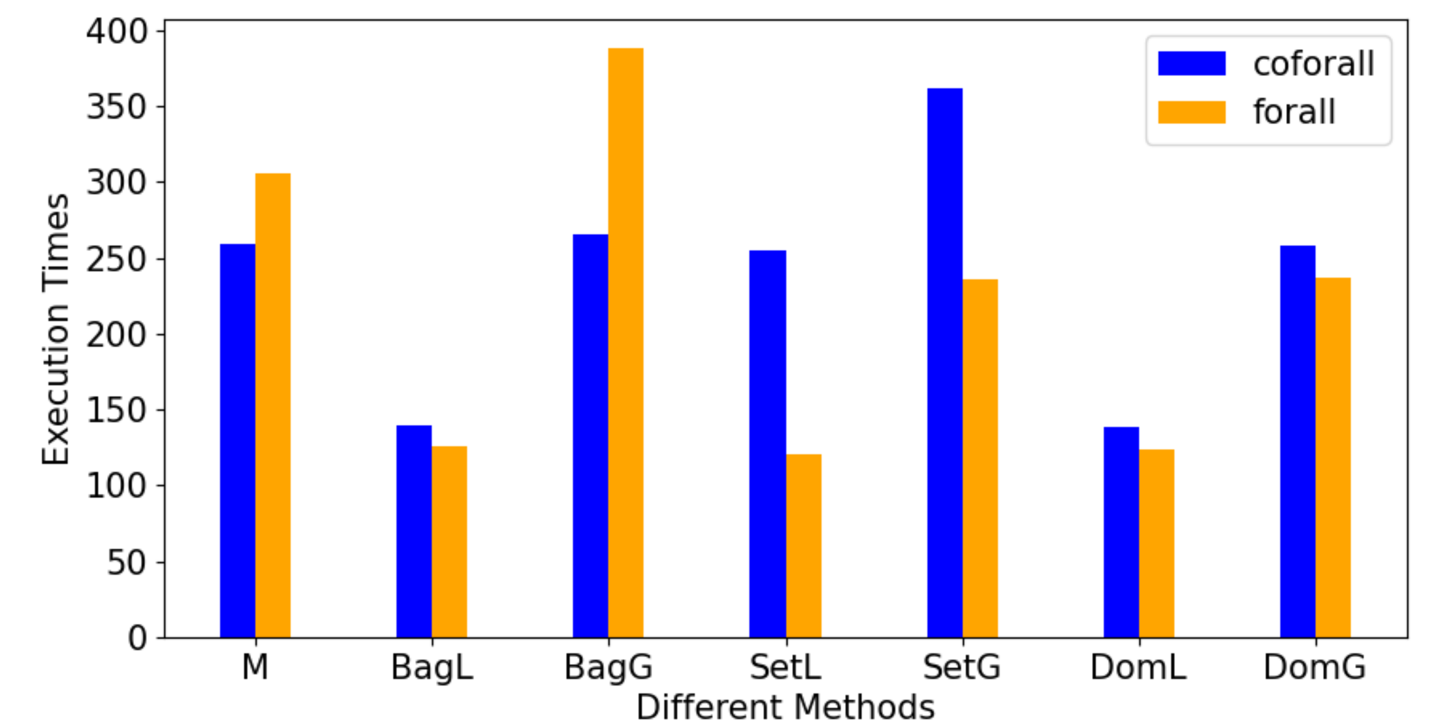

Fig. 3 – Delaunay-n20 graph BFS execution times with differing high-level Chapel data and parallel structures [2].

Conclusion

The need for large scale graph algorithms is growing as the sizes of the graphs we must analyze grow as well. There has been work currently to handle massive graph sizes, and our work is one of many in this area. Further work involves improving the optimization of our algorithms in Chapel and creating more efficient distributed memory graph algorithms.

Acknowledgments

I gratefully acknowledge the support of my Ph.D. advisor, Dr. David Bader. I also thank the support of Dr. Zhihui Du. We all acknowledge the Chapel and Arkouda communities and we also especially acknowledge the support of NSF grant number CCF-2109988 for funding this research.

References

[1] Z. Du, O. A. Rodriguez, J. Patchett, and D. A. Bader, "Interactive Graph Stream Analytics in Arkouda," Algorithms, vol. 14, no. 8, 2021, doi: 10.3390/a14080221.

[2] Z. Du, O. A. Rodriguez, and D. A. Bader, "Enabling Exploratory Large Scale Graph Analytics through Arkouda," 2021 IEEE High Performance Extreme Computing Conference (HPEC), 2021, pp. 1-7, doi: 10.1109/HPEC49654.2021.9622860.

[3] Z. Du, O. A. Rodriguez, and D. A. Bader, "Large Scale String Analytics in Arkouda," 2021 IEEE High Performance Extreme Computing Conference (HPEC), 2021, pp. 1-7, doi: 10.1109/HPEC49654.2021.9622810.