Chapel Graph Library (CGL)

Louis Jenkins
LouisJenkinsCS@hotmail.com
University of Rochester
Rochester, NY

Marcin Zalewski
marcin.zalewski@pnnl.gov
Pacific Northwest National Laboratory
Northwest Institute for Advanced Computing
Seattle, WA

Abstract

In this talk, I summarize prior work on the Chapel HyperGraph Library (CHGL), the Chapel Aggregation Library (CAL), and introduce the more general Chapel Graph Library (CGL). CGL is being designed to enable global-view programming, such that locality is abstracted from the user. CGL is also being designed in a way that is similar to Chapel's multiresolution design philosophy, where graphs are implemented in terms of hyper graphs, and where both the underlying hypergraph and overlying graphs are available for use. Some of the kinds of graphs being designed are bipartite graphs, directed and undirected graphs, and even trees.

CCS Concepts • **Software and its engineering** → **Software libraries and repositories**; • **Mathematics of computing** → *Graph algorithms*; *Hypergraphs*; • **Computing methodologies** → *Distributed programming languages.*

Keywords distributed, parallel, graph library, Chapel

ACM Reference Format:
Louis Jenkins and Marcin Zalewski. 2019. Chapel Graph Library (CGL). In *Proceedings of the ACM SIGPLAN 6th Chapel Implementers and Users Workshop (CHIUW '19), June 22, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3329722.3330149

Introduction

Graphs are important models for computation in both scientific computing and high-performance computing alike. Unfortunately there are many types of graphs, and implementing all of them is extremely time-consuming. As well, a lot of graphs tend share some overlap, wherein that overlap can be abstracted away. One such abstraction is the *hypergraph*, which is a generalization of graphs where edges are sets rather than fixed pairs of vertices.[1] Currently, the Chapel HyperGraph Library (CHGL) [1] is the only distributed-memory library for hypergraphs in Chapel. An example of CHGL application to big data analytics on DNS data can be found in Joslyn et al. [4]. While hypergraphs provide abstraction of complex high-dimensional data, 2-uniform-hypergraphs or just graphs are sufficient for many tasks. In the Chapel Graph Library, we take a novel approach in that we intend to implement all data structures on top of a lower-level hypergraph, similar to how Chapel is implemented. Explicit hypergraphs will still be available for use, but for many tasks more specialized graphs view can be used instead. Thanks to this design, the ability to convert a graph to a hypergraph will be a constant-time operation. For example, it is possible to implement undirected graphs and binary trees with hypergraphs; the result of the latter would look similar to fig. 1

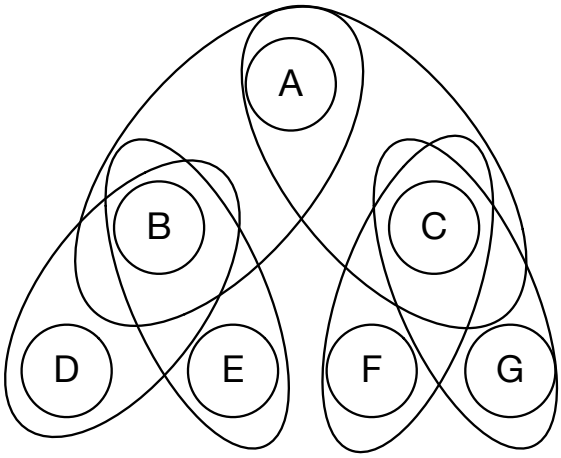

Figure 1. Binary Tree as a HyperGraph

Utilizing the hypergraphs of CHGL allows us to reuse all of the benefits of the existing implementation in CHGL. One such benefit is enabling global-view programming via privatization, where a *locale-private* instance of the class is managed on each locale to vastly improve locality of access,

CHIUW '19, June 22, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6800-1/19/06.
https://doi.org/10.1145/3329722.3330149

[1] Normal graphs are essentially hyper graphs where all hyper edges have a cardinality of 2.

along with its compiler-optimization *remote-value forwarding* which forwards all accesses to the locale-private instance, even when the default intent is by-reference such as in 'forall' loops. This can enable efficient distributed-memory access similar to that of Chapel's distributed arrays. As CHGL uses the Chapel Aggregation Library (CAL) [3], CGL can leverage the benefits of aggregation to aide in cases of fine-grained and irregular computations. Hence, CGL can more easily provide distributed graphs that are not just easy-to-use, but potentially fast depending on implementation. To enable specialized graphs to leverage specific information, CHGL is going to be revised incrementally to enable the implementation of such data structures and ensure their scalability.

Lastly, one exciting improvement for both CHGL and CGL both, is the planned overhaul of concurrent techniques, such as the new Interval-Based Memory Reclamation [5], and RCUArray [2], which both should help to enable concurrent modifications to the distributed hypergraph and graphs alike.

References

[1] Tanveer Hossain Bhuiyan, Sarah Harun, Christopher Lightsey, David Mentgen, Sinan G. Aksoy, Timothy Stavenger, Marcin Zalewski, Hugh R. Medal, and Cliff Joslyn. 2018. Chapel HyperGraph Library (CHGL). *2018 IEEE High Performance extreme Computing Conference (HPEC)* (2018), 1–6.

[2] L. Jenkins. 2018. RCUArray: An RCU-Like Parallel-Safe Distributed Resizable Array. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 925–933.

[3] L. Jenkins, M. Zalewski, and M. Ferguson. 2018. Chapel Aggregation Library (CAL). In *2018 IEEE/ACM Parallel Applications Workshop, Alternatives To MPI (PAW-ATM)*. 34–43.

[4] Cliff A Joslyn, Sinan Aksoy, Dustin Arendt, Louis Jenkins, Brenda Praggastis, Emilie Purvine, and Marcin Zalewski. [n.d.]. High Performance Hypergraph Analytics of Domain Name System Relationships. *HICSS Symposium on Cybersecurity Big Data Analytics* ([n. d.]), 8.

[5] Haosen Wen, Joseph Izraelevitz, Wentao Cai, H. Alan Beadle, and Michael L. Scott. 2018. Interval-Based Memory Reclamation. In *Proceedings of the 23rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '18)*. ACM, New York, NY, USA, 1–13.