Chapel HyperGraph Library (CHGL)

Louis Jenkins

Co-Authors

Tanveer Bhuiyan[2], Sarah Harun[2], Christopher Lightsey[2], David Mentgen[2], Sinan Aksoy[1], Timothy Stavenger[1], Marcin Zalewski[1], Hugh Medal[2], Cliff Joslyn[1]

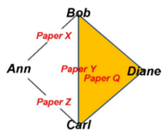
[1] Pacific Northwest National Laboratory, Seattle, Washington, USA.

[2] Mississippi State University, Mississippi State, Mississippi, USA.

Pacific Northwest NATIONAL LABORATORY

What is a Hypergraph?

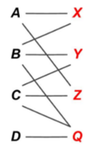
- A hypergraph is a generalization of a graph where edges can connect more than 2 vertices
  - An edge in a hypergraph is also called a *hyperedge*
- Hypergraphs can intuitively represent more complex connections



Hypergraph

	X	Y	Z	Q
A	x		x	
B	x	x		x
C		x	x	x
D				x

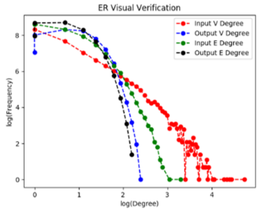
Incidence Matrix

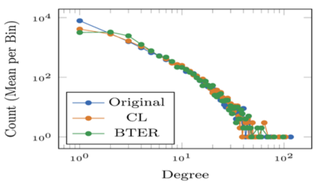


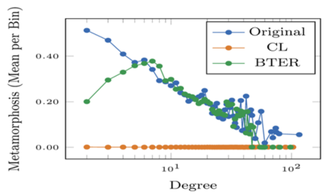
Bipartite Graph

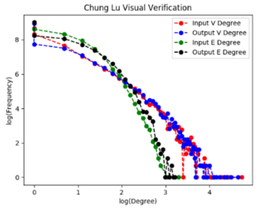
What is a Hypergraph Generator?

- A hypergraph generator can create a synthetic graph that matches characteristics of the original
  - Useful for stripping confidential information associated with the original dataset
  - Can create a larger graph that resembles the original
- More complex hypergraph generators are required to match more complex characteristics
  - Erdős-Rényi (ER) can match average degree but not heavy-tailed degree distribution
  - Chung-Lu (CL) can match degree distribution but not metamorphosis coefficients
  - Block Two-Level Erdős-Rényi (BTER) is capable of matching both degree distribution and metamorphosis coefficients

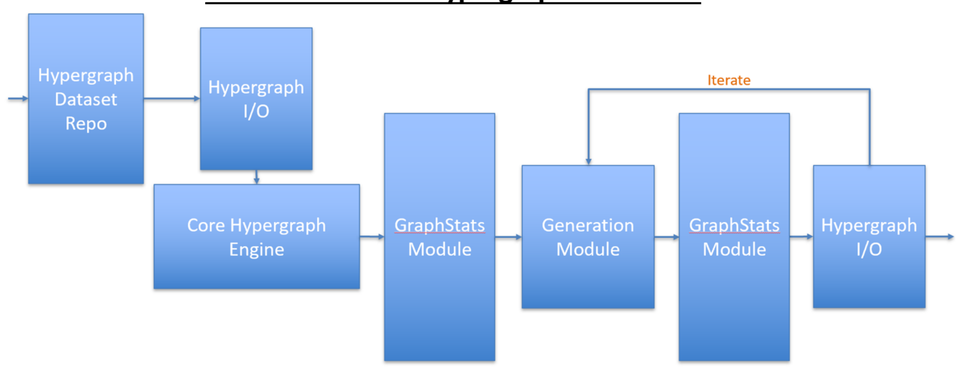








The First Exascale Hypergraph Generator



Design Principles

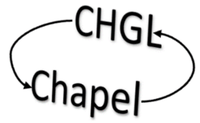
1. Generic and Versatile
   a. Interface-driven such that algorithms are reusable for different types
   b. Easy to compose existing algorithms to create larger more complex algorithms
2. High-Performance
   a. Scales in both shared-memory and distributed contexts
   b. Makes ample use of Chapel's parallelism and locality constructs
3. Easy-to-Use
   a. Intuitive and easy-to-use
   b. Simple and minimal interface

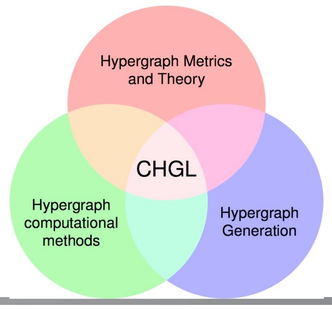
Why Chapel?

- Partitioned Global Address Space (PGAS) language
  - Developed by Cray
  - Funded by DARPA
- Offers very strong HPC abstractions and constructs
  - Task creation, migration, scheduling
  - Data-Driven Locality
    - "Moving the computation to the data"
  - Data-Parallelism
    - Distributed Parallel Iterators
- Makes it significantly easier to write distributed programs

Broader Impact

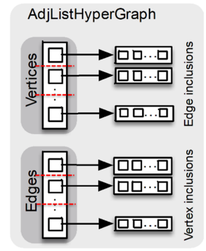
- Collaboration between PNNL and Cray
  - Chapel is not designed for irregular applications
  - Conducted meetings with Cray Chapel developers
  - Feedback loop between Chapel and CHGL
    - Chapel improves itself as flaws are exposed
    - CHGL improves as it uses Chapel
- Chapel Aggregation Library (CAL)
  - Written in Chapel, for Chapel
  - Solves performance problems and enables scalability for tested irregular algorithms
    - Performance testing performed on a Cray-XC50 Supercomputer
  - In-progress self-titled manuscript written in collaboration of PNNL and Cray
    - I am leading author
- Chapel HyperGraph Library (CHGL)
  - Self-titled manuscript accepted and to appear in HPEC-2018
    - I am leading author





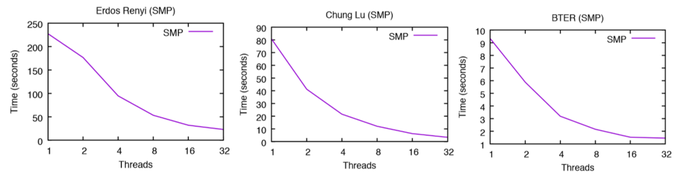
Implementation Overview

- AdjListHyperGraph is an adjacency list implementation of a dual hypergraph
  - If a vertex has an edge in its adjacency list, then that edge also has that vertex in its adjacency list
    - Bidirectional mapping of vertices and edges
  - Each adjacency list is contiguous array of neighbors
- Can be distributed over multiple *locales*, or processing unit
  - Using Chapel's Domain-map Standard Interface (DSI)
    - BlockDist, CyclicDist, etc.
- Make use of low-level runtime and compiler optimizations
  - Privatization – Creates a local copy on each locale
    - All access forward to privatized instance
  - Record-Wrapping – Eliminate communication from accessing in distributed context
    - Makes use of 'remote-value forwarding' compiler optimization

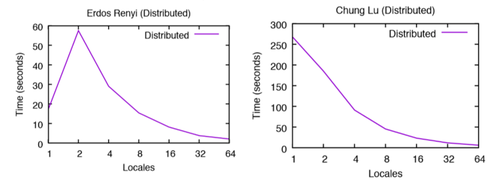


Performance

- Hypergraph scales perfectly in Shared Memory for hypergraph generation



- Hypergraph scales perfectly after 2 locales
  - Demonstrates overhead of adding communication
  - BTER is under construction



Code Sample: Naïve Erdős-Rényi

```
const numVertices = 1024 * 1024;
const numEdges = 2048 * 1024;
const map = new Cyclic(startIdx=0, targetLocales=Locales[4..8 by 2]);
const probability = 0.5;
var graph = new AdjListHyperGraph(numVertices, numEdges, map);
var rng = new RandomStream(real);
forall v in graph.getVertices() {
  forall e in graph.getEdges() {
    if rng.getNext() <= probability {
      graph.addInclusion(v,e);
    }
  }
}
```


U.S. DEPARTMENT OF ENERGY