PowerGraph

Distributed Graph-Parallel Computation on Natural Graphs

Joseph E. Gonzalez, Yucheng Low, Haijie Gu,

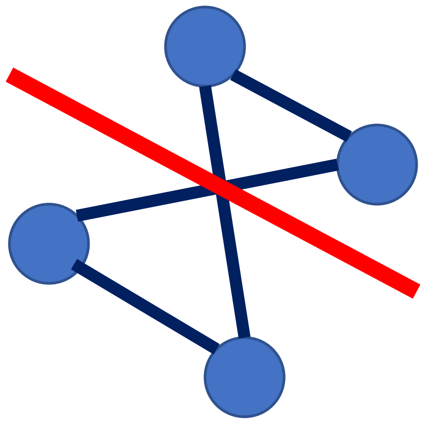
Danny Bickson, Carlos Guestrin

Presentation by: Richard Sollee

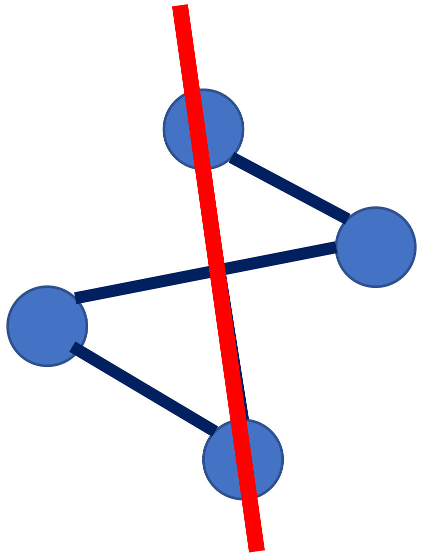
Graph Abstractions

Typical graph abstractions assume each vertex has few neighbors.

They partition the graph for parallel work based on vertices so they used edge cuts.



Edge Cut



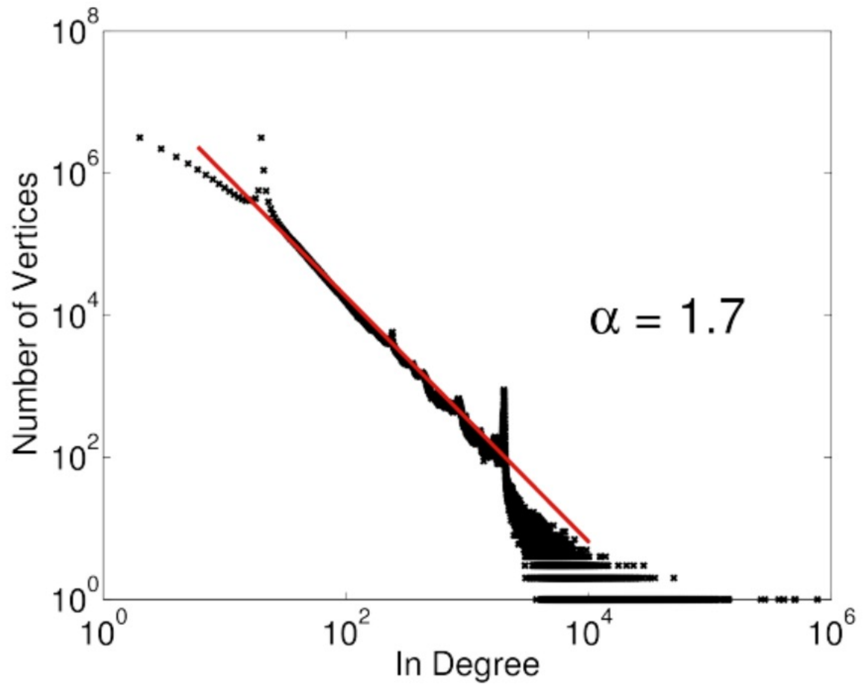
Vertex Cut

Natural Graphs

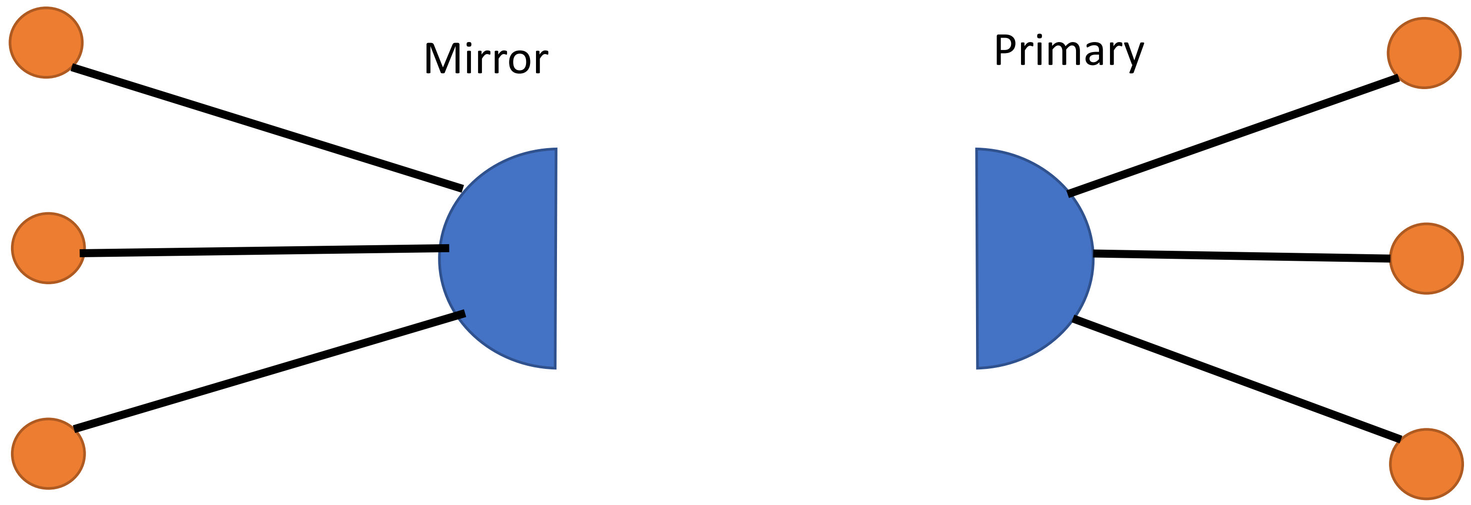
Natural graphs are those found in the real world such as social networks.

They typically have highly skewed power-law degree distributions.

This means that a few vertices have many edges while some have few.

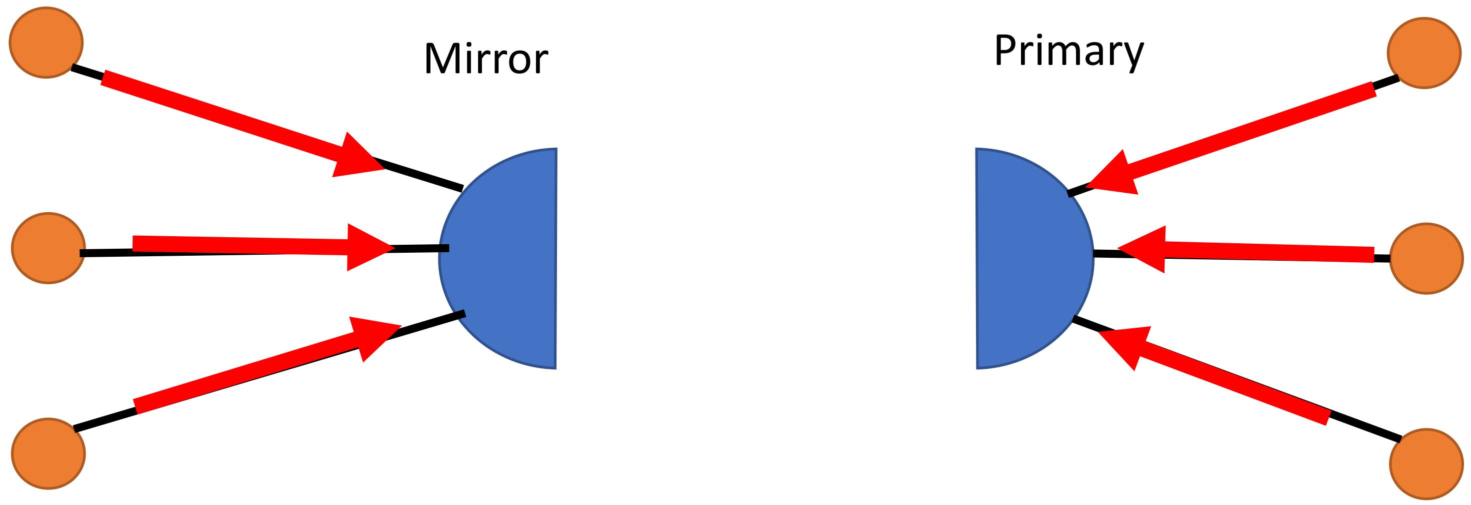


GAS Model (Gather, Apply, Scatter)



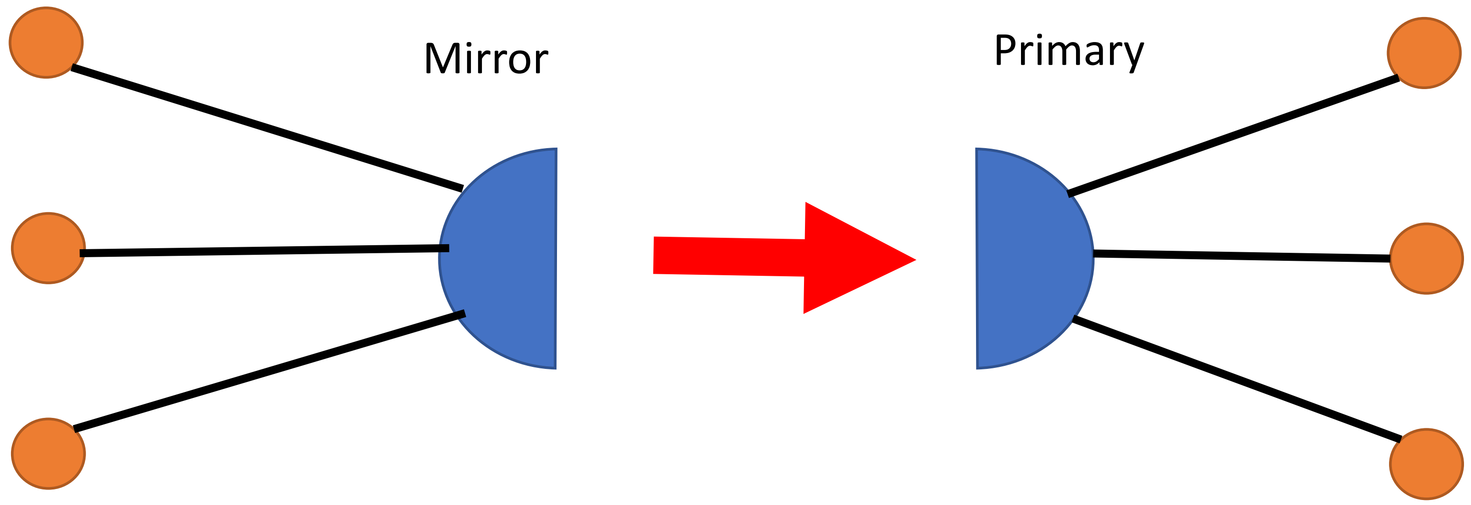
GAS Model (Gather, Apply, Scatter)

First, the algorithm **gathers** the adjacent edge and vertex info.



GAS Model (Gather, Apply, Scatter)

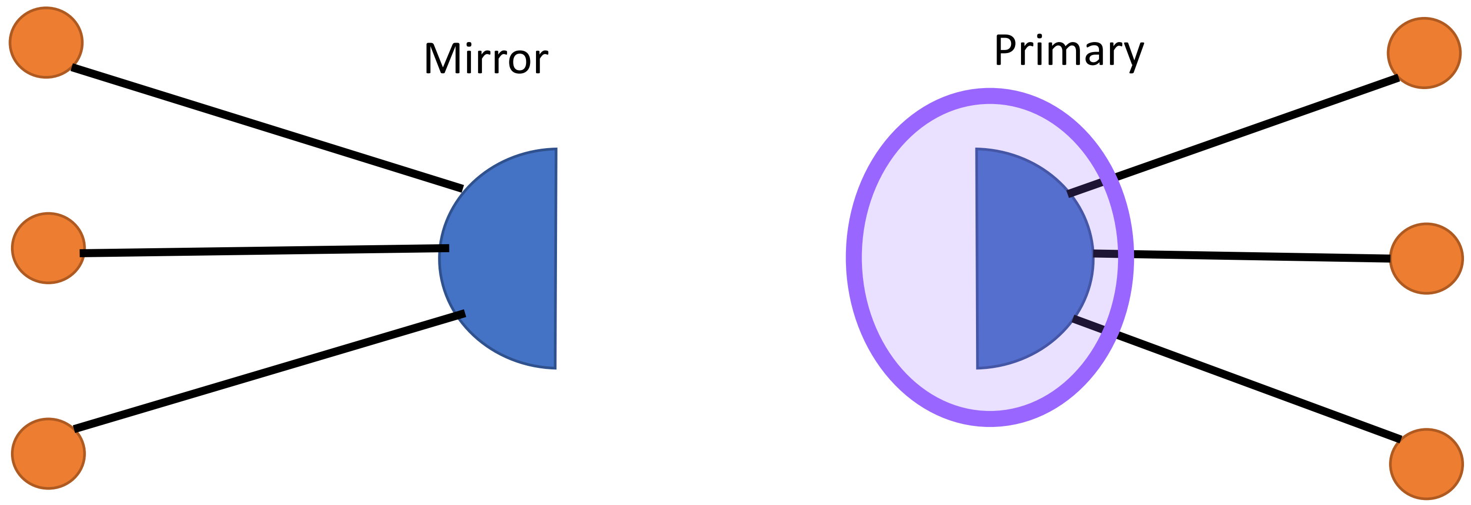
First, the algorithm **gathers** the adjacent edge and vertex info.



GAS Model (Gather, Apply, Scatter)

First, the algorithm **gathers** the adjacent edge and vertex info.

Then, it **applies** an update to the central vertex using the gathered info.

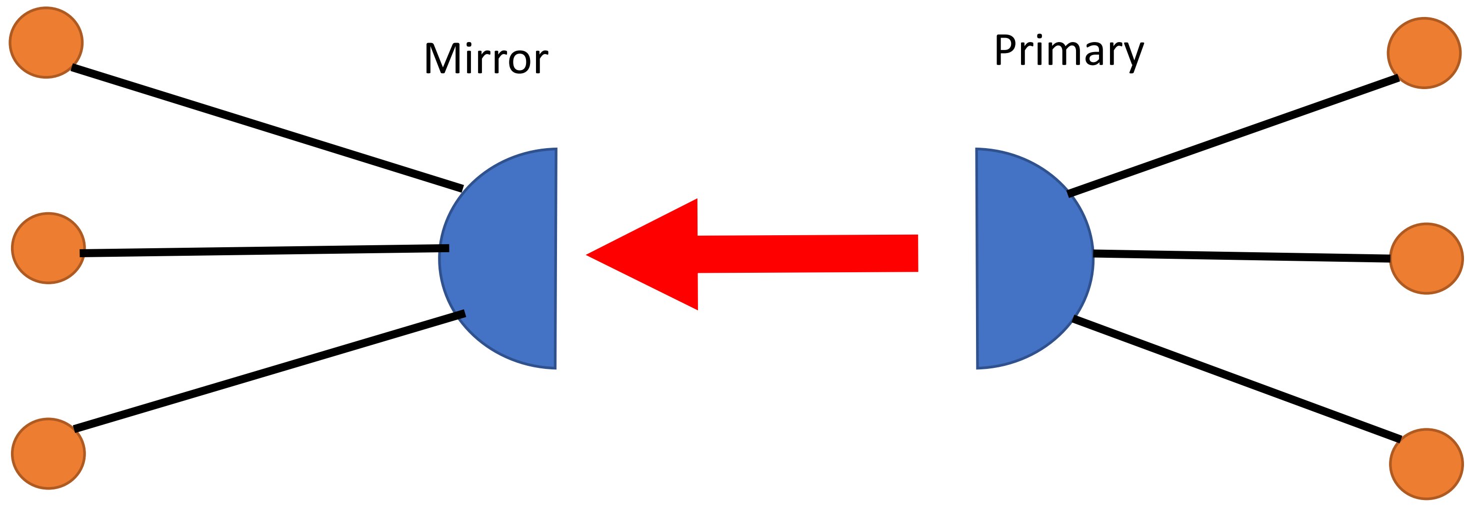


GAS Model (Gather, Apply, Scatter)

First, the algorithm **gathers** the adjacent edge and vertex info.

Then, it **applies** an update to the central vertex using the gathered info.

Finally, it **scatters** the new value of the vertex to update adjacent edges.

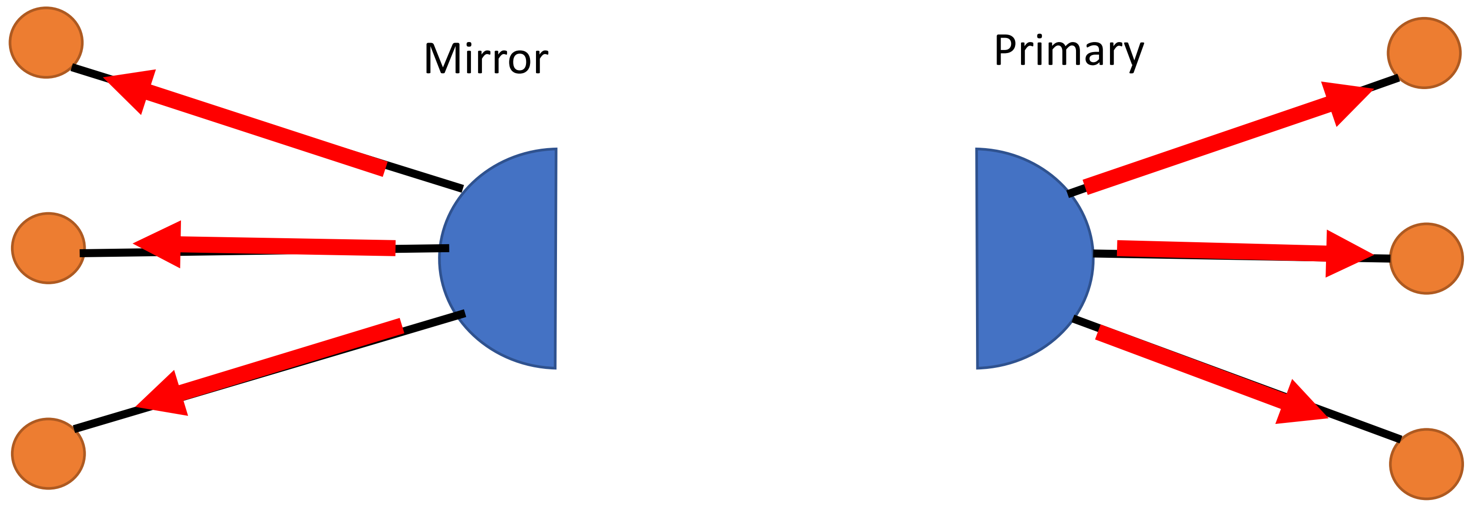


GAS Model (Gather, Apply, Scatter)

First, the algorithm **gathers** the adjacent edge and vertex info.

Then, it **applies** an update to the central vertex using the gathered info.

Finally, it **scatters** the new value of the vertex to update adjacent edges.



PowerGraph

Begin the program by activating a vertex.

The program then runs until no vertices are activated.

Vertices can activate themselves and neighboring vertices.

The order of activated vertex execution is not defined but all activated vertices are guaranteed to eventually execute.

Async or synchronous executions are available.

Vertex Cut

A vertex cut permits a single vertex program to span multiple machines.

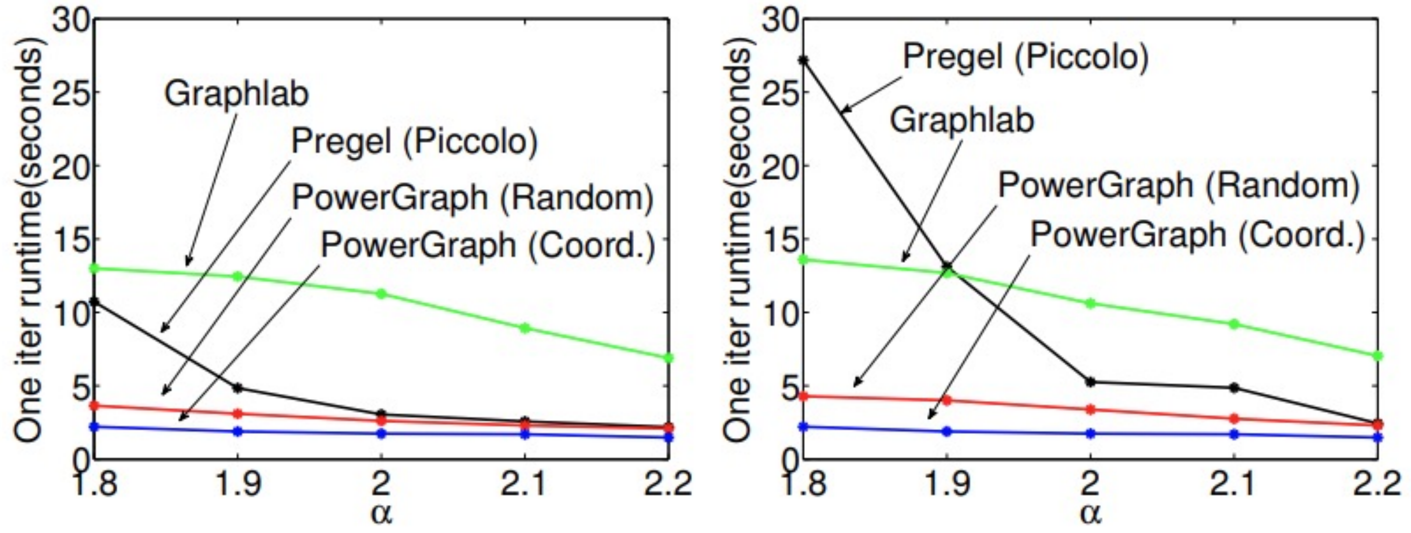
This allows evenly assigning edges.

A random vertex cut can be used.

A greedy vertex cut can improve performance.

- It places edges to minimize conditional expected replication factor.
- It requires machine coordination which increases initial overhead.
- Reduction in communication needs makes it faster in practice.

Performance Comparison



(a) Power-law Fan-In Runtime (b) Power-law Fan-Out Runtime

Figure 10: **Synthetic Experiments Runtime. (a, b)** Per iteration runtime of each abstraction on synthetic power-law graphs.

Recent Work

GraphChi

GraphChi is designed for use on a single machine.

PowerGraph is designed for a distributed system.

PowerGraph outperforms but not proportional to the resources available.

GraphX

GraphX is a distributed graph system like PowerGraph and is built on Apache Spark, a distributed dataflow system.

GraphX has similar performance but scales better.

Ligra

Ligra is designed for a single machine.

Ligra is slightly faster per iteration than PowerGraph on PageRank on the Twitter grpah.

X-Stream

Like PowerGraph, X-Stream maintains state in the vertices and exposes a scatter-gather model.

X-Stream is designed for a single shared-memory machine.

X-Steam uses an edge centric iteration method where the different phases iterate over edges rather than vertices.

X-Stream outperforms GraphChi and Ligra (when taking into account pre-processing time).