Optimising Graph Algorithms on Pregel-Like Systems

S. Salihoglu, J. Widom
Stanford University

Philip Leonard

November 24th, 2014

Pregel Reminder

- Bulk Synchronous Parallel model
- Vertex centric program `vertex.compute()`
- Computation unit is a superstep
- Optional `master.compute()` for serial computation
- Vertices receive data from previous superstep, update locally and then broadcast results
- Open Source implementations; Giraph and GPS

Whats Wrong With Pregel?

- Slow convergence and communication costs
- Performance reflects graph structure
- Deals poorly with natural graphs

Key Costs

- **Communication**
- **Number of supersteps**
- Memory
- Computation

Optimisations will focus primarily on the first two.

The Optimsiations I

- Finish Computations Serially (FCS)
  - Slow convergence arises from graphs with structure
  - FCS reduces convergence time
  - Can be applied to algorithms where graph "shrinks"
  - When active graph small enough, final computation is finished serially in `master.compute()`, and then broadcasts results back to workers

- Storing Edges At Subvertices (SEAS)
  - Set of vertices merged to form supervertices
  - In SEAS, subvertices are kept alive and they retain adjacency matrices
  - Increases communication between sub and supervertices, but reduces computation (runtime)

The Optimisations II

- Edge Cleaning On Demand (ECOD)
  - Pregel deletes edges in a superstep
  - ECOD deletes 'stale' edges when they are discovered in computation
  - Eager vs Lazy cleaning
- Single Pivot (SP)
  - Some graphs exhibit a single giant component
  - SP avoids excessive communication
  - Used to find large components quickly (Useful for finding Strong/Weak Connected Components)

The Algorithms

- Strongly Connected Components
- Minimum Spanning Forest
- Graph Colouring
- Approximate Maximum Weight Matching
- Weakly Connected Components

- Most require multiple computation *“phases”*

Strongly Connected Components

What it does?

Parallel colouring algorithm for finding strongly connected components in a graph.

Optimisations

- Finishing Computations Serially; 1.3x to 2.3x runtime reduction and 26% to 56% reduction in supersteps on webgraphs
- Single Pivot; 1.1x to 1.2x runtime reduction
- FCS + SP; 1.45x to 3.7x runtime reduction

Minimum Spanning Forest

What it does?
Minimum Spanning Trees found in disconnected graph components (forest), then merged in supervertex-formation.

Optimisations

- Storing Edges At Subvertices; 1.15x to 3x runtime reduction
- Plus Edge Cleaning On Demand; 1.9x increase of communication, 1.2x to 3.3x runtime reduction
- Finishing Computations Serially; Can be applied, but convergence isn't slow for MSF (supervertex-formation is fast).

Graph Colouring

What it does?

Greedy heuristic for graph colouring problem, iteratively finding Maximal Independent Set in order to colour a graph in as fewer colours as possible.

Optimisations

- Finishing Computations Serially; 1.1x to 1.4x runtime reduction and 10% to 20% reduction in supersteps on .sk webgraph

Approximate Maximum Weight Matching

What it does?

In a undirected weighted graph, Approximate MWM is a 1/2-approximation algorithm to find a set of vertex-disjoint edges with maximum weight

Optimisations

- Edge Cleaning On Demand; 1.45x runtime reduction, 1.3x to 3.1x reduction in communication and 1.7x to 2.2x increase in number of supersteps

Weakly Connected Components

What it does?

Finding maximal subgraphs of a directed graph such that replacing directed edges with undirected edges produces a connected undirected graph

Optimisations

- Single Pivot; 2.7x to 7.4x runtime reduction on all graphs

Criticisms

Might have benefited from implementing at least one pre-implemented graph algorithm.

Couldn't find a reason for why MWM isn't tested using FCS, even though it claims to optimise it.

Message combiners are placed in related work section?

Conclusion

- Identified a good set of optimisations which appear to work well
- All algorithms optimised in at least one area if not more
- These previously unused algorithms might now be feasible for Pregel-like systems
- Future work might see these optimisations included in a library form for Pregel-like systems