Pregel: A System for Large-Scale Graph Processing

Grzegorz Malewicz, Matthew H. Austern, Aart J. C. Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski

Bogdan-Alexandru Matican

University of Cambridge

February 26, 2013

Table of contents

1 Research questions

2 Design
- Programming Model
- Usability
- Architecture

3 Experiments

4 Conclusion

Main considerations

Typical Google system's paper.
Cross-research influences: MapReduce, Chubby, GFS, BigTable.

Scalability	process graphs of billions of vertexes
Usability	paradigm, API, features
Architecture	Master-Slave, network aggregation, data locality
Transparency	fault tolerance, commodity machines
Performance	resources, speed, scale

Vertex

- local action: vertex and outgoing edges
- message passing communication
- independent state change: synchronicity

System

- supersteps (BSP model)
- message based state alterations
- aggregation performance optimizations
- fault tolerance (check-pointing)

API Design

- simple interface for users to understand
- usage pattern driven: Combiner, Aggregator, Http
- IO format variable for interoperability
- fault tolerance transparent
- data partitioning

Components and Mechanics

- data sharding (graph partitioning)
- Master (ids, sharding, sync, pings)
- Workers (supersteps, state, buffering)
- fault tolerance (check-pointing, confined recovery)
- performance considerations

Scalability

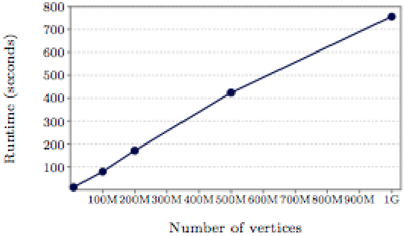

Figure : Binary tree topology for 800 workers, 300 machines.

Linear scaling of runtime for binary fan-out, high vertex count.

Scalability

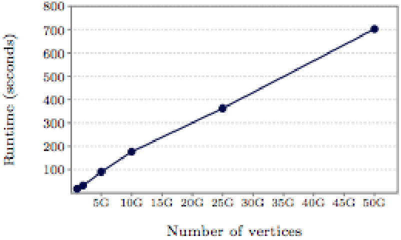

Figure : Social graph topology for 800 workers, 300 machines.

Linear scaling of runtime for relatively sparse graphs with instances of high density.

Notes

- naive implementation of SSSP
- no input pre-processing or special sharding
- comparable results with state-of-the-art systems
- scalable considerably past points shown in paper

Contributions

- programming model
- design simplicity
- concurency avoidance
- fault tolerance
- performance optimizations

Critique and questions

- master failover mechanism?
- evaluation: good enough for us
- evaluation: how much faster?