Solving Massive Graph Problems in GraphChi

Ilias Giechaskiel

Cambridge University, R212
ig305@cam.ac.uk

March 11, 2014

Overview

GraphChi [KBG12]

- Appealing for low-budget graph processing
- Relevance depends on two metrics:
  - Ease of vertex-centric algorithm implementations
  - Efficiency

This Project

- Implementation of traditional graph algorithms
- Experimental (and comparative?) study

Background

GraphChi

- Disk-based, single PC system for massive graphs
- Vertex-centric
- Parallel Sliding Windows (PSW)
  - Each vertex mapped to interval, stored in shard
  - Shard also contains in-edges, fits in memory
  - Asynchronous
  - $\mathcal{O}(P^2)$ random disk accesses per iteration

Motivation

Implementation

- Graph traversal inefficient
- Evaluation focuses on non-traditional algorithms:
  - PageRank, belief propagation, matrix factorization
- Triangle counting

Comment by project member akyrola...@gmail.com, Aug 29, 2012

It is easy to run shortest path with BFS in GraphChi (maybe not super-efficient though):

In the update function, set the distance of the vertex (stored as the value of vertex), to be 1 + minimum of the distances of its neighbors. The distance of a neighbor is read from the inedges: this means, that a vertex must write its distance to its out-edges, so neighbors can read it. See the connected component example, it is quite similar.

Figure: `https://code.google.com/p/graphchi/wiki/CreatingGraphChiApplications`

Example

Triangle Counting

- More than 400 LOC excluding comments
- Source code comments:
  - This algorithm is quite complicated and requires 'trickery' to work well on GraphChi
  - The application involves a special preprocessing step
- https://github.com/GraphChi/graphchi-cpp/blob/master/example_apps/trianglecounting.cpp

This Project

Algorithms

- Many algorithms for same graph problem
  - But which ones can be implemented?
- Connected Components (CC)
  - BFS, DFS, **Union-Find**
  - Goal: Optimize implementation using path compression
- Minimum Spanning Tree (MST)
  - Prim, Kruskal, **Boruvka**, etc.
  - Goal: Implement Kruskal using Union-Find
- Single Source Shortest Path (SSSP)
  - Dijkstra, Bellman-Ford, etc.
  - Reach goal: Implement any algorithm
- Expected result: goals achievable, anything else really hard

Motivation

Efficiency

- Distributed systems up to 40x faster
  - At 256x more power
- Pre-processing up to 37 minutes
  - Slower to partition Yahoo graph than run Webgraph on it!

Application & Graph	Iter.	Comparative result	GraphChi (Mac Mini)
Pagerank & domain	3	GraphLab[31] on AMD server (8 CPUs) **87 s**	**132 s**
Pagerank & twitter-2010	5	Spark [48] with 50 nodes (100 CPUs): **486.6 s**	**790 s**
Pagerank & V=105M, E=3.7B	100	Stanford GPS, 30 EC2 nodes (60 virt. cores), **144 min**	approx. **581 min**
Pagerank & V=1.0B, E=18.5B	1	Piccolo, 100 EC2 instances (200 cores) **70 s**	approx. **26 min**
Webgraph-BP & yahoo-web	1	Pegasus (Hadoop) on 100 machines: **22 min**	**27 min**
ALS & netflix-mm, D=20	10	GraphLab on AMD server: **4.7 min**	**9.8 min** (in-mem) **40 min** (edge-repl.)
Triangle-count & twitter-2010	-	Hadoop, 1636 nodes: **423 min**	**60 min**
Pagerank & twitter-2010	1	PowerGraph, 64 x 8 cores: **3.6 s**	**158 s**
Triange-count & twitter- 2010	-	PowerGraph, 64 x 8 cores: **1.5 min**	**60 min**

Experiments

- Test algorithms runtime
  - Goal: Compare HDD vs. SSD
- Comparison with other systems
  - Goal: X-Stream [RMZ13]
  - Reach goal: Pregel [MAB+10]
  - Impossible: Turbograph [HLP+13]
- Expected result: Pregel $>$ X-Stream $\gg$ SSD $\gg$ HDD

Conclusions

Key Questions

- How easy is it to solve traditional graph problems?
  - Answer for CC, MST, SSSP
- How slow is GraphChi?
  - Compare SSD vs. HDD
  - Compare to X-Stream

Bibliography I

- Wook-Shin Han, Sangyeon Lee, Kyungyeol Park, Jeong-Hoon Lee, Min-Soo Kim, Jinha Kim, and Hwanjo Yu, *Turbograph: A fast parallel graph engine handling billion-scale graphs in a single pc*, Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (New York, NY, USA), KDD '13, ACM, 2013, pp. 77–85.

- Aapo Kyrola, Guy Blelloch, and Carlos Guestrin, *Graphchi: Large-scale graph computation on just a pc*, Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation (Berkeley, CA, USA), OSDI'12, USENIX Association, 2012, pp. 31–46.

Bibliography II

- Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski, *Pregel: A system for large-scale graph processing*, Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (New York, NY, USA), SIGMOD '10, ACM, 2010, pp. 135–146.
- Amitabha Roy, Ivo Mihailovic, and Willy Zwaenepoel, *X-stream: Edge-centric graph processing using streaming partitions*, Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles (New York, NY, USA), SOSP '13, ACM, 2013, pp. 472–488.