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
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
  - $O(P^2)$ 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!

<table>
<thead>
<tr>
<th>Application &amp; Graph</th>
<th>Iter.</th>
<th>Comparative result</th>
<th>GraphChi (Mac Mini)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pagerank &amp; domain</td>
<td>3</td>
<td>GraphLab[^51] on AMD server (8 CPUs): 87 s</td>
<td>132 s</td>
</tr>
<tr>
<td>Pagerank &amp; twitter-2010</td>
<td>5</td>
<td>Spark[^48] with 50 nodes (100 CPUs): 486.6 s</td>
<td>790 s</td>
</tr>
<tr>
<td>Pagerank &amp; V=105M, E=3.7B</td>
<td>100</td>
<td>Stanford GPS, 30 EC2 nodes (60 virt. cores), 144 min</td>
<td>approx. 581 min</td>
</tr>
<tr>
<td>Pagerank &amp; V=1.0B, E=18.5B</td>
<td>1</td>
<td>Piccolo, 100 EC2 instances (200 cores) 70 s</td>
<td>approx. 26 min</td>
</tr>
<tr>
<td>Webgraph-BP &amp; yahoo-web</td>
<td>1</td>
<td>Pegasus (Hadoop) on 100 machines: 22 min</td>
<td>27 min</td>
</tr>
<tr>
<td>ALS &amp; netflix-mm, D=20</td>
<td>10</td>
<td>GraphLab on AMD server: 4.7 min</td>
<td>9.8 min (in-mem)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>40 min (edge-repl.)</td>
</tr>
<tr>
<td>Triangle-count &amp; twitter-2010</td>
<td>-</td>
<td>Hadoop, 1636 nodes: 423 min</td>
<td>60 min</td>
</tr>
<tr>
<td>Pagerank &amp; twitter-2010</td>
<td>1</td>
<td>PowerGraph, 64 x 8 cores: 3.6 s</td>
<td>158 s</td>
</tr>
<tr>
<td>Triangle-count &amp; twitter-2010</td>
<td>-</td>
<td>PowerGraph, 64 x 8 cores: 1.5 min</td>
<td>60 min</td>
</tr>
</tbody>
</table>
This Project

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 ≫ SSD ≫ 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

---

**Ilias Giechaskiel**

ig305@cam.ac.uk

---

**Solving Massive Graph Problems in GraphChi**
