X-Stream: Edge-centric Graph Processing using Streaming Partitions

AMITABHA ROY, IVO MIHAILOVIC, WILLY ZWAENEPOEL
PRESENTED BY: MAREK STRELEC
Motivation

- Large graphs – billions of vertices and edges
- Process on large clusters
  - Pregel, GraphLab, PowerGraph, Niad
  - Complexity and cost
- Process on a single machine
  - GraphChi, X-Stream
- 64 GB RAM, 32 cores, 2 x 200 GB SSD, 3 x 3TB drive
Vertex-centric processing model

- “Think like a vertex”
- Popularized by the Pregel and GraphLab projects
- Mutable states stored in vertices
- Scatter-Gather model
  - Scatter updates along outgoing edges
  - Gather updates from incoming edges
Vertex-centric BFS

Source | Destination
--- | ---
1 | 3
1 | 5
2 | 4
3 | 2
3 | 5
5 | 4
Vertex-centric BFS

- Vertices
  - 1
  - 2
  - 3
  - 4
  - 5

- Source | Destination
  - 1 | 3
  - 1 | 5
  - 2 | 4
  - 3 | 2
  - 3 | 5
  - 5 | 4
Vertex-centric BFS
Vertex-centric BFS
Sequential vs. Random access

- Graph traversal = Random access

- For all storage media (RAM, SSD, and HDD)
  - Sequential bandwidth >> random access bandwidth
  - HDD - 300x higher
  - SSD - 30x higher
  - RAM (1 core) - 4.6x higher
  - RAM (16 cores) - 1.8x higher
X-stream processing model: Edge-centric

- Input to X-stream is an unordered set of directed edges
  - For undirected graphs - pair of directed edges
- Scatter and Gather phases iterate over vertices/edges
- X-stream makes graph access sequential
Edge-centric BFS
Edge-centric BFS
Edge-centric BFS

Vertices
1
2
3
4
5

Source | Destination
--- | ---
1 | 3
1 | 5
2 | 4
3 | 2
3 | 5
5 | 4
Edge-centric BFS

Vertices
1
2
3
4
5

Source | Destination
--- | ---
1 | 3
1 | 5
2 | 4
3 | 2
3 | 5
5 | 4
edge-centric properties

- Many sequential scans of the edge list
- The order of edges is irrelevant
- Tradeoff
  - Sequential access is faster
  - More Scatter/Gather iterations
- The number of iterations might be fewer if the edge set >> vertex set
- Problem: still have random access to vertex set
Streaming partitions

- Partition the graph into streaming partitions
  - vertex set: a subset of vertices that fit into RAM
  - edge list: all edges whose source vertex is in the partition’s vertex set
  - update list: all updates whose destination vertex is in the partition’s vertex set

- Streaming partitions can be processed in parallel
- Vertices (random access) => fast storage, Edges (sequential access) => slow storage
- The number of partitions is crucial for performance
- Shuffle phase - updates must be re-arranged after the scatter phase
Scalability

- Increasing thread count
- Increasing number of I/O devices
- Across devices

Traversals algorithms – BFS, WCC
Multiplication algorithms – PageRank, SpMW
Comparison with Other Systems: Ligra

- Ligra
  - In-memory graph processing system
  - Requires pre-processing

<table>
<thead>
<tr>
<th>Threads</th>
<th>Ligra (s)</th>
<th>X-Stream (s)</th>
<th>Ligra-pre (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BFS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>11.10</td>
<td>168.50</td>
<td>1250.00</td>
</tr>
<tr>
<td>2</td>
<td>5.59</td>
<td>86.97</td>
<td>647.00</td>
</tr>
<tr>
<td>4</td>
<td>2.83</td>
<td>45.12</td>
<td>352.00</td>
</tr>
<tr>
<td>8</td>
<td>1.48</td>
<td>26.68</td>
<td>209.40</td>
</tr>
<tr>
<td>16</td>
<td>0.85</td>
<td>18.48</td>
<td>157.20</td>
</tr>
<tr>
<td></td>
<td>Pagerank</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>990.20</td>
<td>455.06</td>
<td>1264.00</td>
</tr>
<tr>
<td>2</td>
<td>510.60</td>
<td>241.56</td>
<td>654.00</td>
</tr>
<tr>
<td>4</td>
<td>269.60</td>
<td>129.72</td>
<td>355.00</td>
</tr>
<tr>
<td>8</td>
<td>145.40</td>
<td>83.42</td>
<td>211.40</td>
</tr>
<tr>
<td>16</td>
<td>79.24</td>
<td>50.06</td>
<td>160.20</td>
</tr>
</tbody>
</table>
Comparison with Other Systems: GraphChi

- GraphChi
- Traditional vertex-centric approach
- Out-of-core data structure, parallel sliding windows, to reduce the amount of random access to disk
- Needs time to pre-sort the graph into shards

<table>
<thead>
<tr>
<th>System</th>
<th>Graphchi (shard)</th>
<th>Graphchi (run)</th>
<th>X-Stream</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LABOS</td>
<td>Cambridge</td>
<td></td>
</tr>
<tr>
<td>Intel SSDs</td>
<td>486 ± 6.762</td>
<td>908.966 ± 16.667</td>
<td>417.213 ± 3.037</td>
</tr>
<tr>
<td>Disk</td>
<td>591.848 ± 19.885</td>
<td>1507 ± 13.656</td>
<td>616.795 ± 2.271</td>
</tr>
<tr>
<td>Samsung 840</td>
<td>389.569 ± 41.879</td>
<td>943.246 ± 19.754</td>
<td>588.613 ± 5.259</td>
</tr>
<tr>
<td>2xSamsung 840</td>
<td>375.729 ± 35.975</td>
<td>811.359 ± 23.706</td>
<td>443.396 ± 40.446</td>
</tr>
<tr>
<td>OCZ Vertex</td>
<td>423.104 ± 5.218</td>
<td>1079.138 ± 20.600</td>
<td>843.023 ± 276.625</td>
</tr>
<tr>
<td>Disk</td>
<td>590.584 ± 55.165</td>
<td>1879 ± 93.368</td>
<td>1613.174 ± 106.151</td>
</tr>
</tbody>
</table>

Table 2: Results for pagerank
Criticism

- Assumes that the number of edges is larger than the number of vertices
- Performs well only on graphs with a low diameter
- Workload imbalance as the partitions can have different numbers of edges assigned to them
  - Is work stealing sufficient?
Thank you!