Interactive Graph Computing and Weibo Data

Wentao Han
Department of Computer Science and Technology,
Tsinghua University
Graph Data

• An important form of unstructured data
• And usually BIG
• Can represent rich data and relationships
  • Traffic networks
  • Web link graphs
  • Social networks
  • Protein interaction graphs
What Does Researchers Do When Doing Graph Analysis?

From the perspective of operations

• ETL (extract, transform, load)
• Run workflows composed of algorithms
• Improve by modifying algorithms or adjusting parameters
Pain Points in Graph Analysis

• Data is big: deploying a distributed processing system is costly (hardware, maintenance, power)
• Algorithms (or parameters) are changing: not only the actual operations, but also loading operations will take a lot of time (I/O bandwidth bottleneck)
• Workflows are changing, too: results from previous steps may affect what to do next
• Others: visualization, etc.
Requirements for Graph Analysis

Interactivity

Extensibility

Performance
Our Solution: Interactive Graph Computing

• Interactivity: provided by underlying Python shell
• Performance: implementing core data structures and algorithms in C++, in in-memory or out-of-core fashion
• Extensibility: provided by Python interpreter, extending in C++ (core code) or Python (glue code)
Single Machine vs. Distributed Systems

Speed
100x less efficient

<table>
<thead>
<tr>
<th>scalable system</th>
<th>cores</th>
<th>twitter</th>
<th>uk-2007-05</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphChi [12]</td>
<td>2</td>
<td>3160s</td>
<td>6972s</td>
</tr>
<tr>
<td>Stratosphere [8]</td>
<td>16</td>
<td>2250s</td>
<td>-</td>
</tr>
<tr>
<td>X-Stream [21]</td>
<td>16</td>
<td>1488s</td>
<td>-</td>
</tr>
<tr>
<td>Spark [10]</td>
<td>128</td>
<td>857s</td>
<td>1759s</td>
</tr>
<tr>
<td>Giraph [10]</td>
<td>128</td>
<td>596s</td>
<td>1235s</td>
</tr>
<tr>
<td>GraphLab [10]</td>
<td>128</td>
<td>249s</td>
<td>833s</td>
</tr>
<tr>
<td>GraphX [10]</td>
<td>128</td>
<td>419s</td>
<td>462s</td>
</tr>
<tr>
<td>Single thread (SSD)</td>
<td>1</td>
<td>300s</td>
<td>651s</td>
</tr>
<tr>
<td>Single thread (RAM)</td>
<td>1</td>
<td>275s</td>
<td>-</td>
</tr>
</tbody>
</table>

Memory
20x less efficient

Memory Usage (GB)
In-memory: Compact Structures

• Not using complex containers like trees or hashmaps
• Represent graph data in plain arrays
• For example, CSR (compressed sparse row) format, space complexity $O(V + E)$
• Choose data width economically, and even use bit compression (trade-off between time and space)
Out-of-core: Layout and Scheduling

- Layout: to increase data locality
- Scheduling: to reduce I/O amount

![Diagram of data layout and scheduling]

Stream Block (2, 2)
Case Study: Weibo Data (Compared to Twitter)
Weibo Data Profile

• We crawled Weibo data from 2012 to 2013
  • VERY DIRTY WORK!
  • Don’t want to and can’t do it again

• The data contains
  • User profiles (222 million)
  • Following relationships (27 billion)
  • Weibo posts (about 10TB in MongoDB’s data format)
  • Largest OSN available for research at that time

• Compared to Twitter data in 2009
In-degree Distribution
Out-degree Distribution
Reciprocity (degree less than 100)
Reciprocity (overall)
Posting Heatmap

Heatmap of Weibo (2012) and Twitter (2009) over the world
Weibo (37 million posts) in magenta, and Twitter (42 million posts) in green
Heatmap of Asia

Maldives
Heatmap of Europe
Heatmap of North America
Weibo Adoption Rate
Connections between Regions

result of running hierarchical agglomerative clustering
Take-away Messages

• Interactive graph computing is in demand.
• Single-machine systems could be great; don’t use distributed systems unless have to.
• Some interesting results from Weibo with Twitter