Scalable Data Analysis (CIS 602-02)

Graph Analytics

Dr. David Koop
Spark Streaming

• Run a streaming computation as a series of very small, deterministic batch jobs (micro-batches)

• Chop up the live stream into batches of $X$ seconds

• Spark treats each batch of data as RDDs and processes them using RDD operations

• Finally, the processed results of the RDD operations are returned in batches

[D. Koop, CIS 602-02, Fall 2015]

[T. Das, Databricks, Spark Streaming]
Twitter Heron

- Issues with Storm @Twitter:
  - Debugging was hard
  - Storm Worker Architecture:
    • Complex, could run multiple tasks
    • Resource allocation was assumed to be homogeneous
  - Storm Nimbus:
    • Topologies run in isolation -> wastes resources
    • Single point of failure
  - Lack of backpressure
  - Efficiency:
    • Suboptimal Relays
    • Garbage Collection
Heron Architecture

A Heron topology is equivalent to 

5.1 Data Model and API

Figure 3: Heron Architecture

5.2 Architecture overview

Figure 4: Heron Topology Architecture

[S. Kulkarni et al., 2015]
Heron Topology Architecture

[Image: Heron Topology Architecture]

[S. Kulkarni et al., 2015]
Heron Instance

![Heron Instance Diagram]

- **Stream Manager (SM)**
- **Metrics Manager (MM)**
- **Gateway Thread**
- **Task Execution Thread**

**Connections:**
- *data-in*
- *data-out*
- *metrics-out*

**Note:** [S. Kulkarni et al., 2015]
7.3 Word Count Topology

In these set of experiments, we used a simple word count topology. In this topology, the spout tasks generate a set of random words (~175k words) during the initial "open" call, and during every "nextTuple" call. In each call, each spout simply picks a word at random and emits it. Hence spouts are extremely fast, if left unrestricted. Spouts use a fields grouping for their output, and each spout could send tuples to every other bolt in the topology.

Bolts maintain an in-memory map, which is keyed by the word emitted by the spout and updates the count when it receives a tuple. This topology is a good measure of the overhead introduced by either Storm or Heron since it does not do significant work in its spouts and bolts.

For each set of experiments, we varied the number of Storm spout/bolt tasks, Heron spout/bolt instances, Storm workers, and Heron containers as shown below in Table 1.

### Table 1: Experimental setup for the Word Count topology

<table>
<thead>
<tr>
<th>Components</th>
<th>Expt. #1</th>
<th>Expt. #2</th>
<th>Expt. #3</th>
<th>Expt. #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spout</td>
<td>25</td>
<td>100</td>
<td>200</td>
<td>500</td>
</tr>
<tr>
<td>Bolt</td>
<td>25</td>
<td>100</td>
<td>200</td>
<td>500</td>
</tr>
<tr>
<td># Storm workers</td>
<td>25</td>
<td>100</td>
<td>200</td>
<td>500</td>
</tr>
<tr>
<td># Heron containers</td>
<td>25</td>
<td>100</td>
<td>200</td>
<td>500</td>
</tr>
</tbody>
</table>

7.3.1 Acknowledgements Enabled

In these experiments, the word count topology is enabled to receive acknowledgements. We measured the topology throughput, end-to-end latency, and CPU usage, and plot these results in Figure 9, Figure 10, and Figure 11 respectively. Each of these figures has four points on each line, corresponding to the four experimental setup configurations that are shown in Table 1.

As shown in Figure 9, the topology throughput increases linearly for both Storm and Heron. However, for Heron, the throughput is 10-14X higher than that for Storm in all these experiments.

The end-to-end latency graph, plotted in Figure 10, shows that the latency increases far more gradually for Heron than it does for Storm. Heron latency is 5-15X lower than that of Storm. There are many bottlenecks in Storm, as the tuples have to travel through multiple threads inside the worker and pass through multiple queues. (See Section 3.)

In Heron, there are several buffers that a tuple has to pass through as they are transported from one Heron Instance to another (via the SMs). Each buffer adds some latency since tuples are transported in batches. In normal cases, this latency is approximately 20ms, and one would expect the latency to be of the same value since the tuples in this topology have the same number of hops. However, in this topology, the latency increases as the number of containers increase. This increase is a result of the SMs becoming a bottleneck, as they need to maintain buffers for each connection to the other SMs, and it takes more time to consume data from more buffers.

The tuples also live in these buffers for longer time given a constant input rate (only one spout instance per container).

Figure 11 shows the aggregate CPU resources that are utilized across the entire cluster that is used for this topology, as reported
Projects

- Goals: understand a dataset, improve an analysis technique
- Report: Structure like a research paper
  - Introduction, Related Work, Description, Results, Discussion, Conclusion
  - May be written as a notebook
  - 8-10 pages
  - Turn in via myCourses
  - Due Dec. 12
- Code: Turn in code via myCourses
- Presentations: Dec. 12, 11:30am
Extra Credit Opportunity

• User Study on a visualization technique
• 15 points toward assignments
• Sign-up sheets available
Assignment 3

- Spark, EC2, and Natural Language Processing
- Use Million Song Dataset
- Sign up for EC2 if you have not already
- Not late until Dec. 12
Spark EC2 Demo (Continued)
Graphs as Data

Nodes

<table>
<thead>
<tr>
<th>ID</th>
<th>Atom</th>
<th>Electrons</th>
<th>Protons</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>C</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>N</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

Edges

<table>
<thead>
<tr>
<th>ID1</th>
<th>ID2</th>
<th>Bonds</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
Social Networks

[P. Butler, 2010]
Graphs

• "Every time someone visits news feed there are on average 1,500 potential stories from friends, people they follow and pages for them to see, and most people don’t have enough time to see them all" - Lars Backstrom, Facebook Engineer, 2013
TAO: Facebook's Distributed Data Store for the Social Graph

TAO

• Reads much more frequent than writes
• Most edge queries have empty results
• Long-tail distribution
GraphX: Unifying Data-Parallel and Graph-Parallel Analytics

J. Gonzalez
Next...

- Reproducibility and Provenance
- Work on Projects!
- A3