Spark

A Fast and General Engine for Large-scale Data Processing

Shangyu Luo
Department of Computer Science
Rice University
Cluster Computing System

- Scalability
- Speed
- Usability
- Fault Tolerance
Outline

- What is Spark
- Why is Spark
- Experiments
- Discussion
- Conclusion and Future Work
What is Spark

- An open source cluster computing system
- Fast data analysis — fast to run and fast to write. (Map-reduce like system)
- Up to 100X faster than Hadoop MapReduce
- Support Scala, Java and Python
Why is Spark

- **Iterative algorithms** (e.g., machine learning) and interactive applications
- Fast
  - Storing data into memory and then reuse it (cache)
  - E.g., Logistic Regression, 1 billion 9 dimension data points: 3s (normally 70s)
- Easy to use
- Fault tolerance
  - Lineage information
Spark Cluster Structure

Driver Node ➔ Partitioning ➔ Tasking ➔ Worker Nodes
val file = spark.textFile("hdfs://...")
val counts = file.flatMap(line => line.split(" "))
  .map(word => (word, 1))
  .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
Experiments

- Models
  - Gaussian mixture model (GMM)
  - Gaussian mixture model with imputation
  - The Bayesian Lasso
  - Hidden Markov model (HMM) for text
  - Latent Dirichlet allocation (LDA)

- Experimental Platform
  - Amazon EC2 m2.4xlarge machines (8 cores, 68 GB of RAM)

- Evaluated Systems
  - Spark, SimSQL, GraphLab and Giraph
### Gaussian Mixture Model

<table>
<thead>
<tr>
<th></th>
<th>10 dimensions</th>
<th>100 dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5 machines</td>
<td>20 machines</td>
</tr>
<tr>
<td>Lines of code</td>
<td>236</td>
<td>26:04</td>
</tr>
<tr>
<td>Spark (python)</td>
<td>737</td>
<td>12:30</td>
</tr>
<tr>
<td>Spark (java)</td>
<td>197</td>
<td>27:55</td>
</tr>
</tbody>
</table>

Figure 1: GMM; Lines of code and average time per iteration. Format is HH:MM:SS or MM:SS. The size of the dataset is ten million data points per machine.
### Bayesian Lasso

<table>
<thead>
<tr>
<th></th>
<th>Lines of code</th>
<th>5 machines</th>
<th>20 machines</th>
<th>100 machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spark (Python)</td>
<td>168</td>
<td>0:55</td>
<td>0:59</td>
<td>1:12</td>
</tr>
<tr>
<td>SimSQL</td>
<td>100</td>
<td>7:09</td>
<td>8:04</td>
<td>12:24</td>
</tr>
</tbody>
</table>

Figure 2: Bayesian; Lines of code and average time per iteration. Format is HH:MM:SS or MM:SS. The model has 1000 regressor dimensions and a one-dimensional response. The size of the dataset is 100,000 data points per machine.
## Hidden Markov Model

<table>
<thead>
<tr>
<th></th>
<th>Word-based, 5 machines</th>
<th>Document-based, 5 machines</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lines of code</td>
<td>Running time</td>
</tr>
<tr>
<td>Spark</td>
<td>NA</td>
<td>Fail</td>
</tr>
<tr>
<td>SimSQL</td>
<td>131</td>
<td>8:17:07</td>
</tr>
<tr>
<td></td>
<td>214</td>
<td>4:21:36</td>
</tr>
<tr>
<td></td>
<td>123</td>
<td>3:42:40</td>
</tr>
</tbody>
</table>

### Super Vertex Implementation

<table>
<thead>
<tr>
<th></th>
<th>Lines of code</th>
<th>5 machines</th>
<th>20 machines</th>
<th>100 machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spark</td>
<td>215</td>
<td>3:45:58</td>
<td>4:01:02</td>
<td>Fail</td>
</tr>
<tr>
<td>SimSQL</td>
<td>136</td>
<td>2:05:12</td>
<td>2:05:31</td>
<td>2:19:10</td>
</tr>
</tbody>
</table>

Figure 3: HMM; Lines of code and average time per iteration. Format is HH:MM:SS or MM:SS. The size of the dictionary is 10,000 words. The number of topics is 20. The size of the dataset is 2.5 million documents per machine.
Discussion

- Fast: GMM, Bayesian Lasso
  - Reuse the same data in the cache in each iteration

- Slow: GMM with imputation, LDA, HMM for text
  - Data is changing in each iteration
  - Shuffling work is time-consuming

- Scalability: Bad for LDA and HMM, good for others.

- Ease-of-programming: Good. Short programs in most of experiments.
Discussion

- **Python vs. Java**
  - Python: Better support for mathematical calculation; Especially vector/matrix manipulation.
  - Java: Generally faster; More choices of operations on RDD.

- **Spark vs. SimSQL**
  - No overall winner in speed.
  - Spark: Easier for programming.
  - SimSQL: Better scalability. Can run almost all five experiments without problems.
Conclusion & Future Work

- Spark is a fast cluster computing platform, especially suitable for iterative and interactive applications.

- Still under development
  - Extensions (SQL, stream, machine learning libraries, graph, etc.)
  - Not so stable (May have unexpected problems)

- Contributors and users are increasing rapidly.

- SimSQL has the potential to improve its performance and usability.