OPTIMIZATION TECHNIQUES WITHIN THE HADOOP ECO-SYSTEM: A SURVEY

Giulia Rumi, Claudia Colella, Danilo Ardagna

Dipartimento di Elettronica, Informazione e Bioingegneria
Hadoop 1.0

Large Shared Cluster

Courtesy of Microsoft
Hadoop 1.0

Problems to solve
- Who runs?
- Where?
- How much resources?
- Order of execution
- Monitor progress
- Handle failures

Large Shared Cluster

Courtesy of Microsoft
• RM and AMs implemented within YARN: Yet Another Resource Negotiator
Why scheduling is important

![Graph showing task slots and time](image)

- Map
- Shuffle
- Sort
- Reduce

- Task slots
- Time (in seconds)
- Figure 1: Sorting with 64 map and 64 reduce slots.

Details of our testbed can be found in Section 4.1.

- Next, we run the sort benchmark with 8GB input on the testbed. As observed from Figures 1 and 2, it is difficult to predict the total number of task slots required for the job completion.

- Let us demonstrate different executions of the same job, the job consists of 64 map tasks.

- Evaluation of the new scheduler shows that the predicted completion times are within 15% of the measured ground truth.

- We validate our approach using a diverse set of realistic applications. The application profiles are stable and the predicted execution times closely match the measured times in the testbed. The new scheduler effectively meets the deadlines for the production job that is routinely executed on a 64-node Hadoop cluster.

- The amount of allocated resources may drastically impact the performance characteristics of the underlying applications. The application profiles are stable and the predicted execution times closely match the measured times in the testbed. The new scheduler effectively meets the deadlines for the production job that is routinely executed on a 64-node Hadoop cluster.
Why scheduling is important
MapReduce Ecosystem
Optimization policies goals

• Data locality

• Sticky slots

• Skewness and Map-Reduce interdependence

• Poor system utilization

• Starvation

• Fairness
Optimization at two layers

- Application level optimization (e.g., Pig and Hive queries)
- Task scheduling optimization (i.e., FIFO scheduler enhancement)
Application level optimization

1. **Job Profiling**
   - Build application invariant

2. **Performance Modeling**
   - Estimate jobs/workflows completion time

3. **Optimization and Scheduling**
   - Allocate map and reduce slots and fulfill deadlines

Aria & Autotune: HP Labs, Lucy Cherkasowa et al., ref. 21, 22, 26, 27
Signature-based approach: K. Kambatla et al., ref. 9
Job Profiling

**Aria & Autotune**
- Past job runs of the whole application
  - OUTPUT: lower, upper, and average estimated completion time
- Execution of a smaller input data set than the original one over the same cluster configuration

**Signature-based**
- Execution of a small fraction of input data on a smaller number of resources
  - OUTPUT: Resource Consumption Signature Set
Performance modeling

Aria & Autotune

\( (M_{min}, M_{avg}, M_{max}, \text{AvgSize}_{input}^{M}, \text{Selectivity}_{M}) \)

\( (Sh_{avg}^{1}, Sh_{max}^{1}, Sh_{typ}^{1}, Sh_{typ}^{2}) \)

\( (R_{typ}, R_{max}, \text{Selectivity}_{R}) \)

Signature-based

Similarity

\( \chi^2(S_{r_1}^{m}, S_{r_2}^{m}) = \sum_{i=1}^{n} \frac{(S_{r_1}^{m_i} - S_{r_2}^{m_i})^2}{(S_{r_1}^{m_i} + S_{r_2}^{m_i})} \)

Vector-distance between application 1 and 2
Optimization and Scheduling

**Aria & Autotune**

\[(M_{\text{min}}, M_{\text{avg}}, M_{\text{max}}, \text{AvgSi})\]

\[(Sh^1_{\text{avg}}, Sh^1_{\text{max}}), (Sh^2_{\text{avg}}, Sh^2_{\text{max}})\]

\[(R_{\text{avg}}, R_{\text{max}}, \text{Selectivity}_R)\]

\[A_j^\text{low} \cdot \frac{N^j_M}{S^j_M} + B_j^\text{low} \cdot \frac{N^j_R}{S^j_R} = T - C_j^\text{low}\]

\[\frac{a}{m} + \frac{b}{r} = D\]

**Signature-based**

\[\chi^2(S^1_m, S^2_m) = \sum_{i=1}^{n} \left( \frac{S^1_m - S^2_m}{S^1_m + S^2_m} \right)^2\]

**Similarity**

Vector-distance between application 1 and 2

**Lagrange Multiplier Method**

Optimum resource allocation already computed in the DB!
Task scheduling optimization

• Hadoop framework executes its task based on **runtime scheduling** scheme

• **Map** and **reduce** tasks can be executed **without communication among other tasks**, no contention and synchronization cost between running jobs

• The first **scheduler** implementation was **First In First Out (FIFO)**

• Hadoop 2.x:
  • **Fair** and **Capacity** schedulers
  • **Work conserving preemption**
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data locality</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sticky slots</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Skewness</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Starvation</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Utilization</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Fairness</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Released/Prototype</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>Hadoop Version</td>
<td>1.0/2.0</td>
<td>1.0/2.0</td>
<td>1.0/2.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Conclusions and future work

• Technology continuously evolving

• Distributed/hierarchical optimization solutions according to YARN architecture

• Integrate batch analysis with data streaming

• Provide a quantitative evaluation of the solutions presented
Thanks for your attention…

… any questions?
Thanks for your attention...

...any questions?