Semantics-Aware Prediction for Analytic Queries in MapReduce Environment

Weikuan Yu, Zhuo Liu, Xiaoning Ding

Florida State University
Auburn University
New Jersey Institute of Technology
**Background**

- MapReduce is a popular data-centric programming model.
- Hive and Pig are popular data warehouse systems.
  - More than 40% of Hadoop jobs at Yahoo! were Pig programs back in 2009, more with Hive now.
  - In Facebook, 95% of MR jobs are generated by Hive.
- In Hive, each SQL query is compiled and translated into a **DAG** (Directed Acyclic Graph) of MapReduce jobs with inner-dependencies.

---

**Diagram:**

- **Hive**
  - JDBC
  - ODBC
  - Command Line Interface
  - Web Interface
  - Thrift Server
  - Metastore
- **Hadoop** (MAP-REDUCE + HDFS)
  - Job Tracker
  - Name Node
  - Data Node + Task Tracker

---

**Source:** ICDE'10 by Facebook
Motivation

Semantic gap: between Hive and Hadoop

- Hadoop is un-aware of such dependency and inter-job relationship, just treating all jobs as the same.
- Without such awareness, it will be difficult for Hadoop to schedule jobs that belong to a query efficiently.

Problems:

- Suboptimal query response time
- Unfairness among queries
Efficiency issue for queries with varied sizes

- In this test, QA, QB and QC are issued in sequence, where QB is a large query.
- Under HCS, interleaved execution happens among queries’ jobs.
Query Delays shown by GANTT Chart

- QA arrives first with its J1 job, QB and QC afterwards with their jobs listed accordingly.
- Query response time can be improved if the scheduler is aware of the query semantics, therefore the relationship among the jobs.
Semantics-Aware Query Prediction

- Three main techniques
  - Semantics extraction (DAG, operator type, predicates, etc.)
  - Selectivity Estimation
  - Query prediction
Selectivity estimation for a query’s jobs

- Selectivity estimation
  - Predict each job node’s intermediate (Med) and output (Out) data sizes recursively along the DAG (from bottom to top).
  - For different job types, e.g., Groupby, Join, Select, we reply on certain formulas and offline-built histograms to estimate their selectivities.

- Logic: Selectivity estimation => Job/query resource estimation and time modeling => Used for efficient query scheduling
Selectivity Estimation

- IS is used for estimating MOF size
  - $IS = \frac{D_{\text{Med}}}{D_{\text{In}}}$

- Final Selectivity (FS) is defined as:
  - $FS = \frac{|\text{Out}|*W_{\text{Out}}}{D_{\text{In}}}$

- Predicate selectivity (ratio of selected rows to input rows)
  - $S_{\text{pred}} = \frac{|\text{Med}|}{|\text{In}|}$

- Projection selectivity (ratio of selected cols to tuple width)
  - $S_{\text{proj}} = \sum \frac{\text{Width}_{\text{col} \text{ i selected}}}{\text{TupleWidth}}$
Intermediate Selectivity - IS

- For extract job such as select and order by,
  $$IS = S_{\text{pred}} \ast S_{\text{proj}}$$

- For join job:
  $$IS = S_{\text{pred} 1} \ast S_{\text{proj} 1} \ast r_1 + S_{\text{pred} 2} \ast S_{\text{proj} 2} \ast (1-r_1)$$

- Groupby can involve local combine: $$IS = S_{\text{comb}} \ast S_{\text{proj}}$$
  - For clustered keys,
    $$S_{\text{comb}} = \min(1, \frac{T.dxy}{|T|*S_{\text{pred}}}) \ast S_{\text{pred}} = \min(S_{\text{pred}}, \frac{T.dxy}{|T|})$$
  - For randomly distributed keys,
    $$S_{\text{comb}} = \min(S_{\text{pred}}, \frac{T.dxy}{|T|/N_{Maps}})$$
Final Selectivity – Output

For extract job,
- For “top k” job, $|\text{Out}| = \min(|\text{In}|, k)$
- For “order by” job, $|\text{Out}| = |\text{In}|$

For groupby job,
- $|\text{Out}| = \min(|T| * S_{\text{pred}}, T.dxy)$

For join job,
- Equ-join with uniform keys:
  \[ |\text{Out}| = |T_1 \bowtie T_2| = |T_1| * |T_2| * \frac{1}{\max(T_1.dx, T_2.dx)} \]
- Chained joins:
  \[ |\text{Out}| = |T_1.\text{pred}_1 \bowtie T_2.\text{pred}_2 \bowtie T_3.\text{pred}_3| = S_{\text{pred}_1} * S_{\text{pred}_2} * S_{\text{pred}_3} \max(|T_1|, |T_2|, |T_3|) \]
An example for selectivity estimation

- Predict jobs’ selectivity recursively in a query.

```sql
SELECT ps_partkey, sum(ps_supplycost*ps_availqty)
FROM nation n JOIN supplier s ON
    s.s_nationkey=n.n_nationkey AND n.n_name<>'CHINA'
JOIN partsupp ps ON
    ps.ps_suppkey=s.s_suppkey
GROUP BY ps_partkey;
```

<table>
<thead>
<tr>
<th>Job 1</th>
<th>Job 2</th>
<th>Job 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>25</th>
<th>24</th>
<th>9600</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pred</td>
<td>MED1</td>
<td>MED1</td>
</tr>
<tr>
<td>n</td>
<td>s</td>
<td>n( \bowtie )s</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>10000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>9600</th>
<th>800000</th>
<th>200000</th>
</tr>
</thead>
<tbody>
<tr>
<td>MED2</td>
<td>MED2</td>
<td>MED1</td>
</tr>
<tr>
<td>ps</td>
<td>ps</td>
<td>n( \bowtie )s ( \bowtie )ps</td>
</tr>
<tr>
<td>800000</td>
<td>800000</td>
<td>768000</td>
</tr>
</tbody>
</table>

\[|\text{Out}| = 0.96 \times 25 \times 10000 \times \frac{1}{\max(25, 25)}\]

\[|\text{Out}| = 0.96 \times \max(25, 10000, 800000)\]

\[|\text{Out}| = \min(768000, 200000)\]
Multivariate Time Prediction

List of Considered Input Features

- Operators
- Input Data
- Output Data
- Data Growth

Table 1: Input Features for the Model

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>The Operator Type: 1 for Join, 0 for others</td>
</tr>
<tr>
<td>D&lt;sub&gt;in&lt;/sub&gt;</td>
<td>The Size of Input Data</td>
</tr>
<tr>
<td>D&lt;sub&gt;avgmed&lt;/sub&gt;</td>
<td>Avg Intermediate Data Per Reduce Task</td>
</tr>
<tr>
<td>D&lt;sub&gt;out&lt;/sub&gt;</td>
<td>The Size of Output Data</td>
</tr>
<tr>
<td>P(1 − P)D&lt;sub&gt;med&lt;/sub&gt;</td>
<td>The Data Growth of Join Operators</td>
</tr>
</tbody>
</table>
Job Time Prediction Model

- Model job execution time based on selectivity estimation
- Training on over 5647 MR jobs, about 1000 queries from TPC-DS and TPC-H of different scales.
- $\theta$ is trained for extract, groupby and join jobs respectively.

\[
P = \frac{\max(|T_1|S_{pred1}, |T_2|S_{pred2})}{|T_1|S_{pred1} + |T_2|S_{pred2}}, \quad 0 < P < 1
\]

\[
ET = \theta_0 + \theta_1 D_{in} + \theta_2 D_{avgmed} + \theta_3 D_{out} + \theta_4 O \times P(1 - P)D_{med}.
\]
Task Time Prediction Model

- Data size:
  - $TD_{In_i}$ and $TD_{Out_i}$

- The predicted time for the $i$-th task: $ET_i$
  - $ET_i = k0 + k1 \cdot TD_{In_i} + k2 \cdot TD_{Out_i} + k3 \cdot P \cdot (1-P) \cdot TD_{In_i}$
Scheduling with Semantics Awareness

Semantics-Aware Resource Demand

- Weight Resource Demand (WRD): aggregate the demand from all map tasks ($MT_i$) and Reduce tasks ($RT_i$)

\[ WRD = \sum (MT_i \times NM_i) + \sum (RT_i \times NR_i) \]

Experimented with a simple greedy scheduling policy

- Prioritizing smallest Queries for fast turnaround
- Smallest WRD First (SWRD) query scheduling
Evaluation setup

- **Benchmarks.**
  - Built with TPC-H, TPC-DS queries and Terasort/Grep/Wordcount MapReduce jobs.
  - Submitted in Poisson interval

**TABLE IV: Workload Composition**

<table>
<thead>
<tr>
<th>Bin</th>
<th>Input Size</th>
<th>Number of Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Bing</td>
</tr>
<tr>
<td>1</td>
<td>1-10 GB</td>
<td>44</td>
</tr>
<tr>
<td>2</td>
<td>20 GB</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>50 GB</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>100 GB</td>
<td>22</td>
</tr>
<tr>
<td>5</td>
<td>&gt;100 GB</td>
<td>2</td>
</tr>
</tbody>
</table>

- **Metrics**
  - Accuracy of the prediction via semantics awareness
  - Efficiency: query execution time
Estimation of Job Execution time

Accuracy
- On average, 13.98% error rate for the test set of jobs.
Estimation of Task Execution

- Map Task Execution Time
  - Join operators lead to lower accuracy

<table>
<thead>
<tr>
<th>Types</th>
<th>R-squared accuracy</th>
<th>Avg Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Join</td>
<td>85.6%</td>
<td>16.27%</td>
</tr>
<tr>
<td>Groupby</td>
<td>92.4%</td>
<td>24.8%</td>
</tr>
<tr>
<td>Extract</td>
<td>92.74%</td>
<td>14.5%</td>
</tr>
<tr>
<td>Together</td>
<td>87.05%</td>
<td>20.5%</td>
</tr>
</tbody>
</table>

- Reduce Task Execution Time

<table>
<thead>
<tr>
<th>Types</th>
<th>R-squared accuracy</th>
<th>Avg Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Join</td>
<td>85.83%</td>
<td>14.23%</td>
</tr>
<tr>
<td>Groupby</td>
<td>98.82%</td>
<td>4.67%</td>
</tr>
<tr>
<td>Extract</td>
<td>90.03%</td>
<td>6.18%</td>
</tr>
<tr>
<td>Together</td>
<td>90.68%</td>
<td>7.4%</td>
</tr>
</tbody>
</table>
Validation of job and query time estimation

- Predicted time accuracy for queries
  - Error rate is 8.3% on average for 22 100G TPC-H queries.
Benefits of Semantics-Aware Scheduling

Execution time of queries

- Compared to HFS, SWRD improves the execution of Bing and Facebook workloads by 44% and 40%, respectively.
- Compared to HCS, SWRD improves by 27.4% and 72.8%, respectively.
Conclusion and Future Work

- Introduced cross-layer semantics extraction and percolation to increase the semantics awareness of the Hadoop job scheduler
- Formalized the estimation of selectivity for intermediate data and final output
- Developed a multivariate prediction model for job and task execution time and validated the accuracy
  - Leveraged semantics awareness for efficient query scheduling in HIVE
- Plan to pursue further integration of semantics awareness in complex query scheduling and other data analytics systems.
Acknowledgement