Dynamic Resource Allocation for Distributed Dataflows

Lauritz Thamsen
Technische Universität Berlin
04.05.2018
Distributed Dataflows

- E.g. MapReduce, SCOPE, Spark, and Flink

- Used for scalable processing in many domains, e.g.
  - Search and data aggregation
  - Relational processing
  - Graph analysis
  - Machine learning

04.05.18
Distributed Dataflow Jobs

Stage

Synchronization barrier

e.g. Map or Filter

e.g. Reduce
Compared to HPC

- High-level programming abstractions
- Comprehensive frameworks and distributed execution engines
- Usable even without extensive knowledge of parallel and distributed programming
Yet Users Need to Allocate Resources
Constraints of Production Jobs

• Users need to reserve resources for runtime targets:
  • Usability constraints,
  • Service level agreements,
  • Cluster reservation only valid for a certain time

• But do not necessarily understand workload dynamics
Two-Fold Problem

• Runtime Prediction: Estimating performance upfront is difficult as it depends on many factors, e.g. programs, data, systems, and infrastructure

• Runtime Variance: Significant performance variance due to data locality, interference, and failures
Given a runtime target for a recurring batch job, how can minimal resources be allocated automatically?
Methods & Results
Dynamic Resource Allocation

Scale-out Modeling

Similarity Matching

Resource Allocation


Dynamic Resource Allocation

Job + Runtime Target

Workload History

Similarity Matching

Scale-out Models

04.05.18 Dynamic Resource Allocation for Distributed Dataflows
Assumptions

- Distributed dataflow jobs with multiple stages
- Dedicated homogeneous clusters
- Recurring batch processing jobs
Scale-out Modeling

• Find a function that predicts the runtimes of a job based on previous runs

Example Job A:

Example Job B:
Parametric Regression

- Non-Negative Least Squares (NNLS) for a model of distributed data-parallel processing [Ernest]

\[
\text{runtime} = \theta_0 + \theta_1 \cdot \left( \frac{1}{\text{containers}} \right) + \theta_2 \cdot \log(\text{containers}) + \theta_3 \cdot \text{containers}
\]
Non-parametric Regression

- Assuming locally defined behavior with Local Linear Regression (LLR)
Automatic Model Selection

• Goal: model with high accuracy when possible, otherwise use a robust fallback

• Approach:
  • Use parametric model for extrapolation
  • Dynamically select prediction model for interpolation with k-fold cross-validation
Evaluation Setup

- Using up to 60 commodity nodes (8 cores, 16 GB RAM) and exemplary distributed dataflow jobs

<table>
<thead>
<tr>
<th>Job</th>
<th>System</th>
<th>Dataset</th>
<th>Input Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>WordCount</td>
<td>Flink</td>
<td>Wiki</td>
<td>250 GB</td>
</tr>
<tr>
<td>TPC-H Query 10</td>
<td>Flink</td>
<td>Tables</td>
<td>200 GB</td>
</tr>
<tr>
<td>K-Means</td>
<td>Flink</td>
<td>Points</td>
<td>50 GB</td>
</tr>
<tr>
<td>Grep</td>
<td>Spark</td>
<td>Wiki</td>
<td>250 GB</td>
</tr>
<tr>
<td>SGD</td>
<td>Spark</td>
<td>Features</td>
<td>10 GB</td>
</tr>
<tr>
<td>PageRank</td>
<td>Spark</td>
<td>Graph</td>
<td>3.4 GB</td>
</tr>
</tbody>
</table>
Cross-Validation of Models

- Repeated random sub-sampling
Cross-Validation of Models

- Repeated random sub-sampling

With increasing amounts of training data, $n = ...$
Relative Prediction Error

Grep (Spark)

SGD (Spark)
Dynamic Resource Allocation

Scale-out Modeling

Similarity Matching

Resource Allocation

Addresses

Problem of Runtime Prediction

Problem of Runtime Variance
Select Executions for Model Training

- Recurring jobs that run on updated data, code, and parameters, yielding different runtimes
- Idea: Similar executions $\rightarrow$ similar runtimes
Different Similarity Measures

- Functions $s \left( \text{ex}_{\text{current}}, \text{ex}_{\text{previous}} \right) \in [0,1]$

<table>
<thead>
<tr>
<th>Measure</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Input size icon]</td>
<td>Offline</td>
</tr>
<tr>
<td>![Scale-out icon]</td>
<td>Offline</td>
</tr>
<tr>
<td>![Runtime of stages icon]</td>
<td>Online</td>
</tr>
<tr>
<td>![Convergence icon]</td>
<td>Online</td>
</tr>
<tr>
<td>![Resource utilization icon]</td>
<td>Online</td>
</tr>
</tbody>
</table>
Evaluation of Similarity Measures

![Graph showing the relationship between similarity and accuracy/share/similarity with data points scattered across the graph. The graph includes lines demonstrating the trend.]
Normalized Similarity Quality

- Accuracy normalized over share of selected points
Evaluation of Similarity Measures

Connected Components

Page Rank

Stochastic Gradient Descent

Average Accuracy

% of Most Similar Runs

Input

Convergence

Network-I/O

04.05.18 Dynamic Resource Allocation for Distributed Dataflows 27
**Dynamic Resource Allocation**

- **Scale-out Modeling**
- **Similarity Matching**
- **Resource Allocation**

**Addresses**
- Problem of Runtime Prediction
- Problem of Runtime Variance
Resource Allocation

- Search for minimal scale-out for a runtime target
Stage-wise Dynamic Allocations
Prototype System: *Ellis*

- Integrated and usable on a per-job basis with YARN
Evaluation Setup

• Using up to 40 commodity nodes (8 cores, 16 GB RAM) and exemplary iterative Spark jobs (MLlib)

<table>
<thead>
<tr>
<th>Job</th>
<th>Dataset</th>
<th>Input Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-Layer Perceptron (MLP)</td>
<td>Multiclass</td>
<td>29 GB</td>
</tr>
<tr>
<td>Gradient Boosted Trees (GBT)</td>
<td>Vandermonde</td>
<td>111 GB</td>
</tr>
<tr>
<td>Stochastic Gradient Descent (SGD)</td>
<td>Tables</td>
<td>37 GB</td>
</tr>
<tr>
<td>K-Means</td>
<td>Points</td>
<td>50 GB</td>
</tr>
</tbody>
</table>
Results for Gradient Boosted Trees

Ellis static (only initially):  

Ellis dynamic (per stage):
Results for All Benchmark Jobs

• Comparison of $\frac{Ellis\ dynamic}{Ellis\ static}$ in three metrics:

<table>
<thead>
<tr>
<th>Job</th>
<th>Violation Count</th>
<th>Violation Sum</th>
<th>Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD</td>
<td>25%</td>
<td>7%</td>
<td>94%</td>
</tr>
<tr>
<td>K-Means</td>
<td>35%</td>
<td>5%</td>
<td>80%</td>
</tr>
<tr>
<td>GBT</td>
<td>47%</td>
<td>29%</td>
<td>100%</td>
</tr>
<tr>
<td>MLP</td>
<td>69%</td>
<td>13%</td>
<td>127%</td>
</tr>
</tbody>
</table>
Dynamic Resource Allocation for Distributed Dataflows

- **Problem of Runtime Prediction**
- **Problem of Runtime Variance**

### Scale-out Modeling

### Similarity Matching

### Resource Allocation

Addresses
Summary

• New methods for dynamic resource allocation that are applicable to distributed dataflow frameworks

• Prototype integrated and usable with YARN

• Promising evaluation results
References


