Online Parameter Optimization for Elastic Data Stream Processing

Thomas Heinze, Lars Roediger, Andreas Meister, Yuanzhen Ji, Zbigniew Jerzak, Christof Fetzer

August 28, 2015
SCALING POLICY FOR ELASTIC SCALING

• Scaling Policy decides when and how the system scales in/out
  • E.g. user-defined thresholds

• Alternatives:
  ▪ **Auto-scaling Techniques**\(^1\): Controller, Reinforcement Learning, Load Prediction, Queuing Theory
  ▪ **Sampling-based Approaches**\(^2\) for job-based Systems (e.g. MapReduce)

---


ELASTIC DATA STREAM PROCESSING

- Long standing continuous queries over potentially infinite data stream
ELASTIC DATA STREAM PROCESSING

- Long standing continuous queries over potentially infinite data stream

- Unpredictable workload, high variability
  sampling or load prediction doesn’t work

- Stateful, in-memory computation with strict requirements on end to end latency
  limited support by auto-scaling techniques[1]

CONCEPT

Elastic Data Stream Processing Engine

Online Profiler

Parameter Optimization

Distributed Data Stream Processing Engine

Threshold-based Scaling Strategy

Elasticity Manager
PARAMETER OPTIMIZATION

Utilization $util$

**Current Parameter Config $p$**

- $p$, $util$
- $cost(p)$

**Search Algorithm**

- Best Found Config $p^*$, $cost(p^*)$

**Cost Function**

- $p_i$, $util$
- $cost(p_i)$

**Adapt?**

- Yes
  - Adapt to Parameter Config $p^*$
- No
  - No Adaptation

Parameter Optimization

$p$, $cost(p)$
EXAMPLE: COST FUNCTION

INPUT

<table>
<thead>
<tr>
<th>Utilization Statistics</th>
<th>Assignment</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_0$ 0.4 $t_1$ 0.5 $t_2$ 0.4 $t_3$ 0.3</td>
<td>H1 {Op1, Op2} 0.7</td>
<td>thres↓ 0.3</td>
</tr>
<tr>
<td>Op1 0.4</td>
<td>H2 {Op3} 0.3</td>
<td>thres↑ 0.8</td>
</tr>
<tr>
<td>Op2 0.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Op3 0.3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CALCULATION

<table>
<thead>
<tr>
<th>$t_0$</th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ops</td>
<td>Σutil</td>
<td>ops</td>
<td>Σutil</td>
</tr>
<tr>
<td>H1</td>
<td>{Op1, Op2} 0.7</td>
<td>H1</td>
<td>{Op1} 0.5</td>
</tr>
<tr>
<td>H2</td>
<td>{Op3} 0.3</td>
<td>H2</td>
<td>{Op3} 0.4</td>
</tr>
<tr>
<td>H3</td>
<td>{Op2} 0.5</td>
<td>H3</td>
<td>{Op2} 0.4</td>
</tr>
</tbody>
</table>
EVALUATION SETUP

- 12 hosts, duration per measurement: ca. 90 min.

- Measurability violations

\[ \text{lat}_{\text{Thres}} \]
EVALUATION

The image shows a grid of scatter plots comparing Monetary Cost ($) on the x-axis with Latency Violations (#) on the y-axis for different categories such as Financial Day 1, Financial Day 2, Financial Day 3, Twitter Week 1, Twitter Week 2, Twitter Week 3, Energy Week 1, Energy Week 2, and Energy Week 3. Each category has two data points labeled 'Manual' and 'Optimized' represented by blue and red triangles, respectively. The plots illustrate the comparison between manual and optimized scenarios across different weeks and days.
EVALUATION

Param Opt vs. Thresholds:
- Naive: Cost: -19% Lat. Violations: -0.5
- Best: Cost: -10% Lat. Violations: +3

Param Opt vs. Reinforcement Learning:
- Naive: Cost: +6% Lat. Violations: -14
- Best: Cost: +7% Lat. Violations: -6
SUMMARY

- Defining Scaling Policy is error-prone and time-consuming
- **Auto-scaling Techniques** do not support many data stream processing system specifics
- **Online Parameter Optimization** improves trade-off between latency and monetary cost
  - Enhanced modeling of scaling behaviour
  - Allows to react to workload changes