Speedup your Analytics: Automatic Parameter Tuning for Databases and Big Data Systems

Jiaheng Lu, University of Helsinki
Yuxing Chen, University of Helsinki
Herodotos Herodotou, Cyprus University of Technology
Shivnath Babu, Duke University / Unravel Data Systems
Outline

Motivation and Background
History and Classification
Parameter Tuning on Databases
Parameter Tuning on Big Data Systems

**Applications of Automatic Parameter Tuning**

Open Challenges and Discussion
Auto Parameter Tuning in Database Systems

➢ **Oracle Self-driving Database**
   - Automatically set various memory parameters and use of compression using machine learning

➢ **IBM DB2 Self-tuning Memory Manager**
   - Dynamically distributes available memory resources among buffer pools, locking memory, package cache, and sort memory

➢ **Azure SQL Database Automatic Tuning**
   - Memory buffer settings, index management, plan choice correction
Auto Parameter Tuning in Big Data Systems

- **Databricks Optimized Autoscaling**
  - Automatically scale number of executors in Spark up and down

- **Spotfire Data Science Autotuning**
  - Automatically set Spark parameters for number and memory size of executors

- **Sparklens: Qubole’s Spark Tuning Tool**
  - Automatically set memory of Spark executors
Auto Parameter Tuning with Unravel
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>spark.driver.cores</td>
<td>2</td>
</tr>
<tr>
<td>spark.executor.cores</td>
<td>10</td>
</tr>
<tr>
<td>spark.sql.shuffle.partitions</td>
<td>300</td>
</tr>
<tr>
<td>spark.sql.autoBroadcastJoinThreshold</td>
<td>20MB</td>
</tr>
<tr>
<td>SKEW('orders', 'o_custId')</td>
<td>true</td>
</tr>
<tr>
<td>spark.catalog.cacheTable(&quot;orders&quot;)</td>
<td>true</td>
</tr>
</tbody>
</table>

Today, tuning is often by trial-and-error
A New World

INPUTS
1. App = Spark Query
2. Goal = Speedup

“I need to make this app faster”
A New World

In blink of an eye, user gets recommendations to make the app 30% faster.

As user finishes checking email, she has a verified run that is 60% faster.

User comes back from lunch. A verified run that is 90% faster!
Tuning Database Configuration Parameters with iTuned

Songyun Duan, Vamsidhar Thummala, Shivnath Babu*
Department of Computer Science
Duke University
Durham, North Carolina, USA
{syduan,vamsi,shivnath}@cs.duke.edu

ABSTRACT
Database systems have a large number of configuration parameters that control memory distribution, I/O optimization, costing of query plans, parallelism, many aspects of logging, recovery, and other components. For lack of better understanding, many systems have default values for these parameters. When the system is not tuned, performance suffers. Unfortunately, it is common for database tuning parameters to interact in complex ways, making the process of empirical parameter tuning challenging.

Xplus: A SQL-Tuning-Aware Query Optimizer

Herodotos Herodotou and Shivnath Babu*
Department of Computer Science
Duke University
{hero,shivnath}@cs.duke.edu

ABSTRACT
The need for efficient query execution plan discovery and selection is often hampered by the inability of current query optimizers to capture key database properties such as memory distribution, I/O cost, or parallelism. In this work, we introduce Xplus, a SQL-tuning-aware query optimizer that translates the SQL query execution plan discovery problem into a reinforcement learning (RL) task.

Response Surface Methodology

Reinforcement Learning
Autotuning Workflow

Recommendation Algorithm

Monitoring Data

App, Goal

Probe Algorithm

Orchestrator

Cluster Services On-premises and Cloud

$X_{next}$

Historic Data & Probe Data
Outline

Motivation and Background
History and Classification
Parameter Tuning on Databases
Parameter Tuning on Big Data Systems
Applications of Automatic Parameter Tuning

Open Challenges and Discussion
Putting it all Together

<table>
<thead>
<tr>
<th>Approach</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost Modeling</td>
<td>Very efficient for predicting performance</td>
<td>Hard to capture complexity of system internals &amp; pluggable components (e.g., schedulers)</td>
</tr>
<tr>
<td></td>
<td>Good accuracy in many (not complex) scenarios</td>
<td>Models often based on simplified assumptions</td>
</tr>
<tr>
<td></td>
<td>Very efficient for predicting performance</td>
<td>Not effective on heterogeneous clusters</td>
</tr>
<tr>
<td></td>
<td>Good accuracy in many (not complex) scenarios</td>
<td></td>
</tr>
<tr>
<td>Simulation-based</td>
<td>High accuracy in simulating dynamic system behaviors</td>
<td>Hard to comprehensively simulate complex internal dynamics</td>
</tr>
<tr>
<td></td>
<td>Efficient for predicting fine-grained performance</td>
<td>Unable to capture dynamic cluster utilization</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Not very efficient for finding optimal settings</td>
</tr>
<tr>
<td>Experiment-driven</td>
<td>Finds good settings based on real test runs on real systems</td>
<td>Very time consuming as it requires multiple actual runs</td>
</tr>
<tr>
<td></td>
<td>Works across different system versions and hardware</td>
<td>Not cost effective for ad-hoc analytics applications</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Machine Learning</td>
<td>Ability to capture complex system dynamics</td>
<td>Requires large training sets, which are expensive to collect training</td>
</tr>
<tr>
<td></td>
<td>Independence from system internals and hardware</td>
<td>Training from history logs leads to data under-fitting</td>
</tr>
<tr>
<td></td>
<td>Learning based on real observations of system performance</td>
<td>Typically low accuracy for unseen analytics applications</td>
</tr>
<tr>
<td>Adaptive</td>
<td>Finds good settings based on actual task runs</td>
<td>Only applies to long-running analytics applications</td>
</tr>
<tr>
<td></td>
<td>Able to adjust to dynamic runtime status</td>
<td>Inappropriate configuration can cause issues (e.g., stragglers)</td>
</tr>
<tr>
<td></td>
<td>Works well for ad-hoc analytics applications</td>
<td>Neglects efficient resource utilization in the whole system</td>
</tr>
</tbody>
</table>

No single approach is able to provide good prediction accuracy with low overhead in most scenarios.
Open Challenges

Clusters are becoming heterogeneous in nature, both for compute and storage

The proliferation of Cloud leads to multi-tenancy, overheads, perf interaction issues

Real-time analytics pushes boundaries on latency requirements and combination of systems

Ensuring good and robust system performance at scale poses new challenges
Thank you!