Dynamic Workload Management for Very Large Data Warehouses

Juggling Feathers and Bowling Balls

Stefan Krompass
Alfons Kemper
TU München
Germany

Harumi Kuno
Umeshwar Dayal
HP Labs, Palo Alto, CA
USA
Outline

• Problem statement
• Proposed solution
• Evaluation
  – Approach and settings for experiments
  – Impact of problem queries on a workload
  – Impact of execution control
• Conclusion and ongoing work
Background

• HP has been building NeoView, a highly-parallel database engine for business intelligence

• Challenges for DBAs
  – How long should they wait to kill an unexpectedly long-running query?
  – When should they admit a newly arriving query if the currently executing batch of queries is in danger of missing its deadline?
  – What if the newly arrived query was submitted by the CEO?

➔ Automate workload management
Why BI Workloads Differ from OLTP Workloads

• Complexity
• Resource demands
• Different types of queries
• Unpredictability
Why BI Workloads Differ from OLTP Workloads

- Complexity
- Resource demands
- Different types of queries
- Unpredictability
- Problem queries
- Objectives
Vision: Automate Workload Management

Our approach

• Optimize execution of workload subject to service level objectives
• Explicitly consider “problem” queries as an inherent part of the workload
• Propose an architecture that allows us to …
  – … model problem queries with different characteristics
  – … implement and test workload management actions for dealing with problem queries based on their observed behavior
Outline

• Problem statement
• Proposed solution
• Evaluation
  – Implementation and settings for experiments
  – Impact of problem queries on a workload
  – Impact of execution control
• Conclusion and ongoing work
Workload Management Architecture

- Client
- Job (Queries)
- Service Level Objective
- DBMS Engine
  - Query Optimizer
  - Execution Engine
  - Performance statistics
- Workload Manager
  - Admission Control
  - Scheduling
  - Execution Control

Technische Universität München + Hewlett Packard Laboratories
Service Level Objectives and Jobs

Client

Service Level Objective

Job (Queries)

DBMS

Workload Manager
- Admission Control
- Scheduling
- Execution Control

DBMS Engine
- Query Optimizer
- Execution Engine
- Performance statistics
Service Level Objectives (SLOs)

• Job-facing SLOs (e.g., penalty functions used to optimize the scheduling of queries)

• Customer-facing SLOs
  – Minimize response time (derived from “challenges”)
  – Deadline-driven
  – Concrete quantities of computing time
Job Types

• Batch (e.g., reports)
  – Usually repetitive
  – All queries arrive at the database system at once
  – Queries may/may not have precedence constraints
  – SLO is deadline driven

• Interactive (e.g., business analysis)
  – All queries arrive at the database sequentially
  – Arrival time of the first query is not known in advance
  – SLO ("ASAP")
    • Submitted by a special request for business reasons
Execution Engine

Client → Service Level Objective → Job (Queries)

DBMS

Workload Manager
- Admission Control
- Scheduling
- Execution Control

DBMS Engine
- Query Optimizer
- Execution Engine
- Performance statistics
Workload Manager

- Admission Control
- Scheduling
- **Execution Control**
  - Set of actions that apply when certain conditions hold
  - Example:

```plaintext
IF relDBTime IS high AND progress IS low
THEN cancel IS applicable
```
Workload Manager

• Admission Control
• Scheduling
• **Execution Control**
  – Set of actions that apply when certain conditions hold
  – Example:

  $$\text{IF } \text{relDBTime IS high AND progress IS low}$$
  $$\text{THEN cancel IS applicable}$$
Monitored Metrics

• Relative database time (derived from elapsed time of queries and processing time estimates)
• Query progress (derived from progress indicator)
• Number of cancellations
• Resource contention
• Priority
Monitored Metrics

Execution control

Fuzzy Controller

**IF** relDBTime **IS** high **AND**
progress **IS** low
**THEN** cancel **IS** applicable

DBMS Engine

- Query Optimizer
- Execution Engine
- Performance statistics

Estimates

Actions

Monitored Data
Outline

• Problem statement
• Proposed solution
• Evaluation
  – Implementation and settings for experiments
  – Impact of problem queries on a workload
  – Impact of execution control
• Conclusion and ongoing work
Implementation

- Use simulated execution engine instead of real database system installation
  - Inject problem queries
  - Real workloads can take days to process

Commerical Workload → Commercial Database Installation → DB Output files → Knob Settings

- Number of queries in a job
- Number of jobs in a workload
- Number of problem queries
- …

Resource consumption / performance statistics
Settings for Experiments

- **Interactive job**
  - ~1100 feathers
  - Queries arrive sequentially
    - Inter-arrival time 0
    - Does not span entire workload interval

- **Batch job**
  - ~1700 feathers, baseballs, and bowling balls
  - Average execution time of batch queries ~1000 times higher than execution time of interactive queries

*derived from commercial workload runs*
Settings for Experiments

• Normal workload
  – Interactive and batch job executed in parallel
  – No problem queries

• Problem workload
  – Interactive and batch job executed in parallel
  – Problem queries injected into batch workload (75 queries with different “stretch factors”)

Estimated execution time
Actual execution time

Time
Settings for Experiments

• Normal workload
  – Interactive and batch job executed in parallel
  – No problem queries

• Problem workload
  – Interactive and batch job executed in parallel
  – **Problem queries** injected into batch workload (75 queries with different “stretch factors”)
  – Problem queries have a probability for showing the problem behavior after restarting them

• Admit interactive queries first
Admission Control: Admit Interactive First

Queue for interactive queries

Queue for batch queries

Admit query

Execution engine
Admission Control: Admit Interactive First

Queue for interactive queries

Queue for batch queries

Admit query

Execution engine
Outline

• Problem statement
• Proposed solution
• Evaluation
  – Implementation and settings for experiments
  – Impact of problem queries on a workload
  – Impact of execution control
• Conclusion and ongoing work
Impact of Problem Queries on Batch Job

![Graph showing impact of parallelism and thrashing on elapsed time with multi-programming level.]

- **Parallelism**
- **Thrashing**
Impact of Problem Queries on Batch Job

![Graph showing the impact of problem queries on batch job](image)

“Stretched” queries

Elapsed time vs. Multi-programming level graph.
Impact of Problem Queries on Interactive Job

![Graph showing the impact of problem queries on interactive job wait time.](graph.png)
Impact of Problem Queries on Interactive Job

![Graph showing the impact of problem queries on interactive job execution engine](image)
Outline

• Problem statement
• Proposed solution
• Evaluation
  – Implementation and settings for experiments
  – Impact of problem queries on a workload
  – Impact of execution control
• Conclusion and ongoing work
Workload Management Policies

- Fix the MPL at 5
- Varying aggressiveness
  - If query exceeds estimated database time, take action
    
    \[
    \text{relative database time} = \frac{\text{actual database time}}{\text{estimated database time}}
    \]
  - If query is almost finished, do not execute action
- Queries identified as problems are killed and immediately resubmitted ("cancel")
- Canceled queries get two more chances to run to completion
- If queries do not complete, they are killed ("aborted")
Impact of Workload Management Actions

• Batch job: Reduce elapsed time by 81% (problem queries)
• Interactive job: Reduce wait time by 67% (wait time)
• But…
False Positives Lead to Unnecessary Actions
Relative Database Time
False Positives Lead to Unnecessary Actions
Relative Database Time

![Graph showing the relationship between False Positives and Aggressiveness (Relative Database Time)]
False Positives Lead to Unnecessary Actions

Relative Database Time

The graph shows the number of actions taken as a function of aggressiveness, which is measured as a ratio of relative database time. The x-axis represents the aggressiveness level, while the y-axis shows the number of actions. The graph includes three lines: black for false positives, yellow for cancel actions, and red for abort actions. The data indicates how false positives lead to unnecessary actions as the aggressiveness increases.
Number of False Positives and Actions Executed

Reduce number of false positives
Elapsed Time for Batch and Interactive Jobs
Elapsed Time for Batch and Interactive Jobs

Increased elapsed time (queries are restarted over and over again)
Elapsed Time for Batch and Interactive Jobs

Decreased elapsed time (wait time for queries is reduced)
Outline

• Problem statement
• Proposed solution
• Evaluation
  – Implementation and settings for experiments
  – Impact of problem queries on a workload
  – Impact of execution control
• Conclusion and ongoing work
Conclusion

• We implemented a workload management test bed

• Our experiments show that …
  – … even few problem queries have a significant impact on the execution of a mixed workload
  – … the number of false positives leads to an increase in execution time

• Lessons we learned
  – Applying actions too aggressively leads to unnecessary actions
  – Use controller and adjust parameters to right level of aggression
Ongoing Work

- Evaluate impact of admission control and scheduling of BI workloads
- Model query execution on a more detailed level
- Model additional problem types
- Evaluate new workload management techniques
Any Questions?