Maximize MicroStrategy Speed and Throughput with High Performance Tuning

Jochen Demuth, Director Partner Engineering
Maximize MicroStrategy Speed and Throughput with High Performance Tuning

Agenda

1. Introduction to High Performance BI
2. Techniques that Optimize Caching Performance
3. Techniques for Designing In-Memory Cubes
4. Conclusion / Q&A
Introduction to High Performance BI

There is a “Computational Distance” from Raw Data to Finished Report

Total Time: 40 Seconds

“Computational Distance” from Transaction Data to Finished Report
Introduction to High Performance BI

BI Performance is Dominated by Query Time in the Database

“Computational Distance” from Transaction Data to Finished Report

- Transaction Tables
- Transaction Data
- Database
- BI Platform
- Browser
- Transaction Data

1 Second
1 Second
3 Seconds
1 Second
34 Seconds
Total Time: 40 Seconds
Introduction to High Performance BI

Caching Can Dramatically Reduce the Computational Distance

1-2 Seconds

10% of all Reports

Browser

BI Platform

Database

Transaction Tables

Transaction Data

Render

Transmit

Format

Assemble

Predictive Calcs.

Aggregation

Consolidation

Filters

Metric Calcs.

Transmit

Aggregate

Select & Join

#mstrworld

MicroStrategy
Introduction to High Performance BI

DB Aggregates Also Reduce the Computational Distance

- Caching
  - 10% of all Reports

- Predictive Calcs.
- Aggregation
- Consolidation
- Filters
- Metric Calcs.

- DB Aggregates
  - 25% of all Reports

- Render
- Transmit
- Format
- Assemble

- BI Platform

- Browser

- Aggregate
- Select & Join

- Database

- Transaction Tables

- Transaction Data

- 1-2 Seconds
- 2-20 Seconds
Introduction to High Performance BI

“In-Memory Cubes” Can Reduce the Computational Distance for Many Reports

Browser

Render
Transmit
Format
Assemble
Predictive Calcs.
Aggregation
Consolidation
Filters
Metric Calcs.
Transmit

BI Platform

1-2 Seconds
1-3 Seconds
2-20 Seconds

Caching
10% of all Reports

In-Memory Cubes
40%-60% of all Reports

DB Aggregates
25% of all Reports

Transaction Data

Transaction Tables
Introduction to High Performance BI

Benchmark Lab Tests Show that Caching Can Deliver 10x Faster Performance

MicroStrategy’s Caching Architecture Delivers Sub 2 Second Response at High User Loads

Test Configuration:
- Intelligence Server
  4 CPUs Xeon
  16 GB Memory
- Web Server
  4 CPUs Xeon
  8 GB Memory
Introduction to High Performance BI
In-Memory Shifts the Query Work from Databases to In-Memory Sources

**BEFORE**

Performance Profile
*Typical Operation*

- Direct DB Queries
- Cache Hit

**AFTER**

Performance Profile
*In-memory Operation*

- Direct DB Queries
- In-memory Queries
- Cache Hit
Blazing Fast Performance: Tuning Tips and Tricks

Agenda

1. Introduction to High Performance BI
2. Techniques that Optimize Caching Performance
3. Techniques for Designing In-Memory Cubes
4. Conclusion / Q&A
# Introduction to Caching

MicroStrategy’s Layered Caching Architecture Accelerates Performance

<table>
<thead>
<tr>
<th>Cache Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document Cache</td>
<td>Accelerates dashboard performance by storing the final formatted output in memory – Flash, HTML, PDF, Excel</td>
</tr>
<tr>
<td>Results Cache</td>
<td>Accelerates about <strong>10%</strong> of all queries including dashboard datasets by storing report results in-memory</td>
</tr>
<tr>
<td>Element Cache</td>
<td>Accelerates attribute element requests for prompting. <strong>20-40%</strong> of all database queries are used to populate prompt element lists.</td>
</tr>
<tr>
<td>Object Cache</td>
<td>Accelerates metadata object requests for rapid assembly into queries for reports.</td>
</tr>
<tr>
<td>DB Connection Cache</td>
<td>Accelerates connections to the database to ensure fast query submissions</td>
</tr>
</tbody>
</table>

CACHE MANAGER

Identifies Fastest Way to Fulfill Queries
Introduction to Caching

CachingEliminates Multiple DB Queries in theReport Execution Workflow

A Single Report May Require Many Database Queries

User Runs Report

MicroStrategy

Query #1

Object Definitions for Report Construction

Database
Introduction to Caching
Caching Eliminates Multiple DB Queries in the Report Execution Workflow

A Single Report May Require Many Database Queries

User Runs Report

MicroStrategy

Query #1
Object Definitions for Report Construction

Query #2
Element (Lookup) Data to Populate Prompts

Database
Introduction to Caching

Caching Eliminates Multiple DB Queries in the Report Execution Workflow

A Single Report May Require Many Database Queries

User Runs Report

MicroStrategy

Database

Query #1
Object Definitions for Report Construction

Query #2
Element (Lookup) Data to Populate Prompts

Query #3
Report Data For Final Results
Introduction to Caching

Caching Eliminates Multiple DB Queries in the Report Execution Workflow

A Single Report May Require Many Database Queries

User Runs Report

Final Rendering of Report Data

MicroStrategy

Query #3
Report Data for Final Results

Query #2
Element (Lookup) Data to Populate Prompts

Query #1
Object Definitions for Report Construction

Database
Introduction to Caching

Caching Eliminates Multiple DB Queries in the Report Execution Workflow

A Single Report May Require Many Database Queries

MicroStrategy’s Layer Caching Can Eliminate Database Queries for Faster Performance

Object Cache

Database

User Runs Report

MicroStrategy

Final Rendering of Report Data

Query #3
Report Data for Final Results

Query #2
Element (Lookup) Data to Populate Prompts

Query #1
Object Definitions for Report Construction

Database

Object Cache

Object Definitions for Report Construction

#mstrworld
**Introduction to Caching**

Caching Eliminates Multiple DB Queries in the Report Execution Workflow

**A Single Report May Require Many Database Queries**

- User Runs Report
- Final Rendering of Report Data

**Query #1**
- Object Definitions for Report Construction

**Query #2**
- Element (Lookup) Data to Populate Prompts

**Query #3**
- Report Data for Final Results

**MicroStrategy’s Layer Caching Can Eliminate Database Queries for Faster Performance**

- User Runs Report
- Final Rendering of Report Data

**Element Cache**
- Element (Lookup) Data to Populate Prompts

**Object Cache**
- Object Definitions for Report Construction

Database
Introduction to Caching

Caching Eliminates Multiple DB Queries in the Report Execution Workflow

A Single Report May Require Many Database Queries

- User Runs Report
- Database
- MicroStrategy
- Final Rendering of Report Data
- Object Cache
  - Object Definitions for Report Construction
- Query #1
  - Query #2
    - Element (Lookup) Data to Populate Prompts
- Query #3
  - Report Data for Final Results

MicroStrategy’s Layer Caching Can Eliminate Database Queries for Faster Performance

- User Runs Report
- Database
- MicroStrategy
- Results Cache
  - Report Data for Final Results
- Element Cache
  - Element (Lookup) Data to Populate Prompts
- Object Cache
  - Object Definitions for Report Construction
Introduction to Caching

Caching Eliminates Multiple DB Queries in the Report Execution Workflow

A Single Report May Require Many Database Queries

- **Query #1**: Object Definitions for Report Construction
- **Query #2**: Element (Lookup) Data to Populate Prompts
- **Query #3**: Report Data for Final Results

MicroStrategy’s Layer Caching Can Eliminate Database Queries for Faster Performance

- **Document Cache**: Final Rendering of Report Data
- **Results Cache**: Report Data for Final Results
- **Element Cache**: Element (Lookup) Data to Populate Prompts
- **Object Cache**: Object Definitions for Report Construction

Database
### Introduction to Caching

Caching Can Dramatically Speed-up the Typical User Experience, up to 6x Faster

<table>
<thead>
<tr>
<th>Typical User Actions</th>
<th>Processing Steps</th>
<th>Processing Action</th>
<th>No caching</th>
<th>w/Object Caching</th>
<th>Object + Element Caching</th>
<th>Object + Element + Result Caching</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login</td>
<td>Verify Credentials</td>
<td>Object query</td>
<td>2s</td>
<td>&lt;1s</td>
<td>0s</td>
<td>0s</td>
</tr>
<tr>
<td></td>
<td>Load initial page content</td>
<td>Object query</td>
<td>2s</td>
<td>&lt;1s</td>
<td>0s</td>
<td>0s</td>
</tr>
<tr>
<td></td>
<td>Display page</td>
<td>Rendering</td>
<td>1s</td>
<td>1s</td>
<td>1s</td>
<td>1s</td>
</tr>
<tr>
<td>Navigate</td>
<td>Lookup folder content</td>
<td>Object query</td>
<td>4s</td>
<td>&lt;1s</td>
<td>0s</td>
<td>0s</td>
</tr>
<tr>
<td></td>
<td>Display page</td>
<td>Rendering</td>
<td>1s</td>
<td>1s</td>
<td>1s</td>
<td>1s</td>
</tr>
<tr>
<td>Run Report</td>
<td>Load report definition</td>
<td>Object query</td>
<td>10s</td>
<td>1s</td>
<td>1s</td>
<td>1s</td>
</tr>
<tr>
<td></td>
<td>Load prompt content</td>
<td>Element query</td>
<td>10s</td>
<td>10s</td>
<td>1s</td>
<td>1s</td>
</tr>
<tr>
<td></td>
<td>Display prompt page</td>
<td>Rendering</td>
<td>1s</td>
<td>1s</td>
<td>1s</td>
<td>1s</td>
</tr>
<tr>
<td>Type Search term into prompt</td>
<td>Execute search</td>
<td>Element query</td>
<td>2s</td>
<td>2s</td>
<td>2s</td>
<td>2s</td>
</tr>
<tr>
<td></td>
<td>Display updated prompt page</td>
<td>Rendering</td>
<td>1s</td>
<td>1s</td>
<td>1s</td>
<td>1s</td>
</tr>
<tr>
<td>Submit prompt answer</td>
<td>Load report definition</td>
<td>Object query</td>
<td>10s</td>
<td>1s</td>
<td>1s</td>
<td>1s</td>
</tr>
<tr>
<td></td>
<td>Execute multi-pass SQL</td>
<td>Data query</td>
<td>30s</td>
<td>30s</td>
<td>30s</td>
<td>1s</td>
</tr>
<tr>
<td></td>
<td>Display query results</td>
<td>Rendering</td>
<td>2s</td>
<td>2s</td>
<td>2s</td>
<td>2s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Object query</td>
<td>28s</td>
<td>37%</td>
<td>3s</td>
<td>6%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Element query</td>
<td>12s</td>
<td>16%</td>
<td>12s</td>
<td>24%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data query</td>
<td>30s</td>
<td>39%</td>
<td>30s</td>
<td>59%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rendering</td>
<td>6s</td>
<td>8%</td>
<td>6s</td>
<td>12%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Object query</td>
<td>28s</td>
<td>37%</td>
<td>3s</td>
<td>6%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Element query</td>
<td>12s</td>
<td>16%</td>
<td>12s</td>
<td>24%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data query</td>
<td>30s</td>
<td>39%</td>
<td>30s</td>
<td>59%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rendering</td>
<td>6s</td>
<td>8%</td>
<td>6s</td>
<td>12%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total</td>
<td>76s</td>
<td>100%</td>
<td>51s</td>
<td>100%</td>
</tr>
</tbody>
</table>

**Time Savings:**
- **1.5x Faster**
- **~2x Faster**
- **6x Faster**
Introduction to Caching

An Effective Cache Strategy Should Accommodate 90%-95% of Object and Element Queries and 10%-20% of Result Queries

MicroStrategy Application w/o Caching Issues
Significant Amount of Queries to Databases

A Well Tuned Caching Design Should Answer 90% of Object and Element Queries and 10% of Result Queries
Introduction to Caching

Result Caching Accommodates Only 10%-20% of Reports Due to Unique Security Requirements and Prompt Answers

- Different Users
  - Run Same Report
    - Different Prompt Answers
      - User Specific Data Security
        - Unique Prompt Answers and Security cannot share Result Caches
        - Matching Prompt Answers And Data Security can share Result Caches
Improving Result Caching Performance

Tip 1: Reserve Sufficient Memory For All Result Caches to Avoid Reloading Caches

Insufficient Memory reserved for caches causes reloading of cache files from disk.

Allocating sufficient memory eliminates repeated loading of caches.
Improving Result Caching Performance

Tip 2: Avoid Swapping Caused by Allocating Too Much Memory For Result Caches

Reserving too much memory for result caches can cause Swapping to Virtual Memory or disk which slows down all operations. The faster option is to reserve less memory for caches.

Memory Allocated for BI Server

Memory used for result caches

Memory is 1000x Faster than Disk
Improving Result Caching Performance

Tip 3: Turn Off Caching For Reports And Documents With Low Cache Hit Ratios

Not caching reports with low cache hit ratios frees up memory other caches.
Improving Result Caching Performance

Tip 4: Do Not Load Caches Into Memory On Project Startup

Single requests for cached information is fast, even from disk

Cache access from disk

20 - 1000 ms

Loading thousands of caches into memory can delay project start up for hours

BI Application Startup Time

30 minutes to several hours

Caches on Disk

100,000 caches

Cache on Disk
Improving Result Caching Performance
Tip 5: Only Invalidate Caches Affected by Data Load

All caches are invalidated after a data load which decreases performance until caches are regenerated. Only invalidate caches where the data has changed, leaving many caches in place to enhance performance.

1. Database Load Completes
2. Only affected caches are invalidated
3. New Caches begin generating
4. Users can continue to access unaffected caches
Maximize MicroStrategy Speed and Throughput with High Performance Tuning

Agenda

1. Introduction to High Performance BI
2. Techniques that Optimize Caching Performance
3. Techniques for Designing In-Memory Cubes
4. Conclusion / Q&A
Techniques for Designing In-Memory Cubes

40%-60% of Reports are Candidates for In-Memory Performance Acceleration

Distribution of Report Speed

- Ad-hoc Analysis: One Cube for Each Subject Area
- Multiple Overlapping Reports: One Cube Covers Multiple Reports
- Highly-used Reports: One Cube for Each Report

Frequency of Usage:
- Seldom
- Occasional
- Often
- Frequent
- Very Frequent

Avg. User Wait Time:
- Short Wait
- Long Wait

In-Memory Candidates

Good Performance

Caching
Techniques for Designing In-Memory Cubes

Tip 1: Create One Cube For Each Highly Prompted Reports

Each Prompt Variation Generates a New Query

One Highly Prompted Report

40 sec Wait Time

Many Highly Similar Queries

Heavy Database Workload

One Cube Contains the Data for Many Prompts

One Highly Prompted Report

2 sec Wait Time

Create One Cube Containing Data For all Prompt Answers

One-time Database Query
Techniques for Designing In-Memory Cubes

Tip 2: Create Cubes for Reports with High Overlap in Attributes & Metrics

Many Reports with High Overlap in Content

Users Create Many Variations of an Original Report Design

Many Highly Similar Queries

40 sec Wait Time

Heavy Database Workload

One Cube Contains Common Attributes and Metrics

Create One Cube Containing Common Attributes & Metrics

One-time Database Query

2 sec Wait Time
Techniques for Designing In-Memory Cubes – Tip 2 (continued)

Use Cube Advisor to Identify In-Memory Cubes for Overlapping Reports

35% reduced

13% covered

13% of reports covered by the selected cubes. From total 783 reports, 294 (39%) reports are covered by one cube. 0(0%) reports are covered by more than one cubes.
Techniques for Designing In-Memory Cubes
Tip 3: Create Subject Area Cubes for Ad-hoc Drilling Analyses

Ad-hoc Drilling within a Subject Area

Users Drill within a Subject Area

40 sec Wait Time

Heavy Database Workload

Many Ad-hoc Queries of Similar Data

One Cube Contains Data for a Subject Area of Analysis

Users Drill within a Subject Area

2 sec Wait Time

Create One Cube For Each Subject Area of Ad-hoc Analysis

One-time Database Query
Techniques for Deploying In-Memory Cubes

Decisions about Cube Size Affect System Scalability

MicroStrategy’s in-memory operation exhibits almost constant user wait time across the range of cube sizes ...

... However, user and traffic scalability diminish rapidly as cube size get larger
Techniques for Deploying In-Memory Cubes

Tip 4: Design Smaller Cubes (<10GB) to Increase Scalability

One Big Cube that Contains All Prompt Answers → Multiple Smaller Cubes Segmented by Prompts → One Big Cube that Covers Many Reports at Once → Multiple Smaller Cubes That Cover Smaller Subsets of Reports

#mstrworld
Techniques for Deploying In-Memory Cubes

Tip 5: Don’t Build More In-Memory Cubes Than You Have Memory For

In-Memory Cubes are persisted on disk and get loaded into memory when accessed.

Depending on its size, the loading of In-Memory Cubes from disk can add significant delay to the user wait time.

In memory is insufficient to hold all requested In-Memory Cubes, the least recently used In-Memory Cubes will be dropped from memory.
Techniques for Managing In-Memory Cubes

Tip 6: Refresh In-Memory Cubes Frequently To Minimize Data Latency

Traditionally, data latency is mainly dependent on the DB query time.
The data in In-Memory Cubes is as recent as the last refresh cycle, and the user receives optimal performance.
Scheduled refreshing of In-Memory Cubes ensures that the data a user receives is no older than the last cube refresh cycle while delivering the shortest wait time.
Maximize MicroStrategy Speed and Throughput with High Performance Tuning

Agenda

1. Introduction to High Performance BI
2. Caching Techniques that Optimize Performance
3. Techniques for Designing In-Memory Cubes
4. Conclusion / Q&A