Analytics on SparkSQL: How to leverage SparkSQL with MicroStrategy
Safe Harbor Notice

This presentation describes features that are under development by MicroStrategy. The objective of this presentation is to provide insight into MicroStrategy’s technology direction. The functionalities described herein may or may not be released as shown.

This presentation contains statements that may constitute “forward-looking statements” for purposes of the safe harbor provisions under the Private Securities Litigation Reform Act of 1995, including estimates of future technology releases. Forward-looking statements inherently involve risks and uncertainties that could cause actual results of MicroStrategy Incorporated and its subsidiaries (collectively, the “Company”) to differ materially from the forward-looking statements.

Factors that could contribute to such differences include: the Company’s ability to develop, market and deliver on a timely and cost-effective basis new or enhanced offerings that respond to technological change or new customer requirements; delays in the Company’s ability to develop or ship new products; the extent and timing of market acceptance of MicroStrategy’s new offerings; continued acceptance of the Company’s other products in the marketplace; competitive factors; general economic conditions; and other risks detailed in the Company’s registration statements and periodic reports filed with the Securities and Exchange Commission. By making these forward-looking statements, the Company undertakes no obligation to update these statements for revisions or changes after the date of this presentation.
Topics

• Spark Basics
• Spark SQL
• Integration with SparkSQL
• Tuning MicroStrategy with SparkSQL
• Advanced Workflows for Spark
• Q&A
In addition, Spark can be used interactively to query big datasets from the Scala interpreter. We believe that Spark is the first system that allows a general-purpose programming language to be used at interactive speeds for in-memory data mining on clusters.
Spark Architecture

http://spark.apache.org/
Resilient Distributed Data Structures

Collections distributed across nodes to support distributed job. Spark 2.X adds unified DataFrames and Datasets.
Unified Data Structures Unifies Features

Unified Apache Spark 2.0 API

- Untyped API
  - DataFrame = Dataset[Row]
  - Alias

- Typed API
  - Dataset[T]

Key Points for Spark

• Spark doesn’t store information – it uses Hadoop, RDBMS, file system and NoSQL storage for input and output
• Internally, Spark uses in-memory stores to pass data from one processing step to another
• In-memory data is not available after a Spark application completes.
Dual Role of SparkSQL

- Internally SQL is used to simplify set processing
- SQL Enabled data structures evolved from RDDSchema, to DataFrames, to Datasets which adds names, columnar formats, type safety, and optimization
- SparkSQL thru ODBC/JDBC is yet another way to get tables persisted in Hive and uses the Spark Thrift Server
- A Spark script or program needs to persist into Hive to make data available thru SparkSQL ODBC
- There is also Hive on Spark which uses Hive SQL and executes over Spark
Spark Thrift Server was adapted from HiveServer2

Spark Thrift Server
Spark Thrift Server

```
[cloudbreak@ip-10-0-0-248 ~] $ sudo -u /usr/hdp/2.5.0.1-210/spark/sbin/start-thriftserver.sh --master yarn-client --executor-memory 1512m --num-executors 2 --hiveconf hive.server2.thrift.port=10001
```
Hive on Spark
Integration with SparkSQL
There are numerous ways for MicroStrategy to interact with SparkSQL

- Ad-hoc Schema
- Live Connect
- In-Memory Cube
- Modeled Schema
- JDBC Driver and
- ODBC Driver
Integration with Spark SQL ODBC/JDBC

Sources: unstructured, semi-structured, structured

Storage: HDFS, RDBMS, Hive, HBase, Cassandra, MongoDB, SOLR, Elastic, Other NoSQL

Leveraging Spark

- MicroStrategy generates and sends analytical SQL to Spark
- Spark SQL thru ODBC can leverage a range of Spark and Hadoop functionality
- With UDFs, all Spark functionality can be accessed
- Results are collected and returned from persisted Hive storage
Checklist for Spark SQL ODBC/JDBC

- Host name
- Port
- Security Information
- Setup DSN
- Test DSN with DB Query Tool
MicroStrategy generates SparkSQL-specific SQL

- MicroStrategy integrates with SparkSQL’s list of database functions and SQL functionalities to improve analytical performance

- 14 of 14 Basic functions are supported
- 24 of 26 OLAP functions are supported
- 18 of 23 Date-time functions are supported
- 30 of 34 Mathematical functions are supported
- 4 of 12 Statistical functions are supported
- 13 of 13 String functions are supported
MicroStrategy Generates Multi-Pass SQL Queries For Analytical Richness

• By default, MicroStrategy uses derived table syntax to hold intermediate result sets

• Derived table syntax advantage
  • Minimize overhead by combining additional SQL passes in Multi-pass SQL to query blocks in the from clause
  • Use-cases resulting in temporary table creation despite preference for derived table syntax
    • Analytical Query Engine
    • MicroStrategy Partitioning feature

```sql
select distinct a22.STORE_NBR STORE_NBR,
a22.STORE_DESC STORE_DESC,
a21.TOT_SLS_STR TOT_SLS_STR,
(a23. REG_SLS_REG + a21. PRO_STR) TOT_SLS
from
(select
a21.STORE_NBR STORE_NBR,
(sum(a21.REG_SLS_DLR) + sum(a21.PML_SLS)) TOT_SLS_STR,
sum(a21.PML_SLS_DLR) PRO_STR
from STORE_DIVISION a21
where a21.STORE_NBR = 1
group by a21.STORE_NBR) as TEMP1
join LOOKUP_STORE a22
on (TEMP1.STORE_NBR = a22.STORE_NBR)
join
(select
a22.REGION_NBR REGION_NBR,
sum(a21.REG_SLS_DLR) REG_SLS_REG
from STORE_DIVISION a21
join LOOKUP_STORE a22
on (a21.STORE_NBR = a22.STORE_NBR)
where a22.STORE_NBR = 1
group by a22.REGION_NBR) as TEMP2
on (a22.REGION_NBR = TEMP2.REGION_NBR)
```
MicroStrategy Avoids Unnecessary Workload on SparkSQL

Enabling SQL Global Optimization reduces the number of SQL passes improving query performance

Before Global Optimization
Redundant SQL Pass

After Global Optimization Level 1
Redundant SQL Pass automatically removed

Before Global Optimization
Metric definitions force different SQL passes

After Global Optimization Level 2
SQL Engine automatically combines different SQL passes into a single SQL pass
MicroStrategy Pushes Smart SQL to SparkSQL

SQL Global Optimization is enabled by default for SparkSQL
Tuning MicroStrategy for SparkSQL
Spark Table Caching: Cache me if you can

https://community.microstrategy.com/s/article/Pinning-Spark-Tables-Capacity-Planning-in-Ambari
Joining in Spark are one of the most expensive operations.

- **Shuffle join**: most expensive join, requires repartition tables to the same partition key and then join.

<table>
<thead>
<tr>
<th>Name</th>
<th>dept_num</th>
<th>Compensation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peter</td>
<td>100</td>
<td>21</td>
</tr>
<tr>
<td>Nancy</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Robert</td>
<td>300</td>
<td>22</td>
</tr>
<tr>
<td>Martha</td>
<td>400</td>
<td>23</td>
</tr>
<tr>
<td>John</td>
<td>100</td>
<td>22</td>
</tr>
<tr>
<td>Jennifer</td>
<td>200</td>
<td>23</td>
</tr>
<tr>
<td>Charlotte</td>
<td>300</td>
<td>22</td>
</tr>
</tbody>
</table>

### Example Code

Create table `Employee` as

```sql
CREATE TABLE Employee
AS
SELECT name, dept_num, compensation
FROM LU_Employee;
```

Create table `Department` as

```sql
CREATE TABLE Department
AS
SELECT dept_num, dept_name
FROM LU_Department
DISTRIBUTE BY dept_num;
```
Data Modeling: Joins

Joins in Spark are one of the most expensive operations.

Select name, dept_num, dept_name, compensation
from Employee join Department on Employee.dept_num = Department.dept_num;

Shuffle Operation: redistribute over same partition key: dept_num
Data Modeling: Joins

Joins in Spark are one of the most expensive operations

- Shuffle join: most expensive join, requires repartition tables to same partition key and then join:

Create table Employee as
Select name, dept_num, compensation
from LU_Employee
distribute by dept_num;

Create table Department as
Select dept_num, dept_name
from LU_Department
distribute by dept_num;

Shuffle operation Not required!
Data Modeling: Joins

Joins in Spark are one of the most expensive operations

• Good idea to set partition on join columns (create table as select ... distribute by <column1, column2>)

• De-normalize the data model to avoid 1:M joins:
  • Move Description attribute forms (like dept_name) into the fact table.
  • Keep the lookup tables around for quick prompt resolution, and VLDB setting “Attribute Form selection Option for Intermediate Pass” as “Select ID and other forms if they are on the template and available in existing join tree” to avoid joining with Lookup table.

• Use Broadcast join:
  • When Spark SQL plans joins, it will identify whether the tables to join are smaller than the limit spark.sql.autoBroadcastJoinThreshold, and if so, transfer the tables to all the executors. This allows for faster joins. Make sure the value is not set to larger than the execution memory each executor has.
  • spark.sql.autoBroadcastJoinThreshold = 10485760 (bytes)

Ref: https://community.microstrategy.com/s/article/Best-Practices-for-Spark-Thrift-Server-on-YARN
Advanced Workflows for Spark
Other Ways to Leverage Spark

- Spark program that writes to Hive for query with SQL
- Spark program that writes to HDFS for query with Hadoop Gateway
- Spark with Transaction Services
  - Button to invoke Spark program and generate results using web services access using Livy Server
- Spark routine to build a cube
  - Spark routine stages data for Hadoop Gateway load and triggers an event
  - Hadoop Gateway cube update triggered by event
  - Option to use Spark Streaming for continuous updates
- Trigger an event in a Spark routine to trigger MicroStrategy alerts
- Build user defined function, user defined aggregation function, user defined table functions with Spark and call in Spark SQL
- SparkR/SparklyR Integration of R to Spark
- Generate a data mining model with Spark and invoke as part of a dataset
Integration of R with Spark using SparkR or SparklyR

- R integrates to Spark with SparklyR
- R integrates to Spark with SparkR
- MicroStrategy can execute RScript that includes packages using a metric
- You can separate model generation from model scoring
- These scripts can be used for algorithm training as well as to generate statistics for exploratory data analysis

Sources: unstructured, semi-structured, structured

Storage: HDFS, RDBMS, Hive, HBase, Cassandra, MongoDB, SOLR, Elastic, Other NoSQL

Leveraging Spark
Integration of R with Spark using SparkR or SparklyR
Leveraging alerts and transactions for actionable insights

A streaming application can trigger alerts and transactions can invoke processes or control devices.
Moving from one Hadoop SQL Engine to Another
Summary

• Spark offers a broad range of data processing capabilities with a unified approach for data structuring and processing in a large-scale environment
• Spark does not natively store data, but ingests data from multiple data stores and can store data to them as well
• SparkSQL JDBC/ODBC access is a data engine built over Spark’s APIs that accesses data stored in Hive
• Optimize SparkSQL queries through table caching and join avoidance.
• Through SparkSQL you can leverage native features of Spark that are executed in the cluster
• Other Spark touch points allow additional access to Spark features including advanced algorithms, streaming, integrated SQL and procedural logic
References

- https://beta.rstudioconnect.com/content/1705/taxiDemo.nb.html
- http://spark.rstudio.com/examples.html
- https://community.microstrategy.com/s/article/Pinning-Spark-Tables-Capacity-Planning-in-Ambari
References

• **High Performance Spark**  
  **By:** Holden Karau; Rachel Warren  
  **Publisher:** O'Reilly Media, Inc.  
  **Print ISBN:** 978-1-4919-4320-5  
  **Print ISBN-13:** 978-1-4919-4320-5

• **Apache Spark for Data Science Cookbook**  
  **By:** Padma Priya Chitturi  
  **Publisher:** Packt Publishing  
  **Print ISBN-13:** 978-1-78588-010-0

• **Sams Teach Yourself Apache Spark™ in 24 Hours**  
  **By:** Jeffrey Aven  
  **Publisher:** Sams  
  **Print ISBN-13:** 978-0-672-33851-9  
  **Print ISBN-10:** 0-672-33851-3

• **Big Data Analytics**  
  **By:** Venkat Ankam  
  **Publisher:** Packt Publishing  
  **Print ISBN-13:** 978-1-78588-469-6