
ORACLE®

Master Data Management Using Oracle Table Access for Hadoop and Spark

Integrate Master Data in Big Data Analytics

Kuassi Mensah
Director of Product Management
Oracle, Server Technologies
October 28, 2015

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Program Agenda

1. Big Data Analytics & Requirements
2. Direct Access to Master Data
3. Oracle Table Access for Hadoop & Spark
4. Key Features and Benefits

Big Data Analytics

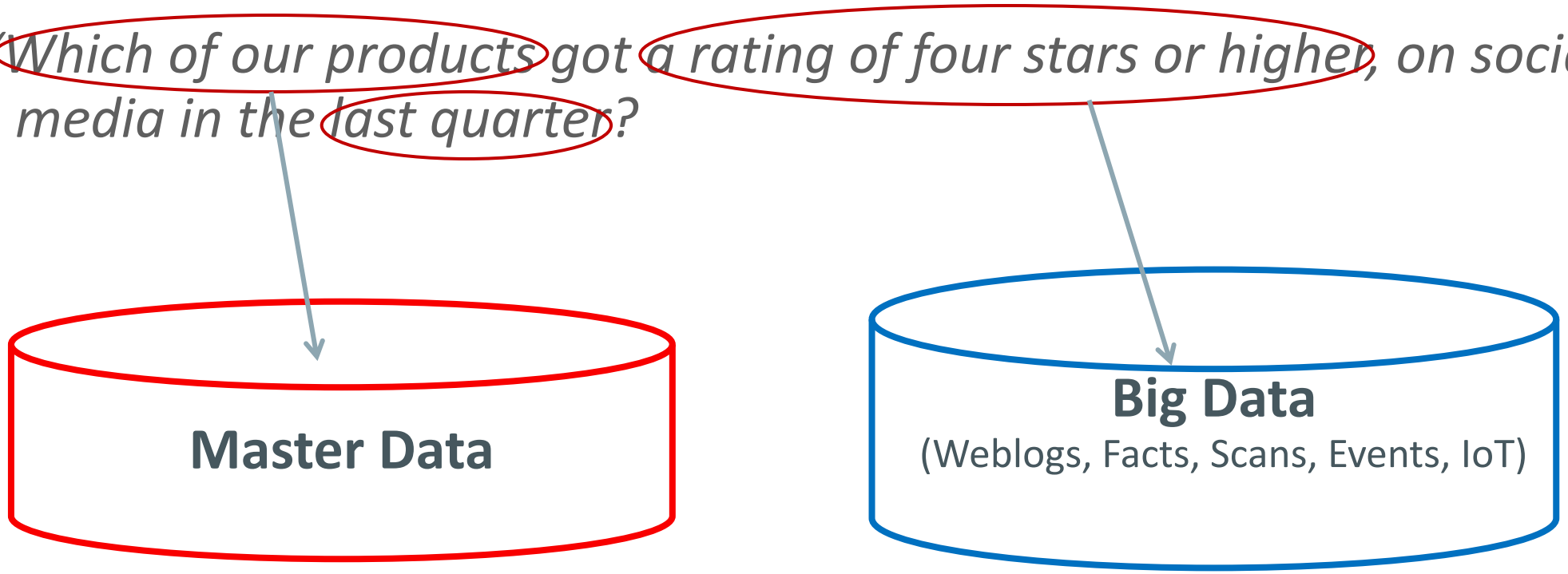
- **Goal**: furnish actionable information to help business decisions making.
- Example

  "Which of our products got a rating of four stars or higher, on social media in the last quarter?

Big Data Analytics and Requirements

- **Goal**: furnish actionable information to help business decisions making.
- Example

"Which of our products got a rating of four stars or higher, on social media in the last quarter?



Program Agenda

1. Big Data Analytics & Requirements
2. Direct Access to Master Data
3. Oracle Table Access for Hadoop & Spark
4. Key Features and Benefits

Access to Master Data: Two Approaches

- ETL Copy: Oracle -> Hadoop
  - Preplanned/scheduled
    - What to copy and when?
    - Always behind
  - Copy is protected using Hadoop file-level security

Oracle CopyToBDA 2.0

- Direct Access from SQL engines
  - Ad-hoc queries, always current
  - Hive SQL, Spark SQL, Impala*, other SQL engines
- Direct Access from APIs
  - All Hadoop & Spark APIs

Oracle Table Access for Hadoop

Direct Access using Query

Hive query for joining tables from Big Data and Oracle

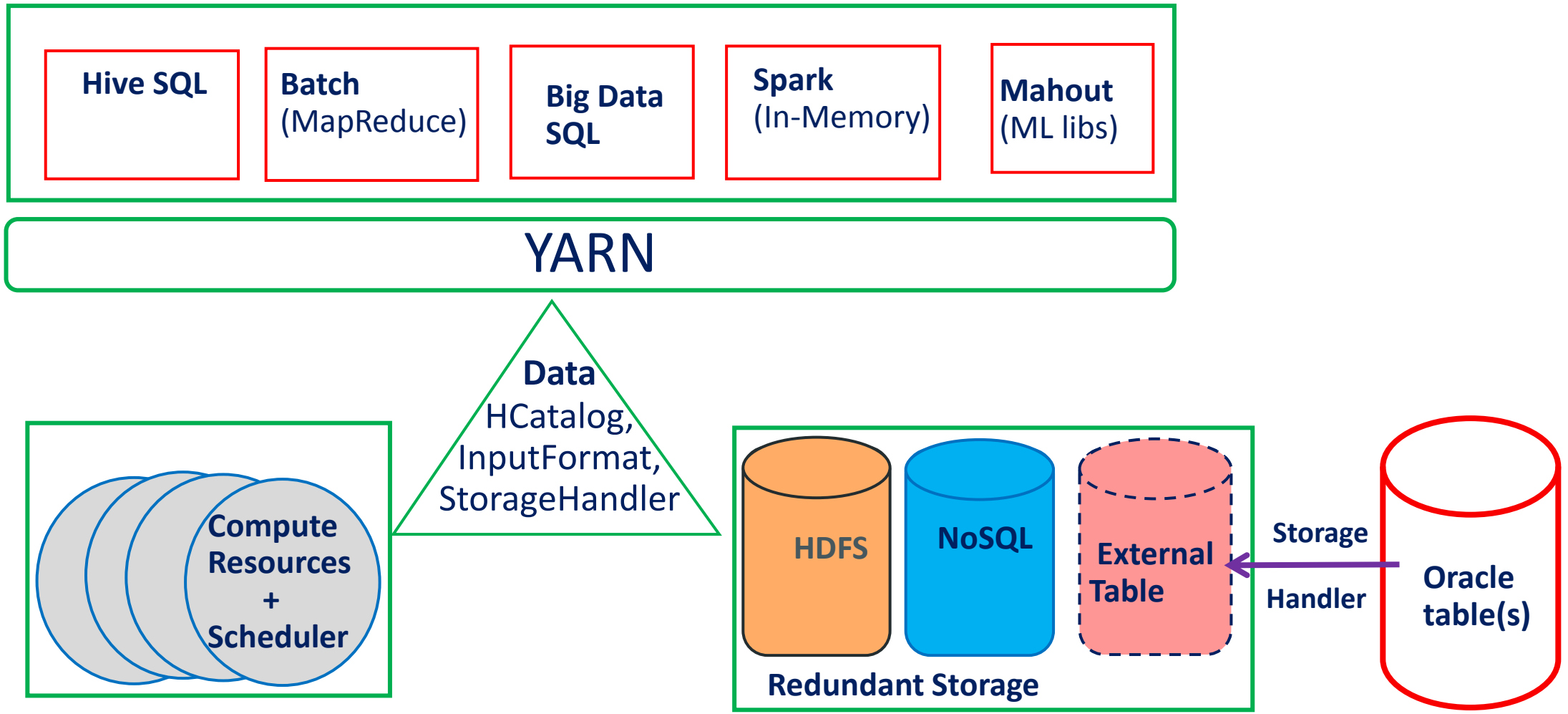
```
SELECT HadoopT.First_Name, HadoopT.Last_Name, OracleT.bonus
  FROM HadoopT join OracleT on  (HadoopT.Emp_ID=OracleT.Emp_ID)
  WHERE  salary > 70000 and bonus > 7000;
```

Demos

Program Agenda

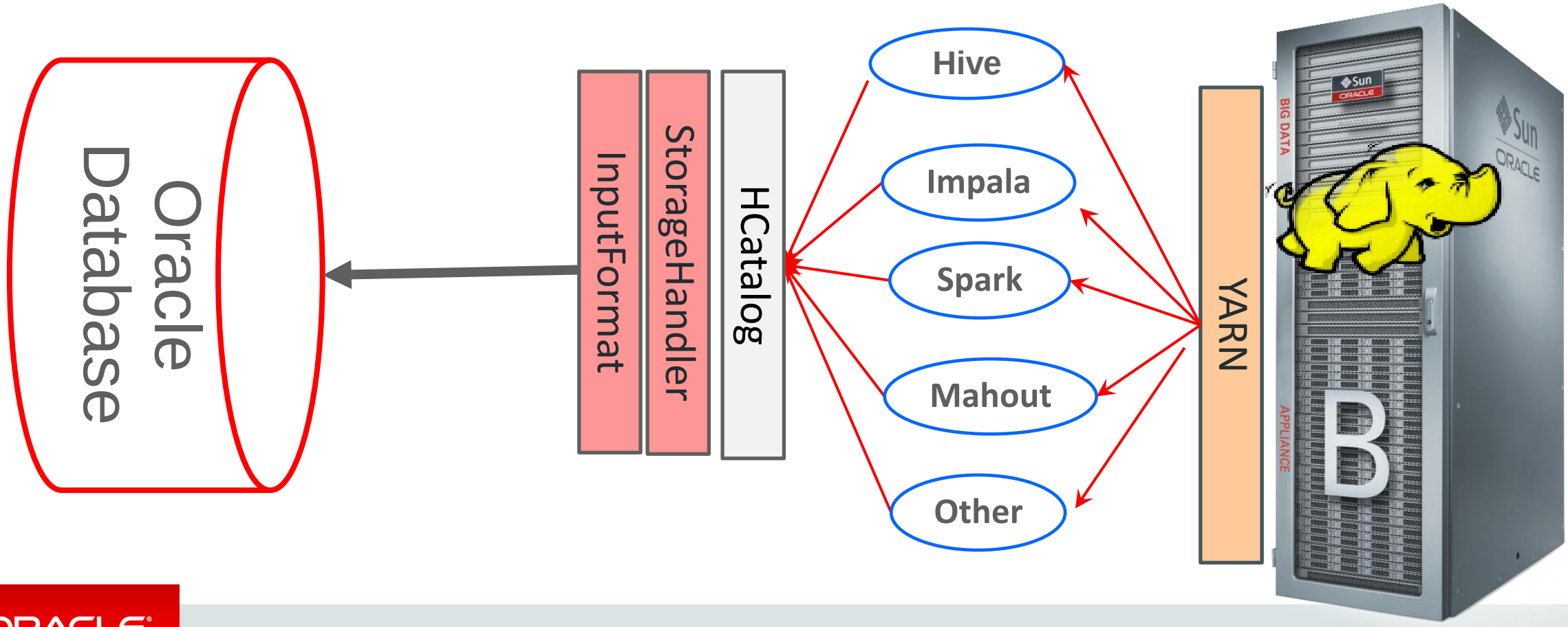
1. Big Data Analytics & Requirements
2. Direct Access to Master Data
3. Oracle Table Access for Hadoop & Spark
4. Key Features and Benefits

Hadoop 2.0 Architecture



Oracle Table Access for Hadoop & Spark (OTA4H)

Direct, parallel, fast secure and consistent access to master data



Oracle Table as Hive External Table

- DDL

```
CREATE EXTERNAL TABLE hive_table1 (
  id DECIMAL,   name STRING,  salary DECIMAL)
  STORED BY 'OracleStorageHandler'
  TBLPROPERTIES (Input.table.name' = 'Oracle_table1', ...);
```

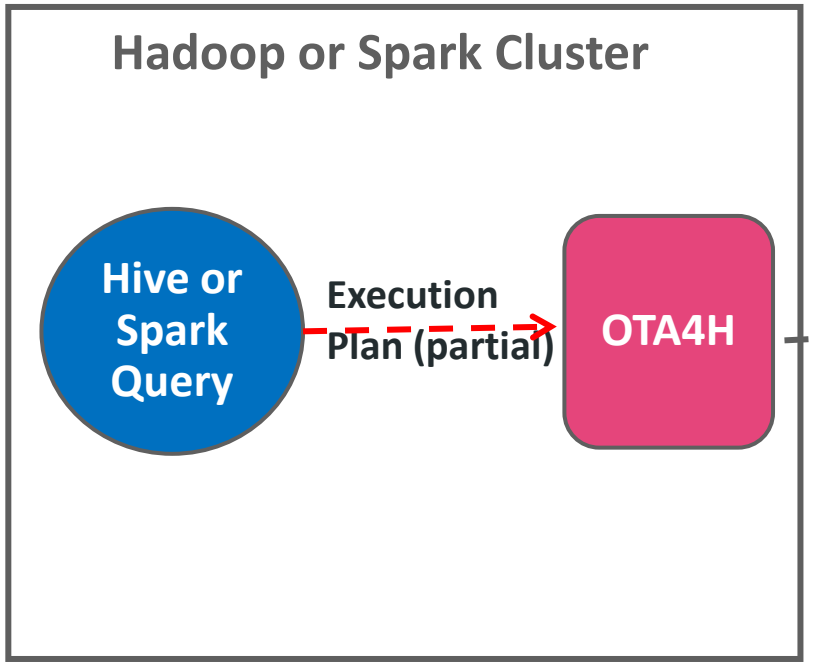
- HiveQL

```
hive  -e "${QUERY}"
```

- Spark SQL

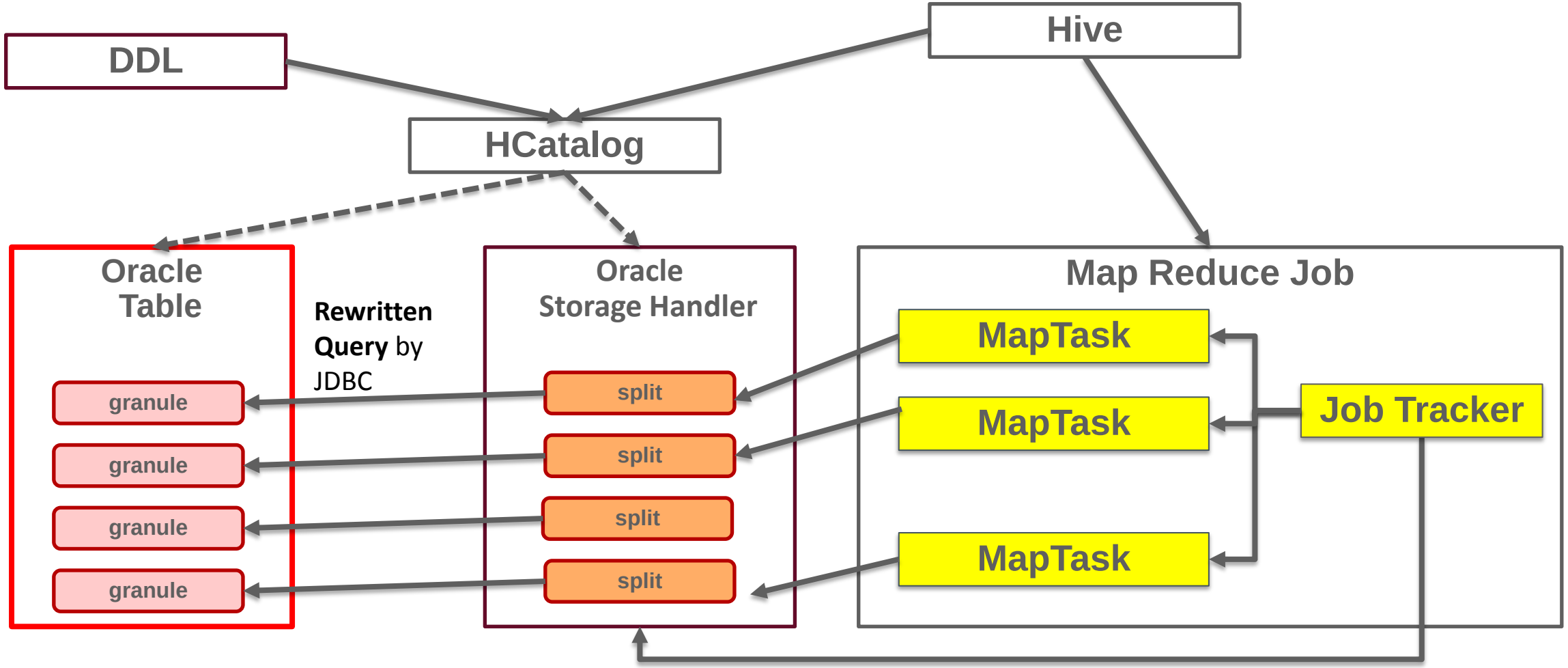
```
/usr/lib/spark/bin/spark-sql -e "${QUERY}"
```

OTA4H Steps



1. Gets a secure connection to DB
2. Generate database Splits (DLDL) with SCN
3. Rewrites HiveQL or Spark SQL into Oracle SQL for each split
4. Each split is processed by a Hadoop/Spark task
5. Matching rows returned to Hadoop/Spark Query coordinator

Putting Everything Together



Program Agenda

1. Big Data Analytics & Requirements
2. Direct Access to Master Data
3. Oracle Table Access for Hadoop & Spark
4. Key Features and Benefits

OTA4H Key Features and Benefits

- Performance and Scalability
- Resource Management and Consistency
- Security
- High Availability
- Data Types
- Future Enhancements (Potential)

Performance and Scalability

Fully exploit Hadoop or Spark clusters and Oracle database servers

- Splitter Patterns
- Optimized JDBC Driver
- Connection Caching
- Integration with Database Resident Connection Pool (DRCP)
- Predicate Pushdown
- Partition Pruning

Parallel Access to Oracle Table: Splitter Patterns

- SINGLE_SPLITTER
- ROW_SPLITTER
  number of rows set in *oracle.hcat.osh.rowsPerSplit*
- BLOCK_SPLITTER
  max # of splits directed by *oracle.hcat.osh.maxStorageBasedSplits*
- PARTITION_SPLITTER

Parallel Access to Partitioned Oracle Table

```
CREATE EXTERNAL TABLE employees (
  EMPLOYEE_ID INT, FIRST_NAME  STRING, LAST_NAME   STRING,
  SALARY      DOUBLE, HIRE_DATE   TIMESTAMP,
  JOB_ID      STRING)
  STORED BY
  'oracle.hcat.osh.storagehandler.OracleStorageHandler ‘
  TBLPROPERTIES (
    'mapreduce.jdbc.url' =
  'jdbc:oracle:thin:@localhost:1521:orcl',
    'mapreduce.jdbc.username' = ‘foobar',
    'mapreduce.jdbc.password' = ‘ dontdothis',
    'mapreduce.jdbc.input.table.name' = 'EMPLOYEES',
    'oracle.hcat.osh.splitterKind' = ‘PARTITION_SPLITTER'
  );
```

Resource Management and Consistency

- MaxSplit
- DRCP

  `maxconnections`
- Hadoop

  `mapred.tasktracker.map.tasks.maximum  in conf/mapred-site.xml`
- Spark

  `spark.dynamicAllocation.enabled`
- System Commit Number (SCN)

Security

OTA4H Supports all Oracle JDBC authentications methods:

- Simple and Strong Authentication
  - Username/password
  - Wallet
  - Kerberos
- Encryption and Integrity
- JVM System Properties
- Hive/Hadoop/Spark environment variables

OTA4H Summary

- Support for Hadoop & Spark query engines: Hive SQL, Spark-SQL, Impala*
- Support for Hadoop programming models: Pig, MapReduce, Pig, etc
- Secure and reliable authentication: Kerberos authentication, SSL, Oracle Wallet
- Efficient translation of HQL to Oracle SQL
- Scalability: splits based on DB meta-data
- Column Projection Pushdown
- Predicate Pushdown
- Partition Pruning
- Connection caching: pool in each Hadoop engine JVM
- Consistent Read: Execute entire query as of a single SCN
- Looking into SchemaRDD and DataFrames

Demos


ORACLE®