Experiences Migrating Hive Workload to SparkSQL

Zhan Zhang, Jie Xiong

zhanzhang@fb.com, jiexiong@fb.com

Overview

- Motivation
- Syntax & Semantics Gap Analysis
- Offline & Online Shadowing
- Performance Optimization
- Challenges and Future Work

Background

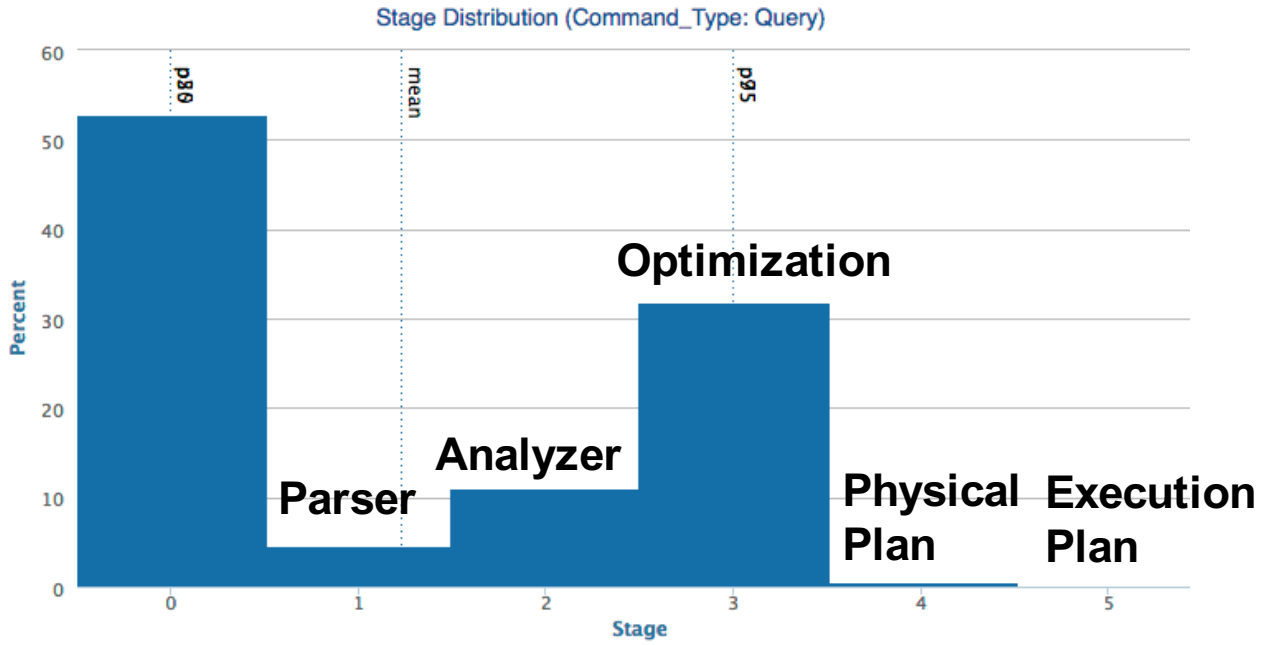
- Make batch compute in Facebook more efficient.
- Bridge the gap between Spark and Hive so Spark can handle production workload in Facebook.
- Unified User Interface for SparkSQL and HQL.

Preparation - Syntax Analysis

- Syntax Gap Analysis
  - Use our daily hive query log to select query candidates.
  - A group of Spark Drivers each runs a subset of the candidates for daily syntax analysis.
    - Parsing, analyzing, optimization, physical plan generation, and executed plan

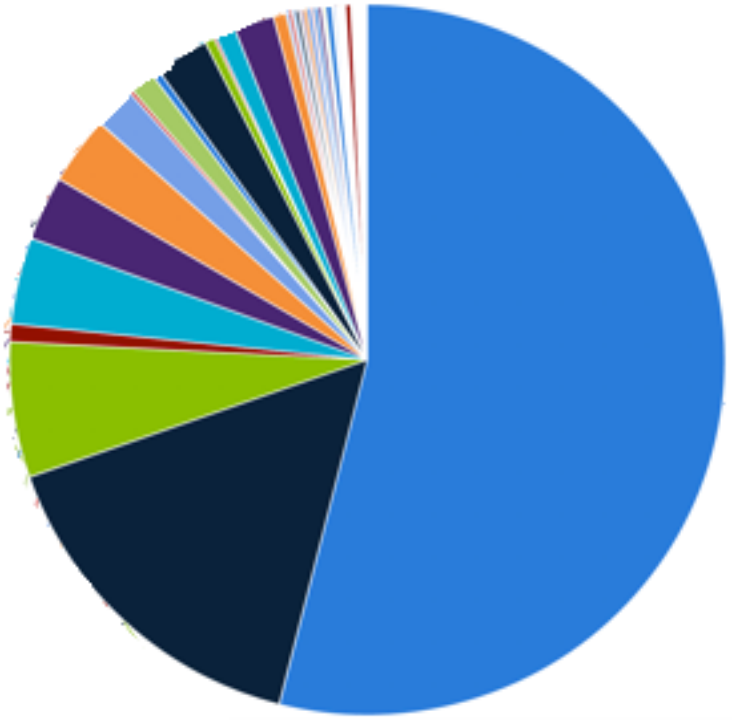
Syntax Analysis – Error Distribution on Each Stage

- 0: Success
- 1: Parser
- 2. Analyzer
- 3. Optimization
- 4. Physical Plan
- 5. Execution Plan



Syntax Analysis – Hive CPU Usage Distribution by Error Category

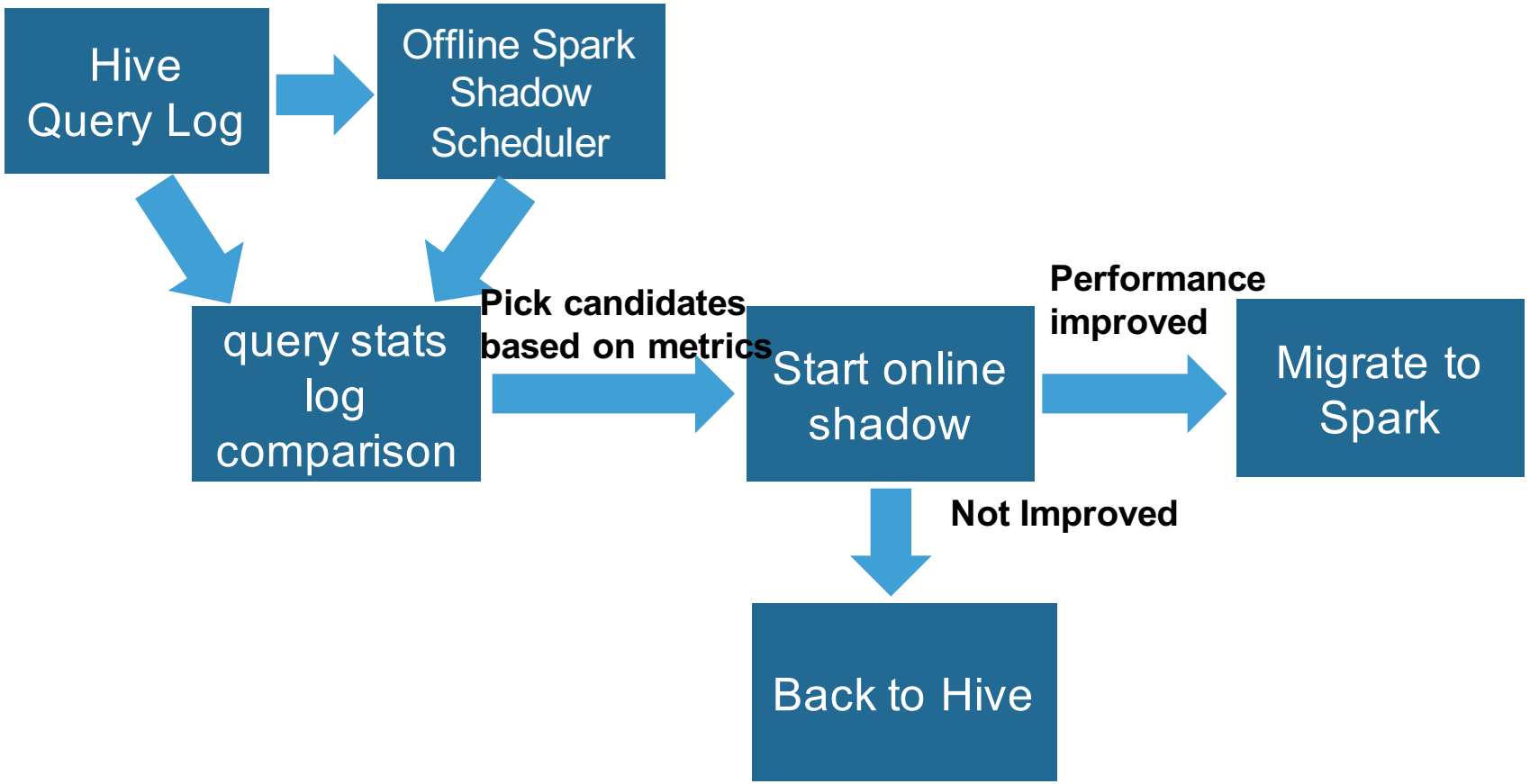
- The computation weight by error category.
  - More than 50% without errors
  - A small number of syntax errors take a big percentage.



Preparation - Semantic Validation

- Avoid affecting production pipelines
  - Rewrite the parsed plan by appending the output table with suffix *_spark_shadow*
- Avoid obsolete data
  - Run the same query on the same source data one day after the hive query finishes.
- Verify the correctness
  - Hash validation: sometimes even heavier than the query itself.
  - Count validation: fast way to filter out obvious errors.

Migration Steps



Migration Metrics

- Correctness
- Wall time
- Reserved CPU time
- Stability

Offline Shadow Process

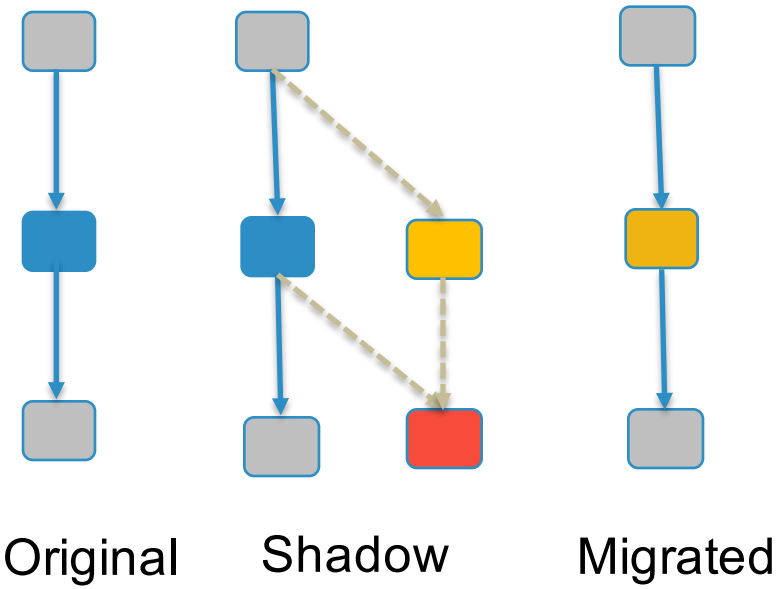
- Pickup the namespace
- Setup pool for offline shadowing
- Select the pipelines above some compute cost threshold
- Set up a Spark application to continuously run the selected pipelines in shadow mode.
- Pick up the candidate for online shadowing using our metrics

Online Shadow Process

- Running in parallel with production pipelines in different pools
- No Change to HQL
  - rewrite the rule to make SparkSQL and HQL consistent
- No impact on production
  - Whether shadow job succeeded or failed, the downstream jobs are not affected

Online Shadow Operator

- Unified Interface for SparkSQL and HQL
  - Rewrite the query plan to bridge the gap
- Support Different Running mode
  - Hive
  - Shadow (Hive &Spark)
  - Spark

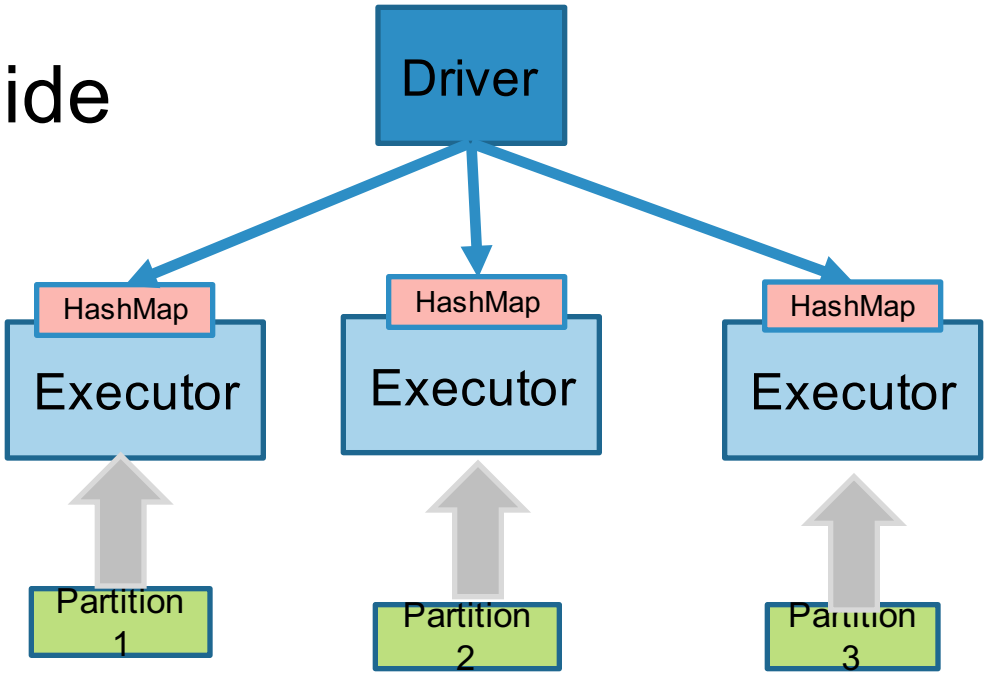


Performance Optimization

- Tradeoff between Shuffle Partitions and Disk Spill
  - The amount of Shuffled data varies per pipeline.
  - Automatic choose partition number based on input data size and hive historical data.
- Avoid unnecessary stage retries
  - Avoid retries when OOM happens.
- Avoid false task failure
  - Do not shut down executor when one task being killed.
- ReuseExchange to avoid redundant table scan.
  - Enable table scan reuse in Spark-SQL
- More accurate input data size estimation
  - A new rule to estimate input data size is added before join selection strategy.
  - Enable more advanced query optimization, e.g., BroadcastJoin/ShuffledHashJoin.

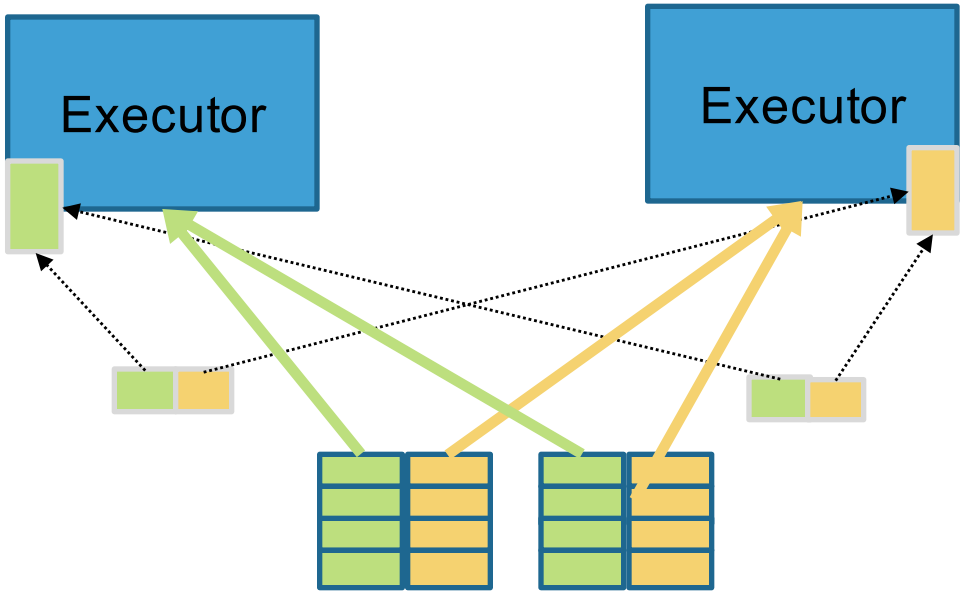
BroadcastJoin

- Driver collects and broadcast the smaller side to all tasks.
- Streaming the bigger side
- Overhead
  - best



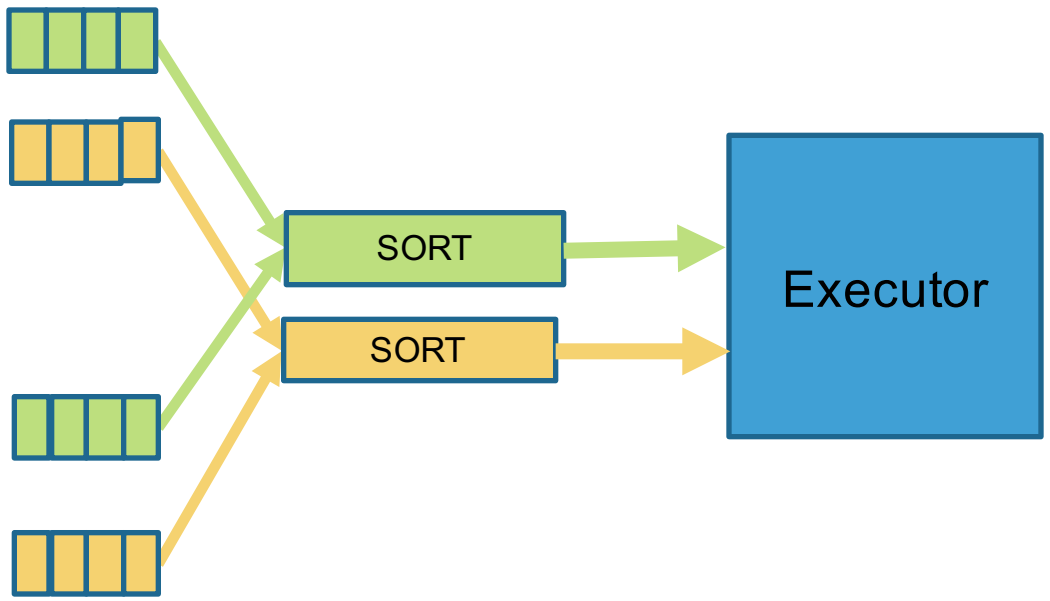
ShuffledHashJoin

- Build Hash on one side.
- Streaming the other side.
- Overhead
  - shuffle



SortMergeJoin

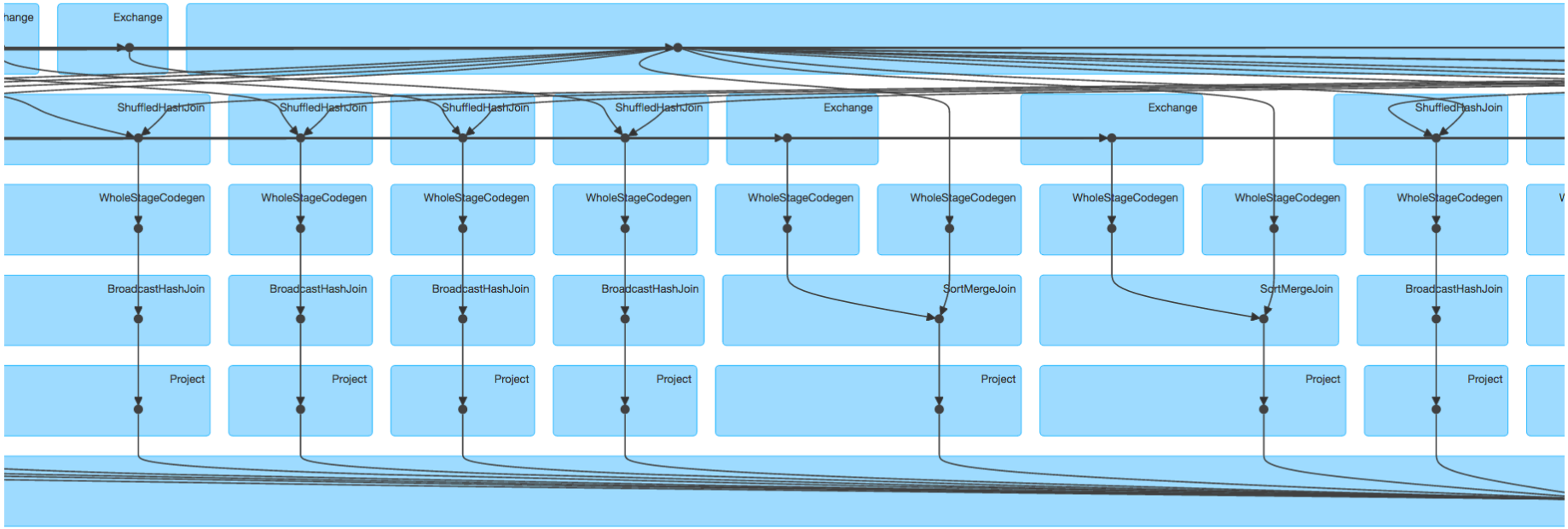
- Shuffle Rows with same keys to same tasks
- Sort both sides
- Join step by step
- Overhead
  - Shuffle/Sort/Spill



Tradeoff

- BroacastJoin
  - No Shuffle or Spill
  - OutOfMemory
- ShuffledHashJoin
  - Shuffle without Sort
  - OutOfMemory
- SortMergeJoin
  - Shuffle & Sort & Spill
  - Robust
- Fallback mechanism
  - Try ShuffledHashJoin.
  - Fallback to SortMergeJoin on failure.

Query Optimization



JIRA

- SPARK-20215: ReuseExchange is broken in SparkSQL
- SPARK-20006: Separate threshold for broadcast and shuffled hash join
- SPARK-19908: Direct buffer memory OOM should not cause stage retries.
- SPARK-19890: Make MetastoreRelation statistics estimation more accurate
- SPARK-19839: Fix memory leak in BytesToBytesMap
- SPARK-17637: Packed scheduling for Spark tasks across executors

Challenges and Future Work

- Non-deterministic UDF makes validation hard
- Performance degradation due to lack of HiveUDF WholeStageCodegen support
- Leverage Run-time/Historical data to get more accurate stats for advanced query optimization
- Maximize the utilization of HashAggregation and ShuffledHashJoin (with fallback mechanism)

Question?