Enhancements on Spark SQL Optimizer

Min Qiu
Huawei Technologies, Inc.
nuwabox.com

SPARK SUMMIT EAST
DATA SCIENCE AND ENGINEERING AT SCALE
FEBRUARY 16-18, 2016 NEW YORK CITY

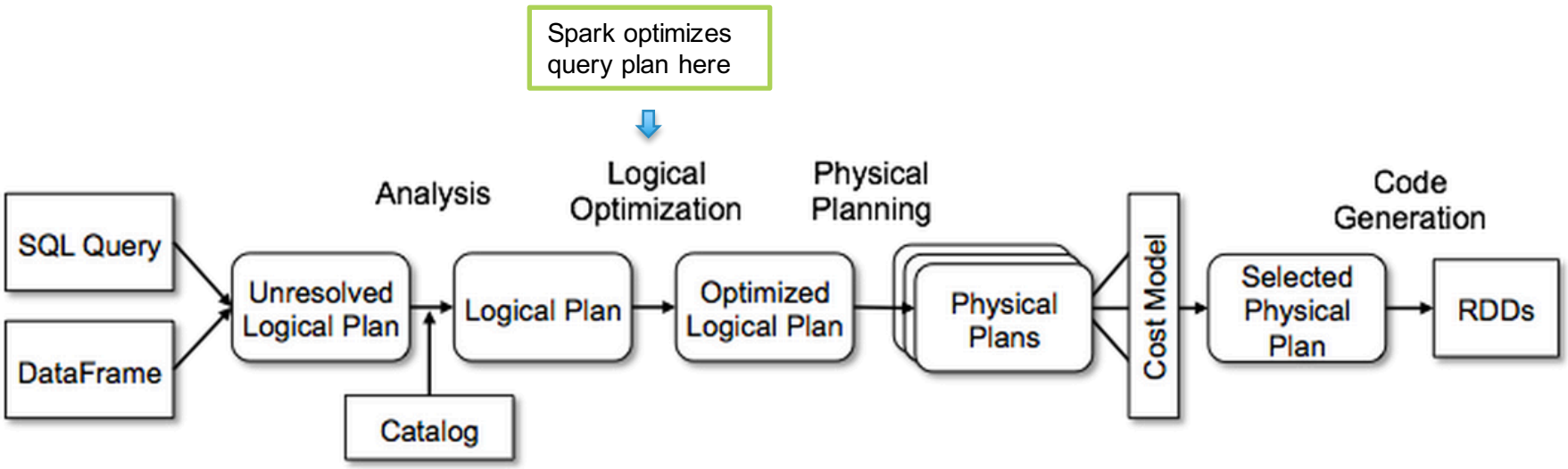
Agenda

- Review of Catalyst Architecture
- Rule-based Optimizations (RBO)
- Cost-based Optimizations (CBO)
- Future Work
- Q & A

Agenda

- Review of Catalyst Architecture
- Rule-based Optimizations
- Cost-based Optimizations
- Future Work
- Q & A

Catalyst Architecture



Reference: Deep Dive into Spark SQL's Catalyst Optimizer, a databricks engineering blog

Spark SQL Optimizer

- Optimizer is a RuleExecutor
- Individual optimization is defined as Rule
  - Equivalent transformations on analyzed plan
- Reduce the data volume and cost of computation

  i.e. Constant Folding, Filter/Project Pushdown…
- Well designed, and easy to extend
  - New rules can easily be plugged in to optimizer

- Very good already, but still incomplete
  - Need new optimization rules to cover more cases

Agenda

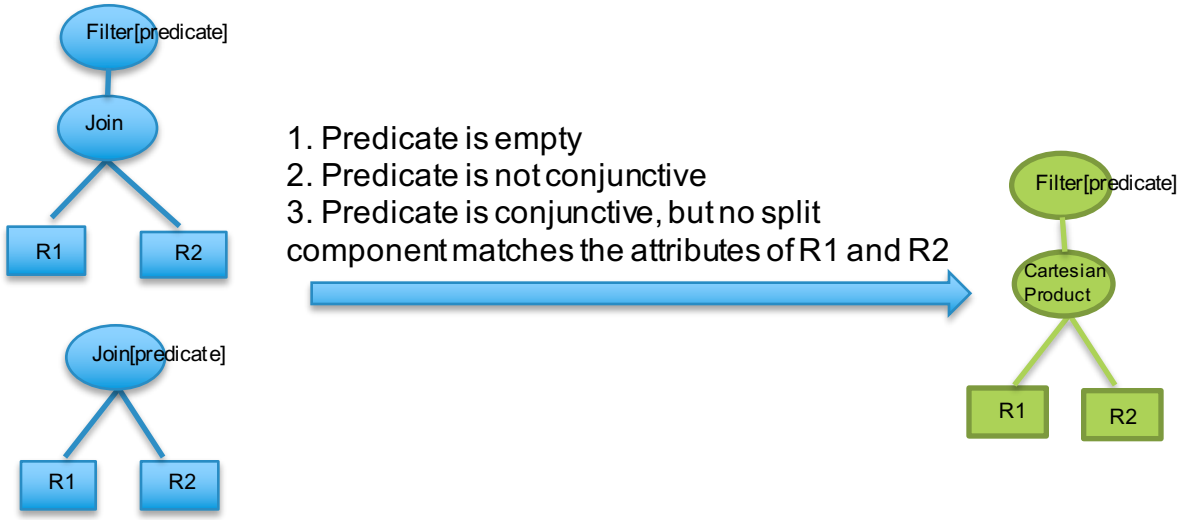
- Review of Catalyst optimizer
- Rule-based Optimizations (RBO)
- Cost-based Optimizations
- Future Work
- Q & A

Rule Based Optimizations

- Join Condition Push Down
  - Predicate Rewrite
  - Join Order Adjustment
- Data Volume Reduction
  - Column Pruning Enhancement

Join-condition Push Down

- In case of push down failure
  - No explicit equal-join condition recognized by EquiJoinSelection Strategy
  - Expensive Join operator could be planned (i.e. CartesianProduct)



Predicate Rewrite

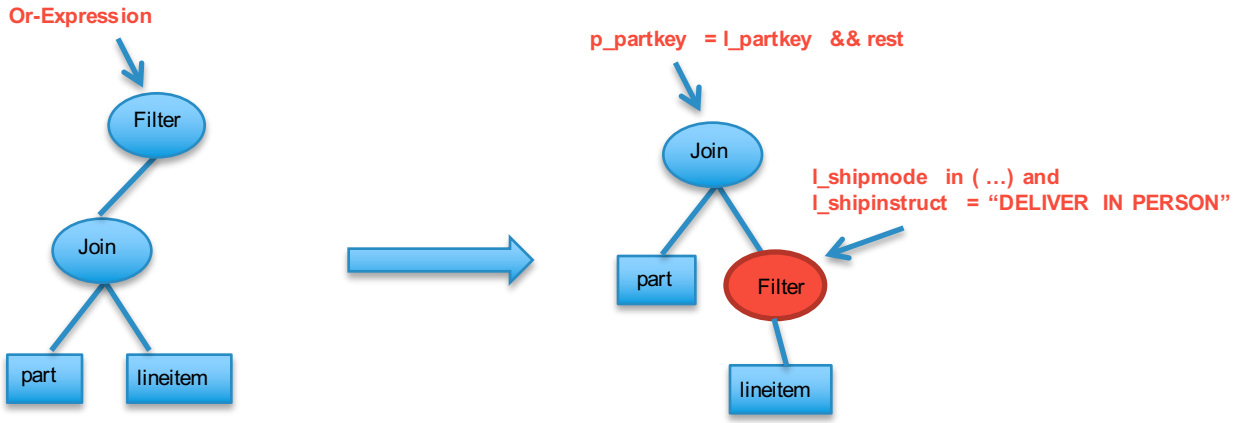
- Issue: SPARK-12032
  - Predicate is OR-expression
  - Join condition cannot be associated with a Join node

```
SELECT sum(l_extendedprice * (1 - l_discount)) AS revenue
FROM part join lineitem
WHERE (p_partkey = l_partkey
     AND p_brand = 'Brand#12'
     AND p_container IN ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
     AND l_quantity >= 1 AND l_quantity <= 1 + 10
     AND p_size BETWEEN 1 AND 5
     AND l_shipmode IN ('AIR', 'AIR REG')
     AND l_shipinstruct = 'DELIVER IN PERSON')
 OR (p_partkey = l_partkey
     AND p_brand = 'Brand#23'
     AND p_container IN ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
     AND l_quantity >= 10 AND l_quantity <= 10 + 10
     AND p_size BETWEEN 1 AND 10
     AND l_shipmode IN ('AIR', 'AIR REG') AND l_shipinstruct = 'DELIVER IN PERSON')
 OR (p_partkey = l_partkey
     AND p_brand = 'Brand#34'
     AND p_container IN ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
     AND l_quantity >= 20 AND l_quantity <= 20 + 10
     AND p_size BETWEEN 1 AND 15
     AND l_shipmode IN ('AIR', 'AIR REG') AND l_shipinstruct = 'DELIVER IN PERSON')
```

- Reason
  - Equality condition is hidden in DNF
  - The equality condition matches a join but cannot be pushed down to Join node

Predicate Rewrite 2

- Solution and Performance gain
  - Pull Request 10087
  - Extract the predicate that is common in each disjunctive component
  - OR expression is rewritten to AND expression
  - Query runs several times faster



Join Order Adjustment

- Issue: SPARK-12032
  - Join condition isn't pushed down to join relation due to mismatching join order
  - Expensive Join operator is planned

```
select
  orders.o_orderpriority as c0,orders.o_orderdate as c1,
  nation.n_name as c2, count(orders.o_orderkey) as m0
from orders, nation, customer
where orders.o_orderkey = customer.c_custkey and
    customer.c_nationkey = nation.n_nationkey and
    nation.n_name in ('ALGERIA', 'ARGENTINA', 'BRAZIL',
                      'ETHIOPIA', 'KENYA', 'MOROCCO', 'MOZAMBIQUE')
group by orders.o_orderpriority,orders.o_orderdate,  nation.n_name
```

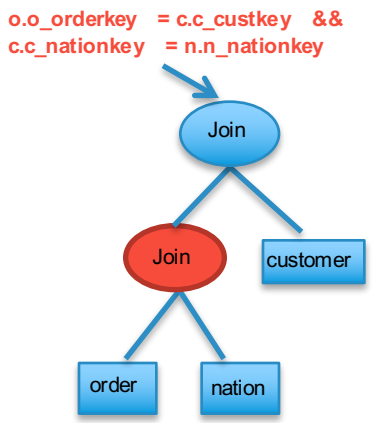
OOM if CartesianProduct

Query runs slowly

Join Order Adjustment 2

- Reason
  - Multi-table join order in query plan is the same as they appear in SQL statement
  - Join condition isn't associated with correct Join node

```
select
  orders.o_orderpriority as c0,orders.o_orderdate as c1,
  nation.n_name as c2, count(orders.o_orderkey) as m0
from orders, nation, customer
where orders.o_orderkey = customer.c_custkey and
    customer.c_nationkey = nation.n_nationkey and
    nation.n_name in ('ALGERIA', 'ARGENTINA', 'BRAZIL',
                      'ETHIOPIA', 'KENYA', 'MOROCCO', 'MOZAMBIQUE')
group by orders.o_orderpriority,orders.o_orderdate,  nation.n_name
```

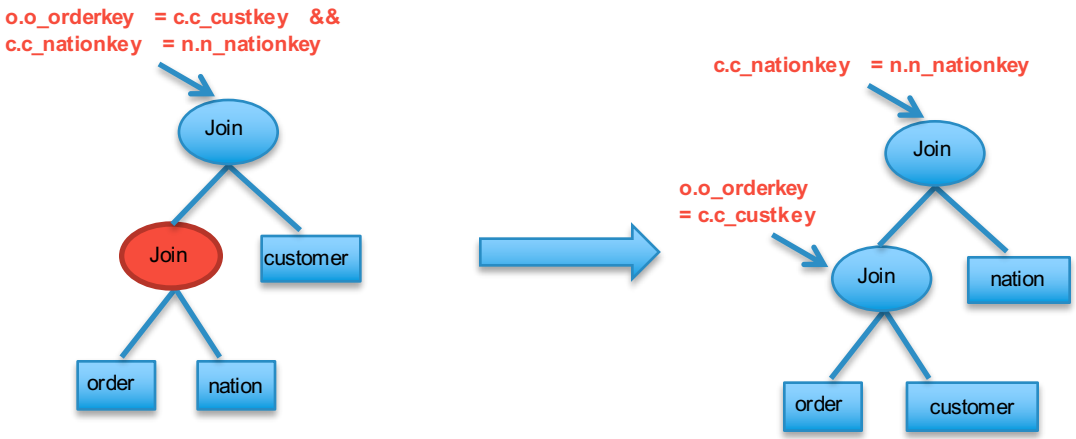


c_nationkey is from customer table
n_nationkey is from nation table

They don't match attributes from relation order and relation nation

Join Order Adjustment 3

- Solution and Performance gain
  - Pull Request #10258 (By me)
  - Pull Request #10073 (By other engineer)
  - Group join relations with matched join condition together
  - No OOM, Query runs several to tens of times faster



Column Pruning Enhancement

- Issue: SPARK-12114

  - ColumnPruning Rule fails in the following case
  - Unnecessary columns are read from disk and are involved in data shuffling

```
SELECT c_name, c_custkey, o_orderkey, o_orderdate, o_totalprice, sum(l_quantity)
FROM customer join orders join lineitem
    on c_custkey = o_custkey AND o_orderkey = l_orderkey
left outer join
    (
      SELECT l_orderkey tmp_orderkey
      FROM lineitem
      GROUP BY l_orderkey
      HAVING sum(l_quantity) > 300
    ) tmp
    on o_orderkey = tmp_orderkey
WHERE tmp_orderkey IS NOT NULL
GROUP BY c_name, c_custkey, o_orderkey, o_orderdate, o_totalprice
ORDER BY o_totalprice DESC, o_orderdate
```

Expected referenced columns:

c_name, c_custkey, o_orderkey, o_orderdate, o_totalprice, l_orderkey, l_quantity

In fact:

all columns from table customer, order and lineitem

a few hundred GB shuffling volume for 100GB DB

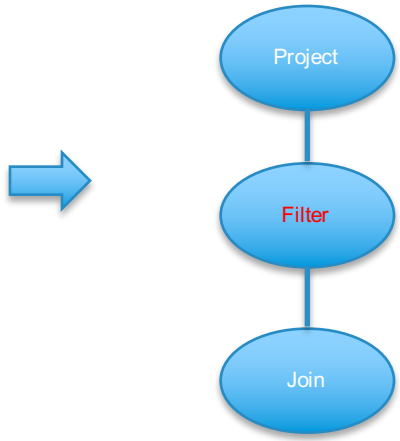
OOM, long running time

Column Pruning Enhancement 2

- Reason
  - The rule only considers the following patterns

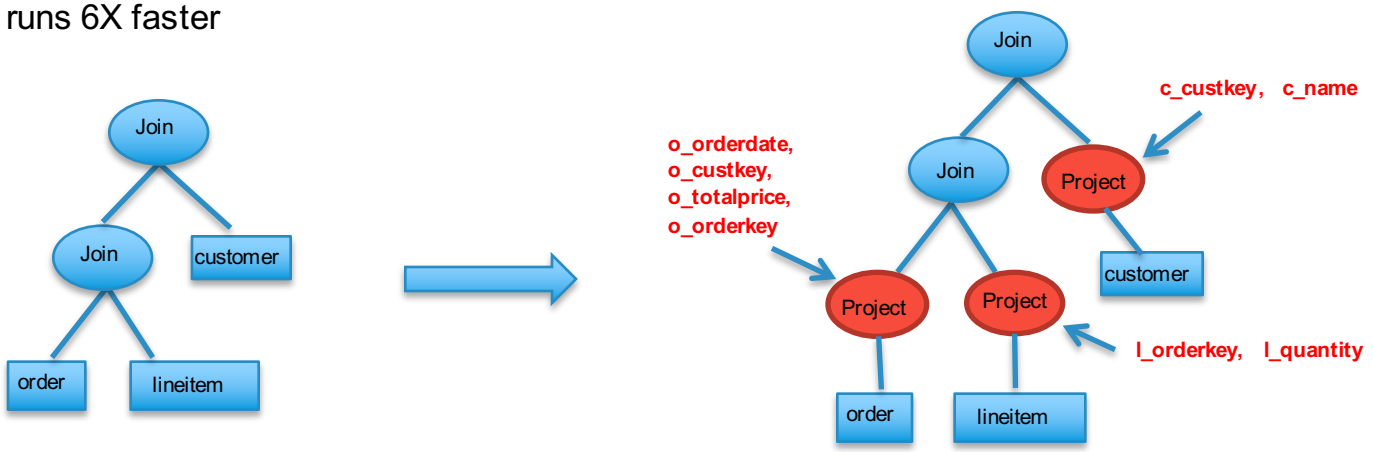
    Aggregate, Generate, Project <- Join, LeftSemiJoin
  - Query plan for the example below has a unrecognized pattern

```
SELECT c_name, c_custkey, o_orderkey, o_orderdate, o_totalprice, sum(l_quantity)
FROM customer join orders join lineitem
    on c_custkey = o_custkey AND o_orderkey = l_orderkey
left outer join
    (
      SELECT l_orderkey tmp_orderkey
      FROM lineitem
      GROUP BY l_orderkey
      HAVING sum(l_quantity) > 300
    ) tmp
    on o_orderkey = tmp_orderkey
WHERE tmp_orderkey IS NOT NULL
GROUP BY c_name, c_custkey, o_orderkey, o_orderdate, o_totalprice
ORDER BY o_totalprice DESC, o_orderdate
```



Column Pruning Enhancement 3

- Solution and performance gain
  - Pull Request #10117
  - Added a pattern to ColumnPruning rule
  - Project nodes are inserted to the places where desired
  - Data shuffling volume is down by 90%
  - Query runs 6X faster



Agenda

- Review of Catalyst optimizer
- Rule-based Optimizations
- Cost-based Optimizations (CBO)
- Future Work
- Q & A

Cost Based Optimization

The more we know about the database objects,
the better the query plan could be generated

- Implemented statistics collection for table and columns
- Statistics are saved in Hive Metastore
- Implemented histogram to improve the estimation accuracy
- Estimate number of rows and size of intermediate results

Cost Based Optimization 2

- The optimizations we have done
  - Dynamically determine the number of partitions, and the broadcast threshold
  - Select the join type based on the cost estimation
    i.e. Sort Merge join vs. Broadcast join vs. Hash join
  - Adjust multi-table join order based on the cost estimation

  - Choose appropriate build side of a hash join operator (Prior to 1.5)

Cost Based Optimization 3

- Configuration
  - SF100 (100GB)
  - 2x E5-2680 v3 (16 cores, 32 HT)
  - 250GB memory
- Preliminary Results
  - Performance evaluation shows up to five times speedup
  - Small part of queries perform worse even with CBO (due to inaccurate estimation)

Agenda

- Review of Catalyst optimizer
- Rule-based Optimizations
- Cost-based Optimizations
- Future Work
- Q & A

Future Work

- Enumerate Space of Query Plans
  - The query plan picked by query planner is not always optimal from CBO perspective
  - Our CBO essentially only makes adjustments on the plan picked by query planner
  - Need to apply the cost model on set of equivalent query plans

- Make the cost model smarter
  - The estimation algorithm isn't accurate enough
    i.e. Bad filter factor estimation for String
    Bad estimation on the expressions/user-defined function

THANK YOU!

min.qiu@huawei.com

SPARK SUMMIT EAST
DATA SCIENCE AND ENGINEERING AT SCALE
FEBRUARY 16-18, 2016 NEW YORK CITY