How to understand and analyze Apache Hive query execution plan for performance debugging

Pengcheng Xiong and Ashutosh Chauhan
Hortonworks Inc., Apache Hive Community
{pxiong,ashutosh}@hortonworks.com

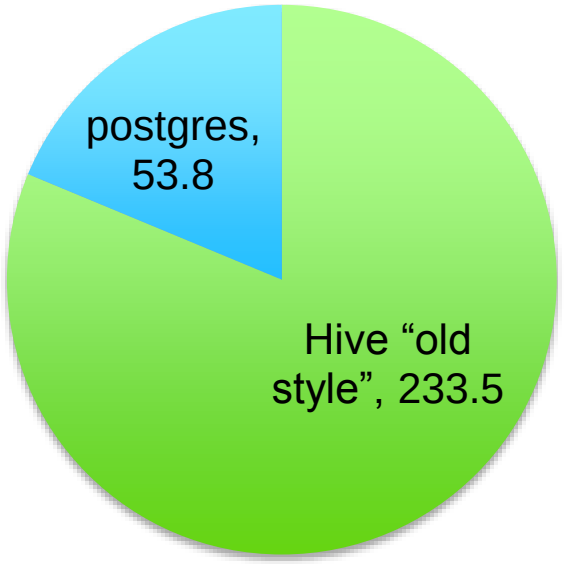
Goals:

- **WHY old style Hive explain plan is hard to read**
  - Compare the old style explain with Postgres over a body of 500+ realistic SQL queries.
- **WHAT is new style Hive explain plan**
  - Show orchestration of the tasks and operator trees, join sequences and algorithms, operator execution costs
- **HOW to performance debug a real query by analyzing the new Hive explain plan**
  - Identify the potential improvement by changing join sequence, join algorithm and etc
  - Show the real improvement by running the query in real cluster
- **Integration/interaction with other system/tools**
- **Future work**

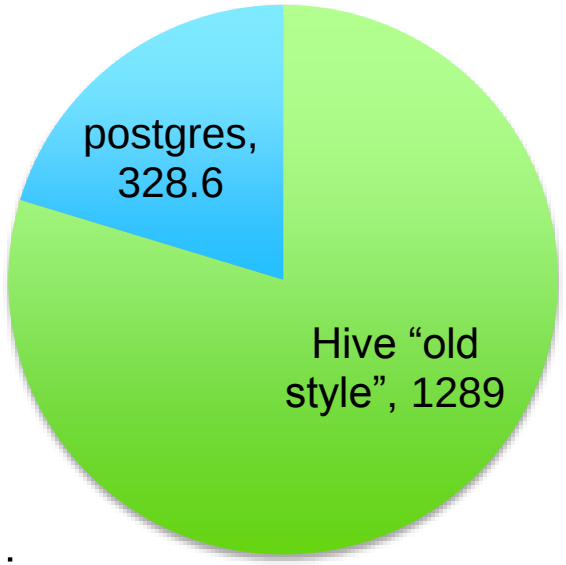
WHY old style Hive explain plan is hard to read

- **M**** company's schema and queries.**
- **Comparison of explain plans between Hive and Postgres for the 528 queries they can both execute.**
- **Hive: "explain", Postgres: "explain verbose"**

Average Lines Per Explain Plan



Average Words Per Explain Plan



N = 528 queries

Hortonworks

We can see that Hive old style explain is quite verbose, is it necessary?

```
select      a11.PBTNAME  PBTNAME
from        PMT_INVENTORY        a11
            join        LU_MONTH        a12
            on         (a11.QUARTER_ID = a12.QUARTER_ID)
where       a12.MONTH_ID in (200607, 200606)
group by    a11.PBTNAME;
```

High level plan comparison: Postgres

```
 QUERY PLAN
------------------------------------------------------------------------------
 Group (cost=3.83..3.84 rows=2 width=18)
  Output: a11.pbtname
  Group Key: a11.pbtname
  -> Sort (cost=3.83..3.84 rows=2 width=18)
     Output: a11.pbtname
     Sort Key: a11.pbtname
     -> Hash Join (cost=2.62..3.83 rows=2 width=18)
        Output: a11.pbtname
        Hash Cond: (a11.quarter_id = a12.quarter_id)
        -> Seq Scan on public.pmt_inventory a11 (cost=0.00..1.12 rows=12 width=22)
           Output: a11.quarter_id, a11.pbtname
        -> Hash (cost=2.60..2.60 rows=2 width=4)
           Output: a12.quarter_id
           -> Seq Scan on public.lu_month a12 (cost=0.00..2.60 rows=2 width=4)
              Output: a12.quarter_id
              Filter: (a12.month_id = ANY ('{200607,200606}'::integer[]))
```

16 lines

Hortonworks

High level plan comparison: Hive

Too verbose, need a magnifier!

80+ lines

```
STAGE DEPENDENCIES:
 Stage-1 is a root stage
 Stage-0 depends on stages: Stage-1

STAGE PLANS:
 Stage: Stage-1
  Tez
   Edges:
    Map 2 <- Map 1 (BROADCAST_EDGE)
    Reducer 3 <- Map 2 (SIMPLE_EDGE)
   DagName: carter_20151114133018_2f2f0101-d14d-4688-bbb1-db67055016c3:946
   Vertices:
    Map 1
      Map Operator Tree:
          TableScan
           alias: a11
           filterExpr: quarter_id is not null (type: boolean)
           Statistics: Num rows: 12 Data size: 1260 Basic stats: COMPLETE Column stats: NONE
           Filter Operator
            predicate: quarter_id is not null (type: boolean)
            Statistics: Num rows: 6 Data size: 630 Basic stats: COMPLETE Column stats: NONE
            Reduce Output Operator
             key expressions: quarter_id (type: int)
             sort order: +
             Map-reduce partition columns: quarter_id (type: int)
             Statistics: Num rows: 6 Data size: 630 Basic stats: COMPLETE Column stats: NONE
             value expressions: pbtname (type: string)
      Execution mode: vectorized
    Map 2
      Map Operator Tree:
          TableScan
           alias: a12
           filterExpr: (quarter_id is not null and (month_id) IN (200607, 200606)) (type: boolean)
           Statistics: Num rows: 48 Data size: 46752 Basic stats: COMPLETE Column stats: NONE
           Filter Operator
            predicate: (quarter_id is not null and (month_id) IN (200607, 200606)) (type: boolean)
            Statistics: Num rows: 12 Data size: 11688 Basic stats: COMPLETE Column stats: NONE
            Map Join Operator
             condition map:
                Inner Join 0 to 1
             keys:
              0 quarter_id (type: int)
              1 quarter_id (type: int)
             outputColumnNames: _col1
             input vertices:
              0 Map 1
             Statistics: Num rows: 13 Data size: 12856 Basic stats: COMPLETE Column stats: NONE
             HybridGraceHashJoin: true
             Group By Operator
              keys: _col1 (type: string)
              mode: hash
              outputColumnNames: _col0
              Statistics: Num rows: 13 Data size: 12856 Basic stats: COMPLETE Column stats: NONE
              Reduce Output Operator
               key expressions: _col0 (type: string)
               sort order: +
               Map-reduce partition columns: _col0 (type: string)
               Statistics: Num rows: 13 Data size: 12856 Basic stats: COMPLETE Column stats: NONE
      Execution mode: vectorized
    Reducer 3
      Reduce Operator Tree:
        Group By Operator
         keys: KEY._col0 (type: string)
         mode: mergepartial
         outputColumnNames: _col0
         Statistics: Num rows: 6 Data size: 5933 Basic stats: COMPLETE Column stats: NONE
         File Output Operator
          compressed: false
          Statistics: Num rows: 6 Data size: 5933 Basic stats: COMPLETE Column stats: NONE
          table:
            input format: org.apache.hadoop.mapred.TextInputFormat
            output format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
            serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe

 Stage: Stage-0
  Fetch Operator
   limit: -1
   Processor Tree:
    ListSink
```


Hortonworks

Map 2 Operator

Data flows from top to bottom

```
Map 2
      Map Operator Tree:
          TableScan
            alias: a12
            filterExpr: (quarter_id is not null and (month_id) IN (200607, 200606)) (type: boolean)
            Statistics: Num rows: 48 Data size: 46752 Basic stats: COMPLETE Column stats: NONE
            Filter Operator
              predicate: (quarter_id is not null and (month_id) IN (200607, 200606)) (type: boolean)
              Statistics: Num rows: 12 Data size: 11688 Basic stats: COMPLETE Column stats: NONE
              Map Join Operator
                condition map:
                   Inner Join 0 to 1
                keys:
                  0 quarter_id (type: int)
                  1 quarter_id (type: int)
                outputColumnNames: _col1
                input vertices:
                  0 Map 1
                Statistics: Num rows: 13 Data size: 12856 Basic stats: COMPLETE Column stats: NONE
                HybridGraceHashJoin: true
                Group By Operator
                  keys: _col1 (type: string)
                  mode: hash
                  outputColumnNames: _col0
                  Statistics: Num rows: 13 Data size: 12856 Basic stats: COMPLETE Column stats: NONE
                  Reduce Output Operator
                    key expressions: _col0 (type: string)
                    sort order: +
                    Map-reduce partition columns: _col0 (type: string)
                    Statistics: Num rows: 13 Data size: 12856 Basic stats: COMPLETE Column stats: NONE
      Execution mode: vectorized
```

Each operator has 0 or 1 child

Hortonworks

Map 2 Operator

Must scroll to another part of the plan to see what this is

```
Map 2
     Map Operator Tree:
       TableScan
        alias: a12
        filterExpr: (quarter_id is not null and (month_id) IN (200607, 200606)) (type: boolean)
        Statistics: Num rows: 48 Data size: 46752 Basic stats: COMPLETE Column stats: NONE
        Filter Operator
         predicate: (quarter_id is not null and (month_id) IN (200607, 200606)) (type: boolean)
         Statistics: Num rows: 12 Data size: 11688 Basic stats: COMPLETE Column stats: NONE
         Map Join Operator
          condition map:
             Inner Join 0 to 1
          keys:
           0 quarter_id (type: int)
           1 quarter_id (type: int)
          outputColumnNames: _col1
          input vertices:
           0 Map 1
          Statistics: Num rows: 13 Data size: 12856 Basic stats: COMPLETE Column stats: NONE
          HybridGraceHashJoin: true
          Group By Operator
           keys: _col1 (type: string)
           mode: hash
           outputColumnNames: _col0
           Statistics: Num rows: 13 Data size: 12856 Basic stats: COMPLETE Column stats: NONE
           Reduce Output Operator
            key expressions: _col0 (type: string)
            sort order: +
            Map-reduce partition columns: _col0 (type: string)
            Statistics: Num rows: 13 Data size: 12856 Basic stats: COMPLETE Column stats: NONE
     Execution mode: vectorized
```


Hortonworks

Map 1 Operator

Actual table name (PMT_INVENTORY) not mentioned anywhere, only the alias

```
Map 1
      Map Operator Tree:
        TableScan
         alias: a11
         filterExpr: quarter_id is not null (type: boolean)
         Statistics: Num rows: 12 Data size: 1260 Basic stats: COMPLETE Column stats: NONE
         Filter Operator
          predicate: quarter_id is not null (type: boolean)
          Statistics: Num rows: 6 Data size: 630 Basic stats: COMPLETE Column stats: NONE
          Reduce Output Operator
           key expressions: quarter_id (type: int)
           sort order: +
           Map-reduce partition columns: quarter_id (type: int)
           Statistics: Num rows: 6 Data size: 630 Basic stats: COMPLETE Column stats: NONE
           value expressions: pbtname (type: string)
      Execution mode: vectorized
```

Map 1 Operator

How much of this information is really necessary to SQL users?

```
Map 1
      Map Operator Tree:
        TableScan
         alias: a11
         filterExpr: quarter_id is not null (type: boolean)
         Statistics: Num rows: 12 Data size: 1260 Basic stats: COMPLETE Column stats: NONE
         Filter Operator
          predicate: quarter_id is not null (type: boolean)
          Statistics: Num rows: 6 Data size: 630 Basic stats: COMPLETE Column stats: NONE
          Reduce Output Operator
           key expressions: quarter_id (type: int)
           sort order: +
           Map-reduce partition columns: quarter_id (type: int)
           Statistics: Num rows: 6 Data size: 630 Basic stats: COMPLETE Column stats: NONE
           value expressions: pbtname (type: string)
      Execution mode: vectorized
```


Hortonworks

Back to Postgres

Data flows bottom to top

```
 QUERY PLAN
-------------------------------------------------------------------------------
 Group (cost=3.83..3.84 rows=2 width=18)
  Output: a11.pbtname
  Group Key: a11.pbtname
  -> Sort (cost=3.83..3.84 rows=2 width=18)
     Output: a11.pbtname
     Sort Key: a11.pbtname
     -> Hash Join (cost=2.62..3.83 rows=2 width=18)
        Output: a11.pbtname
        Hash Cond: (a11.quarter_id = a12.quarter_id)
        -> Seq Scan on public.pmt_inventory a11 (cost=0.00..1.12 rows=12 width=22)
           Output: a11.quarter_id, a11.pbtname
        -> Hash (cost=2.60..2.60 rows=2 width=4)
           Output: a12.quarter_id
           -> Seq Scan on public.lu_month a12 (cost=0.00..2.60 rows=2 width=4)
              Output: a12.quarter_id
              Filter: (a12.month_id = ANY ('{200607,200606}'::integer[]))
```


Hortonworks

Back to Postgres

Operators have multiple children when it makes sense

```
 QUERY PLAN
------------------------------------------------------------------------------
 Group (cost=3.83..3.84 rows=2 width=18)
  Output: a11.pbtname
  Group Key: a11.pbtname
  -> Sort (cost=3.83..3.84 rows=2 width=18)
     Output: a11.pbtname
     Sort Key: a11.pbtname
     -> Hash Join (cost=2.62..3.83 rows=2 width=18)
        Output: a11.pbtname
        Hash Cond: (a11.quarter_id = a12.quarter_id)
        -> Seq Scan on public.pmt_inventory a11 (cost=0.00..1.12 rows=12 width=22)
           Output: a11.quarter_id, a11.pbtname
        -> Hash (cost=2.60..2.60 rows=2 width=4)
           Output: a12.quarter_id
           -> Seq Scan on public.lu_month a12 (cost=0.00..2.60 rows=2 width=4)
              Output: a12.quarter_id
              Filter: (a12.month_id = ANY ('{200607,200606}'::integer[]))
```


Hortonworks

Back to Postgres

Join is done using a scan of pmt_inventory and a hash following a scan of lu_month.
All this info is available without referring to a stage plan.
IOW you don't have to jump around in the plan.

```
 QUERY PLAN
------------------------------------------------------------------------------
 Group (cost=3.83..3.84 rows=2 width=18)
  Output: a11.pbtname
  Group Key: a11.pbtname
  -> Sort (cost=3.83..3.84 rows=2 width=18)
     Output: a11.pbtname
     Sort Key: a11.pbtname
     -> Hash Join (cost=2.62..3.83 rows=2 width=18)
        Output: a11.pbtname
        Hash Cond: (a11.quarter_id = a12.quarter_id)
        -> Seq Scan on public.pmt_inventory a11 (cost=0.00..1.12 rows=12 width=22)
           Output: a11.quarter_id, a11.pbtname
        -> Hash (cost=2.60..2.60 rows=2 width=4)
           Output: a12.quarter_id
           -> Seq Scan on public.lu_month a12 (cost=0.00..2.60 rows=2 width=4)
              Output: a12.quarter_id
              Filter: (a12.month_id = ANY ('{200607,200606}'::integer[]))
```


Hortonworks

Back to Postgres

Actual schema / table names visible

```
 QUERY PLAN
------------------------------------------------------------------------------
 Group (cost=3.83..3.84 rows=2 width=18)
  Output: a11.pbtname
  Group Key: a11.pbtname
  -> Sort (cost=3.83..3.84 rows=2 width=18)
     Output: a11.pbtname
     Sort Key: a11.pbtname
     -> Hash Join (cost=2.62..3.83 rows=2 width=18)
        Output: a11.pbtname
        Hash Cond: (a11.quarter_id = a12.quarter_id)
        -> Seq Scan on public.pmt_inventory a11 (cost=0.00..1.12 rows=12 width=22)
           Output: a11.quarter_id, a11.pbtname
        -> Hash (cost=2.60..2.60 rows=2 width=4)
           Output: a12.quarter_id
           -> Seq Scan on public.lu_month a12 (cost=0.00..2.60 rows=2 width=4)
              Output: a12.quarter_id
              Filter: (a12.month_id = ANY ('{200607,200606}'::integer[]))
```


Hortonworks

Back to Postgres

Cost mentioned once per operator
Cost monotonically increases as you go up

```
 QUERY PLAN
------------------------------------------------------------------------------
 Group (cost=3.83..3.84 rows=2 width=18)
  Output: a11.pbtname
  Group Key: a11.pbtname
  -> Sort (cost=3.83..3.84 rows=2 width=18)
     Output: a11.pbtname
     Sort Key: a11.pbtname
     -> Hash Join (cost=2.62..3.83 rows=2 width=18)
        Output: a11.pbtname
        Hash Cond: (a11.quarter_id = a12.quarter_id)
        -> Seq Scan on public.pmt_inventory a11 (cost=0.00..1.12 rows=12 width=22)
           Output: a11.quarter_id, a11.pbtname
        -> Hash (cost=2.60..2.60 rows=2 width=4)
           Output: a12.quarter_id
           -> Seq Scan on public.lu_month a12 (cost=0.00..2.60 rows=2 width=4)
              Output: a12.quarter_id
              Filter: (a12.month_id = ANY ('{200607,200606}'::integer[]))
```


Hortonworks

WHAT is new style Hive explain plan (HIVE-9780)

- **Set hive.explain.user=true; (by default). Use Tez, LLAP, etc**

```
Stage-1
    Reducer 3
    File Output Operator [FS_14]
      Group By Operator [GBY_12] (rows=8 width=101)
        Output:["_col0"],keys:KEY._col0
      <-Map 2 [SIMPLE_EDGE]
        SHUFFLE [RS_11]
          PartitionCols:_col0
          Group By Operator [GBY_10] (rows=8 width=101)
            Output:["_col0"],keys:_col1
            Map Join Operator [MAPJOIN_19] (rows=33 width=101)
              Conds:RS_6._col0=SEL_5._col1(Inner),HybridGraceHashJoin:true,Output:["_col1"]
            <-Map 1 [BROADCAST_EDGE]
              BROADCAST [RS_6]
                PartitionCols:_col0
                Select Operator [SEL_2] (rows=12 width=105)
                  Output:["_col0","_col1"]
                  Filter Operator [FIL_17] (rows=12 width=105)
                    predicate:quarter_id is not null
                    TableScan [TS_0] (rows=12 width=105)
                      m****@pmt_inventory,a11,Tbl:COMPLETE,Col:COMPLETE,Output:["quarter_id","pbtname"]
            <-Select Operator [SEL_5] (rows=48 width=8)
                Output:["_col1"]
                Filter Operator [FIL_18] (rows=48 width=8)
                  predicate:((month_id) IN (200607, 200606) and quarter_id is not null)
                  TableScan [TS_3] (rows=48 width=8)
                    m****@lu_month,a12,Tbl:COMPLETE,Col:COMPLETE,Output:["month_id","quarter_id"]
```

Immediate Notes:

1. **Much smaller**
2. **Can be read in order**

WHAT is new style Hive explain plan (HIVE-9780)

Data flows bottom to top

```
Stage-1
    Reducer 3
    File Output Operator [FS_14]
      Group By Operator [GBY_12] (rows=8 width=101)
        Output:["_col0"],keys:KEY._col0
      <-Map 2 [SIMPLE_EDGE]
        SHUFFLE [RS_11]
          PartitionCols:_col0
          Group By Operator [GBY_10] (rows=8 width=101)
            Output:["_col0"],keys:_col1
            Map Join Operator [MAPJOIN_19] (rows=33 width=101)
              Conds:RS_6._col0=SEL_5._col1(Inner),HybridGraceHashJoin:true,Output:["_col1"]
            <-Map 1 [BROADCAST_EDGE]
              BROADCAST [RS_6]
                PartitionCols:_col0
                Select Operator [SEL_2] (rows=12 width=105)
                  Output:["_col0","_col1"]
                  Filter Operator [FIL_17] (rows=12 width=105)
                    predicate:quarter_id is not null
                    TableScan [TS_0] (rows=12 width=105)
                      m****@pmt_inventory,a11,Tbl:COMPLETE,Col:COMPLETE,Output:["quarter_id","pbtname"]
            <-Select Operator [SEL_5] (rows=48 width=8)
                Output:["_col1"]
                Filter Operator [FIL_18] (rows=48 width=8)
                  predicate:((month_id) IN (200607, 200606) and quarter_id is not null)
                  TableScan [TS_3] (rows=48 width=8)
                    m****@lu_month,a12,Tbl:COMPLETE,Col:COMPLETE,Output:["month_id","quarter_id"]
```

WHAT is new style Hive explain plan (HIVE-9780)

```
Stage-1
    Reducer 3
    File Output Operator [FS_14]
      Group By Operator [GBY_12] (rows=8 width=101)
        Output:["_col0"],keys:KEY._col0
      <-Map 2 [SIMPLE_EDGE]
        SHUFFLE [RS_11]
          PartitionCols:_col0
          Group By Operator [GBY_10] (rows=8 width=101)
            Output:["_col0"],keys:_col1
          Map Join Operator [MAPJOIN_19] (rows=33 width=101)
            Conds:RS_6._col0=SEL_5._col1(Inner),HybridGraceHashJoin:true,Output:["_col1"]
          <-Map 1 [BROADCAST_EDGE]
            BROADCAST [RS_6]
              PartitionCols:_col0
              Select Operator [SEL_2] (rows=12 width=105)
                Output:["_col0","_col1"]
                Filter Operator [FIL_17] (rows=12 width=105)
                  predicate:quarter_id is not null
                  TableScan [TS_0] (rows=12 width=105)
                    m****@pmt_inventory,a11,Tbl:COMPLETE,Col:COMPLETE,Output:["quarter_id","pbtname"]
          <-Select Operator [SEL_5] (rows=48 width=8)
              Output:["_col1"]
              Filter Operator [FIL_18] (rows=48 width=8)
                predicate:((month_id) IN (200607, 200606) and quarter_id is not null)
                TableScan [TS_3] (rows=48 width=8)
                  m****@lu_month,a12,Tbl:COMPLETE,Col:COMPLETE,Output:["month_id","quarter_id"]
```

Operators have multiple children when it makes sense

WHAT is new style Hive explain plan (HIVE-9780)

```
Stage-1
    Reducer 3
    File Output Operator [FS_14]
      Group By Operator [GBY_12] (rows=8 width=101)
        Output:["_col0"],keys:KEY._col0
      <-Map 2 [SIMPLE_EDGE]
        SHUFFLE [RS_11]
          PartitionCols:_col0
          Group By Operator [GBY_10] (rows=8 width=101)
            Output:["_col0"],keys:_col1
            Map Join Operator [MAPJOIN_19] (rows=33 width=101)
              Conds:RS_6._col0=SEL_5._col1(Inner),HybridGraceHashJoin:true,Output:["_col1"]
            <-Map 1 [BROADCAST_EDGE]
              BROADCAST [RS_6]
                PartitionCols:_col0
                Select Operator [SEL_2] (rows=12 width=105)
                  Output:["_col0","_col1"]
                  Filter Operator [FIL_17] (rows=12 width=105)
                    predicate:quarter_id is not null
                    TableScan [TS_0] (rows=12 width=105)
                      m****@pmt_inventory,a11,Tbl:COMPLETE,Col:COMPLETE,Output:["quarter_id","pbtname"]
            <-Select Operator [SEL_5] (rows=48 width=8)
                Output:["_col1"]
                Filter Operator [FIL_18] (rows=48 width=8)
                  predicate:((month_id) IN (200607, 200606) and quarter_id is not null)
                  TableScan [TS_3] (rows=48 width=8)
                    m****@lu_month,a12,Tbl:COMPLETE,Col:COMPLETE,Output:["month_id","quarter_id"]
```

Join's information is clear
pmt_inventory is broadcasted to lu_month
and a MapJoin is done

WHAT is new style Hive explain plan (HIVE-9780)

Cost mentioned once per operator

```
Stage-1
    Reducer 3
    File Output Operator [FS_14]
      Group By Operator [GBY_12] (rows=8 width=101)
        Output:["_col0"],keys:KEY._col0
      <-Map 2 [SIMPLE_EDGE]
        SHUFFLE [RS_11]
          PartitionCols:_col0
          Group By Operator [GBY_10] (rows=8 width=101)
            Output:["_col0"],keys:_col1
            Map Join Operator [MAPJOIN_19] (rows=33 width=101)
              Conds:RS_6._col0=SEL_5._col1(Inner),HybridGraceHashJoin:true,Output:["_col1"]
            <-Map 1 [BROADCAST_EDGE]
              BROADCAST [RS_6]
                PartitionCols:_col0
                Select Operator [SEL_2] (rows=12 width=105)
                  Output:["_col0","_col1"]
                  Filter Operator [FIL_17] (rows=12 width=105)
                    predicate:quarter_id is not null
                    TableScan [TS_0] (rows=12 width=105)
                      m****@pmt_inventory,a11,Tbl:COMPLETE,Col:COMPLETE,Output:["quarter_id","pbtname"]
            <-Select Operator [SEL_5] (rows=48 width=8)
                Output:["_col1"]
                Filter Operator [FIL_18] (rows=48 width=8)
                  predicate:((month_id) IN (200607, 200606) and quarter_id is not null)
                  TableScan [TS_3] (rows=48 width=8)
                    m****@lu_month,a12,Tbl:COMPLETE,Col:COMPLETE,Output:["month_id","quarter_id"]
```

HOW to performance debug a real query (TPC-DS Q3, 1TB)

```
select
    dt.d_year, item.i_brand_id brand_id, item.i_brand brand, sum(ss_ext_sales_price) sum_agg
from
    date_dim dt, store_sales, item
where
    dt.d_date_sk = store_sales.ss_sold_date_sk
    and store_sales.ss_item_sk = item.i_item_sk
    and item.i_manufact_id = 436
    and dt.d_moy = 12
group by dt.d_year , item.i_brand , item.i_brand_id
order by dt.d_year , sum_agg desc , brand_id
limit 10
```

partitioned by ss_sold_date_sk

Hortonworks

```
<-Reducer 3 [SIMPLE_EDGE]
 SHUFFLE [RS_15]
  PartitionCols:_col0, _col1, _col2
  Group By Operator [GBY_14] (rows=1 width=116)
   Output:["_col0","_col1","_col2","_col3"],aggregations:["sum(_col45)"],keys:_col6, _col65, _col64
   Select Operator [SEL_13] (rows=76515 width=128)
    Output:["_col6","_col65","_col64","_col45"]
    Filter Operator [FIL_23] (rows=76515 width=128)
     predicate:((_col0 = _col53) and (_col32 = _col57))
     Merge Join Operator [MERGEJOIN_28] (rows=306061 width=128)
      Conds:RS_8._col32=RS_34.i_item_sk(Inner),Output:["_col0","_col6","_col32","_col45","_col53","_col57"
     <-Map 7 [SIMPLE_EDGE] vectorized
      SHUFFLE [RS_34]
       PartitionCols:i_item_sk
       Filter Operator [FIL_33] (rows=434 width=111)
        predicate:(i_item_sk is not null and (i_manufact_id = 436))
        TableScan [TS_2] (rows=300000 width=111)
         tpcds_bin_partitioned_orc_1000@item,item,Tbl:COMPLETE,Col:COMPLETE,Output:["i_item_sk","i_brand_id","i_brand","i_manufact_id"]
     <-Reducer 2 [SIMPLE_EDGE]
      SHUFFLE [RS_8]
       PartitionCols:_col32
       Merge Join Operator [MERGEJOIN_27] (rows=211562452 width=20)
        Conds:RS_30.d_date_sk=RS_32.ss_sold_date_sk(Inner),Output:["_col0","_col6","_col32","_col45","_col53"]
       <-Map 1 [SIMPLE_EDGE] vectorized
        SHUFFLE [RS_30]
         PartitionCols:d_date_sk
         Filter Operator [FIL_29] (rows=5619 width=12)
          predicate:(d_date_sk is not null and (d_moy = 12))
          TableScan [TS_0] (rows=73049 width=12)
           tpcds_bin_partitioned_orc_1000@date_dim,dt,Tbl:COMPLETE,Col:COMPLETE,Output:["d_date_sk","d_year","d_moy"]
       <-Map 6 [SIMPLE_EDGE] vectorized
        SHUFFLE [RS_32]
         PartitionCols:ss_sold_date_sk
         Filter Operator [FIL_31] (rows=2750387156 width=11)
          predicate:ss_item_sk is not null
          TableScan [TS_1] (rows=2750387156 width=11)
           tpcds_bin_partitioned_orc_1000@store_sales,store_sales,Tbl:COMPLETE,Col:COMPLETE,Output:["ss_item_sk","ss_ext_sales_price"]
```

Original plan runs 163.33s. Sounds like column pruning and predicate push down are working fine. However, the join sequence store_sales✕date_dim✕item is not good enough. A better one is store_sales✕item✕date_dim

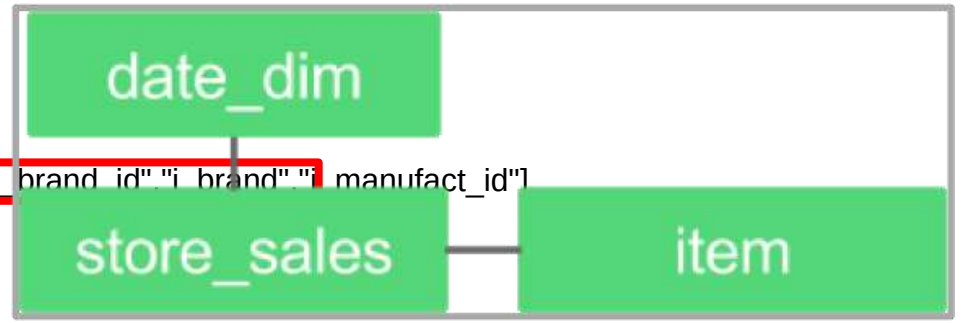


Table	Cardinality	Cardinality after filter	Selectivity
date_dim	73K	5619	7.6%
item	300K	434	0.14%

```
SHUFFLE [RS_17]
  PartitionCols:_col0, _col1, _col2
  Group By Operator [GBY_16] (rows=9 width=116)
    Output:["_col0","_col1","_col2","_col3"],aggregations:["sum(_col1)"],keys:_col8, _col4, _col5
    Select Operator [SEL_15] (rows=306061 width=112)
      Output:["_col8","_col4","_col5","_col1"]
      Merge Join Operator [MERGEJOIN_29] (rows=306061 width=112)
        Conds:RS_12._col2=RS_38._col0(Inner),Output:["_col1","_col4","_col5","_col8"]
      <-Map 7 [SIMPLE_EDGE] vectorized
        SHUFFLE [RS_38]
          PartitionCols:_col0
          Select Operator [SEL_37] (rows=5619 width=12)
            Output:["_col0","_col1"]
            Filter Operator [FIL_36] (rows=5619 width=12)
              predicate:((d_moy = 12) and d_date_sk is not null)
              TableScan [TS_6] (rows=73049 width=12)
                tpcds_bin_partitioned_orc_1000@date_dim,dt,Tbl:COMPLETE,Col:COMPLETE,Output:["d_date_sk","d_year","d_moy"]
      <-Reducer 2 [SIMPLE_EDGE]
        SHUFFLE [RS_12]
          PartitionCols:_col2
          Merge Join Operator [MERGEJOIN_28] (rows=3978894 width=112)
            Conds:RS_32._col0=RS_35._col0(Inner),Output:["_col1","_col2","_col4","_col5"]
          <-Map 1 [SIMPLE_EDGE] vectorized
            SHUFFLE [RS_32]
              PartitionCols:_col0
              Select Operator [SEL_31] (rows=2750387156 width=11)
                Output:["_col0","_col1","_col2"]
                Filter Operator [FIL_30] (rows=2750387156 width=11)
                  predicate:ss_item_sk is not null
                  TableScan [TS_0] (rows=2750387156 width=11)
                    tpcds_bin_partitioned_orc_1000@store_sales,store_sales,Tbl:COMPLETE,Col:COMPLETE,Output:["ss_item_sk","ss_ext_sales_price"]
          <-Map 6 [SIMPLE_EDGE] vectorized
            SHUFFLE [RS_35]
              PartitionCols:_col0
              Select Operator [SEL_34] (rows=434 width=111)
                Output:["_col0","_col1","_col2"]
                Filter Operator [FIL_33] (rows=434 width=111)
                  predicate:((i_manufact_id = 436) and i_item_sk is not null)
                  TableScan [TS_3] (rows=300000 width=111)
                    tpcds_bin_partitioned_orc_1000@item,item,Tbl:COMPLETE,Col:COMPLETE,Output:["i_item_sk","i_brand_id","i_brand","i_manufact_id"]
```

CBO on, new plan runs 143.97s with new join sequence store_sales✕item✕date_dim.

The input data size of one branch of join is pretty small, should use map join, rather than merge join.

```
SHUFFLE [RS_44]
  PartitionCols:_col0, _col1, _col2
  Group By Operator [GBY_43] (rows=9 width=116)
    Output:["_col0","_col1","_col2","_col3"],aggregations:["sum(_col1)"],keys:_col8, _col4, _col5
    Select Operator [SEL_42] (rows=306061 width=112)
      Output:["_col8","_col4","_col5","_col1"]
      Map Join Operator [MAPJOIN_41] (rows=306061 width=112)
        Conds:MAPJOIN_40._col2=RS_37._col0(Inner),HybridGraceHashJoin:true,Output:["_col1","_col4","_col5","_col8"]
      <-Map 5 [BROADCAST_EDGE] vectorized
        BROADCAST [RS_37]
          PartitionCols:_col0
          Select Operator [SEL_36] (rows=5619 width=12)
            Output:["_col0","_col1"]
            Filter Operator [FIL_35] (rows=5619 width=12)
              predicate:((d_moy = 12) and d_date_sk is not null)
              TableScan [TS_6] (rows=73049 width=12)
                tpcds_bin_partitioned_orc_1000@date_dim,dt,Tbl:COMPLETE,Col:COMPLETE,Output:["d_date_sk","d_year","d_moy"]
      <-Map Join Operator [MAPJOIN_40] (rows=3978894 width=112)
        Conds:SEL_39._col0=RS_34._col0(Inner),HybridGraceHashJoin:true,Output:["_col1","_col2","_col4","_col5"]
      <-Map 4 [BROADCAST_EDGE] vectorized
        BROADCAST [RS_34]
          PartitionCols:_col0
          Select Operator [SEL_33] (rows=434 width=111)
            Output:["_col0","_col1","_col2"]
            Filter Operator [FIL_32] (rows=434 width=111)
              predicate:((i_manufact_id = 436) and i_item_sk is not null)
              TableScan [TS_3] (rows=300000 width=111)
                tpcds_bin_partitioned_orc_1000@item,item,Tbl:COMPLETE,Col:COMPLETE,Output:["i_item_sk","i_brand_id","i_brand","i_manufact_id"]
      <-Select Operator [SEL_39] (rows=2750387156 width=11)
          Output:["_col0","_col1","_col2"]
          Filter Operator [FIL_38] (rows=2750387156 width=11)
            predicate:ss_item_sk is not null
            TableScan [TS_0] (rows=2750387156 width=11)
              tpcds_bin_partitioned_orc_1000@store_sales,store_sales,Tbl:COMPLETE,Col:COMPLETE,Output:["ss_item_sk","ss_ext_sales_price"]
```

Increase hive.auto.convert.join.noconditionaltask.size= 1,359,688,499, we can see it is now using map join operators. New plan runs 45.84s.

store_sales is a partitioned table on the join key ss_sold_date_sk with date_dim table.

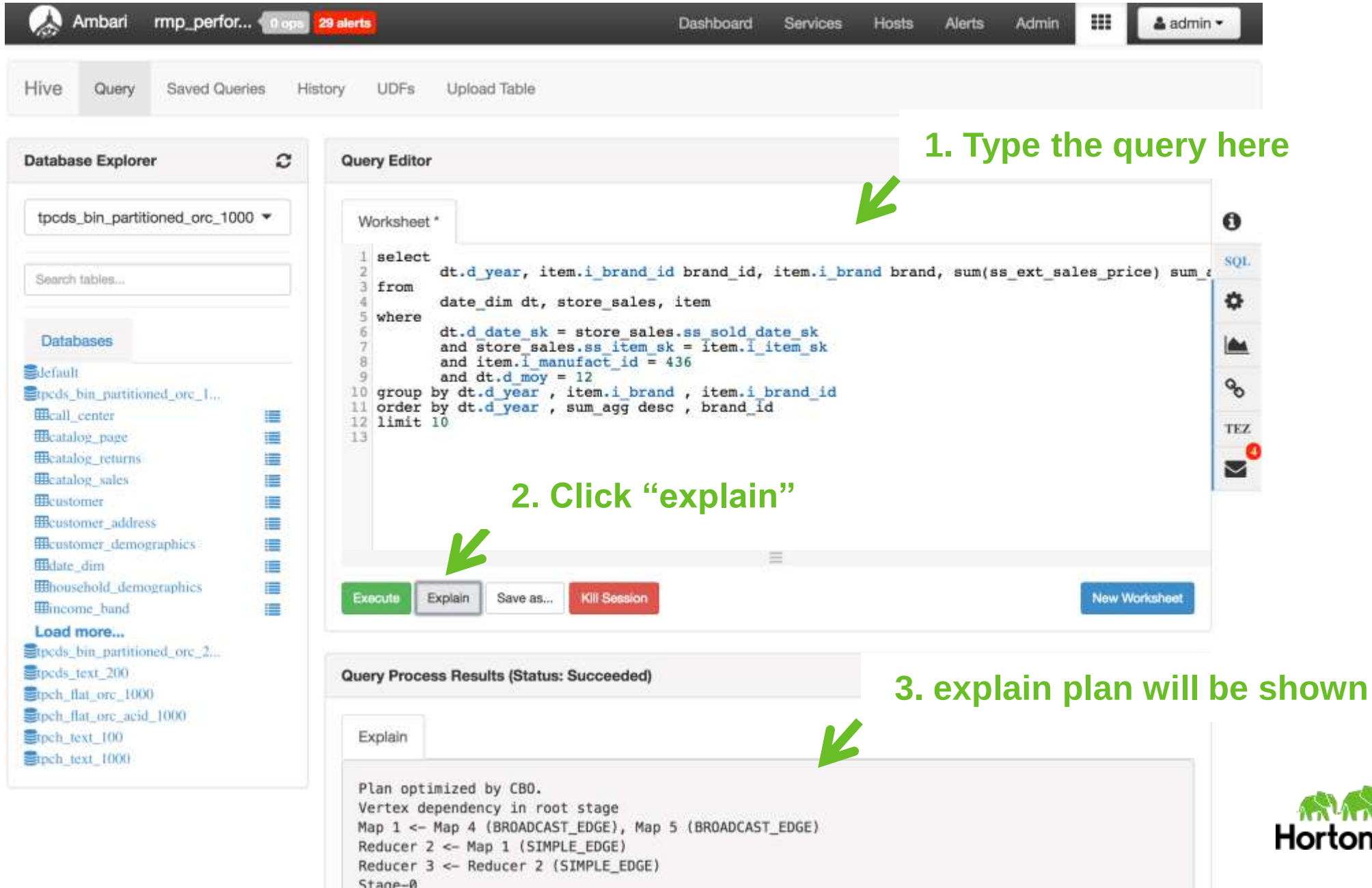
```
SHUFFLE [RS_55]
  PartitionCols:_col0, _col1, _col2
  Group By Operator [GBY_54] (rows=9 width=116)
    Output:["_col0","_col1","_col2","_col3"],aggregations:["sum(_col1)"],keys:_col8, _col4, _col5
    Select Operator [SEL_53] (rows=306061 width=112)
      Output:["_col8","_col4","_col5","_col1"]
      Map Join Operator [MAPJOIN_52] (rows=306061 width=112)
        Conds:MAPJOIN_51._col2=RS_45._col0(Inner),HybridGraceHashJoin:true,Output:["_col1","_col4","_col5","_col8"]
      <-Map 5 [BROADCAST_EDGE] vectorized
        BROADCAST [RS_45]
          PartitionCols:_col0
          Select Operator [SEL_44] (rows=5619 width=12)
            Output:["_col0","_col1"]
            Filter Operator [FIL_43] (rows=5619 width=12)
              predicate:((d_moy = 12) and d_date_sk is not null)
              TableScan [TS_6] (rows=73049 width=12)
                tpcds_bin_partitioned_orc_1000@date_dim,dt,Tbl:COMPLETE,Col:COMPLETE,Output:["d_date_sk","d_year","d_moy"]
      Dynamic Partitioning Event Operator [EVENT_48] (rows=2809 width=12)
        Group By Operator [GBY_47] (rows=2809 width=12)
          Output:["_col0"],keys:_col0
          Select Operator [SEL_46] (rows=5619 width=12)
            Output:["_col0"]
             Please refer to the previous Select Operator [SEL_44]
      <-Map Join Operator [MAPJOIN_51] (rows=3978894 width=112)
          Conds:SEL_50._col0=RS_42._col0(Inner),HybridGraceHashJoin:true,Output:["_col1","_col2","_col4","_col5"]
        <-Map 4 [BROADCAST_EDGE] vectorized
          BROADCAST [RS_42]
            PartitionCols:_col0
            Select Operator [SEL_41] (rows=434 width=111)
              Output:["_col0","_col1","_col2"]
              Filter Operator [FIL_40] (rows=434 width=111)
                predicate:((i_manufact_id = 436) and i_item_sk is not null)
                TableScan [TS_3] (rows=300000 width=111)
                  tpcds_bin_partitioned_orc_1000@item,item,Tbl:COMPLETE,Col:COMPLETE,Output:["i_item_sk","i_brand_id","i_brand","i_manufact_id"]
        <-Select Operator [SEL_50] (rows=2750387156 width=11)
            Output:["_col0","_col1","_col2"]
            Filter Operator [FIL_49] (rows=2750387156 width=11)
              predicate:ss_item_sk is not null
              TableScan [TS_0] (rows=2750387156 width=11)
                tpcds_bin_partitioned_orc_1000@store_sales,store_sales,Tbl:COMPLETE,Col:COMPLETE,Output:["ss_item_sk","ss_ext_sales_price"]
```

By setting hive.tez.dynamic.partition.pruning=true**, we can see dynamic partitioning event operators. See more about this in HIVE-7826. New plan runs 31.35s.**

In the run time, dynamic partition event operator will send values needed to prune to the application master - where splits are generated and tasks are submitted. Using these values we can strip out any unneeded partitions dynamically, while the query is running.

Performance debugging summary (TPC-DS Q3, 1TB)

Integration with Apache Ambari



1. Type the query here

2. Click “explain”

3. explain plan will be shown

Hortonworks

Can be used along with Tez vertex runtime stats

- “set hive.tez.exec.print.summary=true;”

```
Status: Running (Executing on YARN cluster with App id application_1466700718395_1117)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........  SUCCEEDED    364        364        0        0       0       0
Map 10 .........  SUCCEEDED      5          5        0        0       0       0
Map 11 .........  SUCCEEDED    155        155        0        0       0       0
Map 13 .........  SUCCEEDED      1          1        0        0       0       0
Map 14 .........  SUCCEEDED      5          5        0        0       0       0
Map 5 ..........  SUCCEEDED      1          1        0        0       0       0
Map 6 ..........  SUCCEEDED      5          5        0        0       0       0
Map 7 ..........  SUCCEEDED    261        261        0        0       0       0
Map 9 ..........  SUCCEEDED      1          1        0        0       0       0
Reducer 12 .....  SUCCEEDED     14         14        0        0       0       0
Reducer 2 ......  SUCCEEDED     53         53        0        0       0       0
Reducer 3 ......  SUCCEEDED     48         48        0        0       0       0
Reducer 4 ......  SUCCEEDED      1          1        0        0       0       0
Reducer 8 ......  SUCCEEDED     28         28        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 14/14  [==========================>>] 100%  ELAPSED TIME: 68.41 s
--------------------------------------------------------------------------------
Status: DAG finished successfully in 68.41 seconds

METHOD                          DURATION(ms)
parse                                 12
semanticAnalyze                    5,805
TezBuildDag                          999
TezSubmitToRunningDag                403
TotalPrepTime                     19,792

VERTICES         TOTAL_TASKS  FAILED_ATTEMPTS  KILLED_TASKS  DURATION_SECONDS  CPU_TIME_MILLIS  GC_TIME_MILLIS  INPUT_RECORDS  OUTPUT_RECORDS
Map 1                    364                0             0             47.95        7,483,570         180,910    111,939,501       8,713,152
Map 10                     5                0             0              6.65           53,610             303      1,600,000       1,600,000
Map 11                   155                0             0             50.25        4,402,300          80,010     30,450,613       2,299,535
Map 13                     1                0             0              1.32            3,480              42         10,000             366
Map 14                     5                0             0              5.58           51,100             242      1,600,000       1,600,000
Map 5                      1                0             0              1.74            3,480              50         10,000             366
Map 6                      5                0             0              6.08           44,020             193      1,600,000       1,600,000
Map 7                    261                0             0             48.59        5,958,830         116,856     58,763,101       5,904,712
Map 9                      1                0             0              1.53            3,560              44         10,000             366
Reducer 12                14                0             0             12.88          256,010           2,077      2,299,535       2,299,535
Reducer 2                 53                0             0             21.30        1,046,620          12,535      8,713,152       8,713,152
Reducer 3                 48                0             0              8.53          882,790           6,192     16,917,399              48
Reducer 4                  1                0             0              2.74            1,590               0             48               0
Reducer 8                 28                0             0             22.74          552,280           6,485      5,904,712       5,904,712
OK
7893781
Time taken: 88.875 seconds, Fetched: 1 row(s)
```

Runtime stats

- Get more insights on query performance

Hortonworks

Summary

- **Show old style Hive explain plan is hard to read.**
  - Verbose with too much redundant information, hard to follow how data flows, cost of operator is unclear
  - Compare with Postgres over a body of 500+ realistic SQL queries and identify the candidate improving points
- **Introduce new style Hive explain plan**
  - Use a concrete example to help understand the explain: execution cost, join sequence and orchestration of the operator tree
- **Use the new Hive explain plan to performance debug TPC-DS Q3**
  - Show the improvement after join re-ordering, join selection, and dynamic partition pruning
- **Integration/interaction with other system/tools**

Hortonworks

Future work -- Some gaps remain after HIVE-9780

- Put the real schema, table and column names in the explain plan, e.g., no more _col0 etc.
  - This will help users to understand the plan better
  - HIVE-8681: CBO: Column names are missing from join expression in Map join with CBO enabled
- Get an equivalent of “EXPLAIN ANALYZE” – such as operator level runtime stats and warnings.
  - This will help users to find out the gap between estimated cost and real cost
  - HIVE-14362: Support explain analyze in Hive

```
SHUFFLE [RS_55]
  PartitionCols:d_year, i_brand, i_brand_id
  Group By Operator [GBY_54] (rows=9/10 width=116)
    Output:["d_year","sum_agg","i_brand","i_brand_id"],aggregations:["sum(ss_ext_sales_price)"],keys:d_year, i_brand, i_brand_id
    Select Operator [SEL_53] (rows=306061/324651 width=112)
      Output:["d_year","i_brand","i_brand_id","ss_ext_sales_price"]
      Map Join Operator [MAPJOIN_52] (rows=306061/324651 width=112)
        Conds:MAPJOIN_51. ss_sold_date_sk=RS_45. d_date_sk(Inner),HybridGraceHashJoin:true,Output:["d_year","i_brand","i_brand_id","ss_ext_sales_price"]
      <-Map 5 [BROADCAST_EDGE] vectorized
        BROADCAST [RS_45]
          PartitionCols:d_date_sk
          Select Operator [SEL_44] (rows=5619/6034 width=12)
            Output:["d_date_sk","d_year"]
            Filter Operator [FIL_43] (rows=5619/6034 width=12)
              predicate:((d_moy = 12) and d_date_sk is not null)
              TableScan [TS_6] (rows=73049/73049 width=12)
                tpcds_bin_partitioned_orc_1000@date_dim,dt,Tbl:COMPLETE,Col:COMPLETE,Output:["d_date_sk","d_year","d_moy"]
        Dynamic Partitioning Event Operator [EVENT_48] (rows=2809 width=12)
          Group By Operator [GBY_47] (rows=2809/2324 width=12)
            Output:["d_date_sk"],keys:d_date_sk
            Select Operator [SEL_46] (rows=5619/6034 width=12)
              Output:["d_date_sk"]
               Please refer to the previous Select Operator [SEL_44]
      <-Map Join Operator [MAPJOIN_51] (rows=3978894/4202377 width=112)
          Conds:SEL_50.ss_item_sk=RS_42. i_item_sk(Inner),HybridGraceHashJoin:true,Output:["ss_ext_sales_price",” ss_sold_date_sk","i_brand","i_brand_id"]
        <-Map 4 [BROADCAST_EDGE] vectorized
          BROADCAST [RS_42]
            PartitionCols:i_item_sk
            Select Operator [SEL_41] (rows=434/453 width=111)
              Output:[” i_item_sk","i_brand_id","i_brand"]
              Filter Operator [FIL_40] (rows=434/453 width=111)
                predicate:((i_manufact_id = 436) and i_item_sk is not null)
                TableScan [TS_3] (rows=300000/300000 width=111)
                  tpcds_bin_partitioned_orc_1000@item,item,Tbl:COMPLETE,Col:COMPLETE,Output:["i_item_sk","i_brand_id","i_brand","i_manufact_id"]
        <-Select Operator [SEL_50] (rows=2750387156/2750387156  width=11)
            Output:["ss_item_sk","ss_ext_sales_price","ss_sold_date_sk"]
            Filter Operator [FIL_49] (rows=2750387156/2750387156 width=11)
              predicate:ss_item_sk is not null
              TableScan [TS_0] (rows=2750387156/2750387156 width=11)
                tpcds_bin_partitioned_orc_1000@store_sales,store_sales,Tbl:COMPLETE,Col:COMPLETE,Output:["ss_item_sk","ss_ext_sales_price"]
```

Acknowledgement

- We thank all the anonymous reviewers' votes to give us this opportunity to share our work.
- Part of the slides are borrowed from or modified based on Carter Shanklin and Rajesh Balamohan's slides.
- We thank Gunther Hagleitner for all the support and inputs.
- We thank Sapin Amin for setting up the testing cluster.

Hortonworks

Thank you! Questions?

Hortonworks