




[Log in](#) to create and rate content, and to follow, bookmark, and share content with other members.

How to enable Fetch Task instead of MapReduce Job for simple query in Hive


 Document created by **Hao Zhu**  on Feb 18, 2016

 **Version 1**

 Like • 0

 Comment • 0



Author:  [Hao Zhu](#)

Original Publication Date: January 15, 2015


Goal:

Certain simple Hive queries can utilize fetch task, which can avoid the overhead of starting MapReduce job.This article explains how to enable this feature.

Solution:

1. hive.fetch.task.conversion

This parameter controls which kind of simple query can be converted to a single fetch task.It was added in Hive 0.10 per HIVE-2925.

- Value "none" is added in Hive 0.14 to disable this feature, per HIVE-8389.
- Value "minimal" means **SELECT ***, **FILTER** on partition columns (**WHERE** and **HAVING** clauses), **LIMIT** only.

In this case, "select * from table limit 1" may finish very soon however "select col0 from table limit 1" may take longer since it needs to start MapReduce job.Look at below explain plans:

```
hive> set hive.fetch.task.conversion;
hive.fetch.task.conversion=minimal
hive> explain select * from passwords limit 1;
OK
STAGE DEPENDENCIES:
  Stage-0 is a root stage

STAGE PLANS:
  Stage: Stage-0
    Fetch Operator
    limit: 1
    Processor Tree:
      TableScan
      alias: passwords
      Statistics: Num rows: 9963904 Data size: 477218560 Basic stats: COMPLETE Column stats: NONE
    Select Operator
      expressions: col0 (type: string), col1 (type: string), col2 (type: string), col3 (type: string), col4 (type: string), col5 (type: string), col6 (type: string)
      outputColumnNames: _col0, _col1, _col2, _col3, _col4, _col5, _col6
      Statistics: Num rows: 9963904 Data size: 477218560 Basic stats: COMPLETE Column stats: NONE
    Limit
      Number of rows: 1
      Statistics: Num rows: 1 Data size: 47 Basic stats: COMPLETE Column stats: NONE
    ListSink

hive> explain select col0 from passwords limit 1;
OK
STAGE DEPENDENCIES:
  Stage-1 is a root stage
  Stage-0 is a root stage

STAGE PLANS:
  Stage: Stage-1
```

Map Reduce

Map Operator Tree:

TableScan

alias: passwords

Statistics: Num rows: 9963904 Data size: 477218560 Basic stats: COMPLETE Column stats: NONE

Select Operator

expressions: col0 (type: string)

outputColumnNames: _col0

Statistics: Num rows: 9963904 Data size: 477218560 Basic stats: COMPLETE Column stats: NONE

Limit

Number of rows: 1

Statistics: Num rows: 1 Data size: 47 Basic stats: COMPLETE Column stats: NONE

File Output Operator

compressed: false

Statistics: Num rows: 1 Data size: 47 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

- **Value "more" means SELECT, FILTER, LIMIT only (including TABLESAMPLE, virtual columns)."more" can take any kind of expressions in the SELECT clause, including UDFs.(UDTFs and lateral views are not yet supported – see [HIVE-5718](#).)**

"more" is actually to allow "select col from table " to use fetch task per [HIVE-887](#)

initially. See below explain plan:

```
hive> set hive.fetch.task.conversion=more;
hive> explain select col0 from passwords ;
OK
STAGE DEPENDENCIES:
  Stage-0 is a root stage

STAGE PLANS:
  Stage: Stage-0
  Fetch Operator
  limit: -1
  Processor Tree:
  TableScan
  alias: passwords
  Statistics: Num rows: 9963904 Data size: 477218560 Basic stats: COMPLETE Column stats: NONE
  Select Operator
  expressions: col0 (type: string)
  outputColumnNames: _col0
  Statistics: Num rows: 9963904 Data size: 477218560 Basic stats: COMPLETE Column stats: NONE
  ListSink
```

And it also allows "select * " with "filter" to use fetch task, eg:select * from passwords where col0='root';So we have to use "more" very carefully and only use it after fully testing. Because fetch can not utilize the parallelism of MapReduce framework.

2. hive.fetch.task.aggr

Aggregation queries with no group-by clause (for example, select count(*) from src) execute final aggregations in a single reduce task. If this parameter is set to true, Hive delegates the final aggregation stage to a fetch task, possibly decreasing the query time. It was added in Hive 0.12 per [HIVE-4002](#). It can help when the output of the mappers are very small, by reducing the time for shuffle and launching reduce task. For example, compare below differences of explain plan:

```
hive> set hive.fetch.task.aggr;
```

```
hive.fetch.task.aggr=false
hive> explain select count(*) from passwords;
OK
STAGE DEPENDENCIES:
  Stage-1 is a root stage
  Stage-0 is a root stage

STAGE PLANS:
  Stage: Stage-1
    Map Reduce
    Map Operator Tree:
      TableScan
      alias: passwords
      Statistics: Num rows: 9963904 Data size: 477218560 Basic stats: COMPLETE Column stats: COMPLETE
    Select Operator
      Statistics: Num rows: 9963904 Data size: 477218560 Basic stats: COMPLETE Column stats: COMPLETE
    Group By Operator
      aggregations: count()
      mode: hash
      outputColumnNames: _col0
      Statistics: Num rows: 1 Data size: 8 Basic stats: COMPLETE Column stats: COMPLETE
    Reduce Output Operator
      sort order:
      Statistics: Num rows: 1 Data size: 8 Basic stats: COMPLETE Column stats: COMPLETE
      value expressions: _col0 (type: bigint)
Reduce Operator Tree:
      Group By Operator
      aggregations: count(VALUE._col0)
      mode: mergepartial
      outputColumnNames: _col0
      Statistics: Num rows: 1 Data size: 8 Basic stats: COMPLETE Column stats: COMPLETE
    Select Operator
      expressions: _col0 (type: bigint)
      outputColumnNames: _col0
      Statistics: Num rows: 1 Data size: 8 Basic stats: COMPLETE Column stats: COMPLETE
    File Output Operator
      compressed: false
      Statistics: Num rows: 1 Data size: 8 Basic stats: COMPLETE Column stats: COMPLETE
      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
```

```
Time taken: 1.471 seconds, Fetched: 44 row(s)
hive> set hive.fetch.task.aggr=true;
hive> explain select count(*) from passwords;
OK
STAGE DEPENDENCIES:
  Stage-1 is a root stage
  Stage-0 is a root stage
```

```
STAGE PLANS:
  Stage: Stage-1
    Map Reduce
    Map Operator Tree:
      TableScan
```

alias: passwords

Statistics: Num rows: 9963904 Data size: 477218560 Basic stats: COMPLETE Column stats: COMPLETE

Select Operator

Statistics: Num rows: 9963904 Data size: 477218560 Basic stats: COMPLETE Column stats: COMPLETE

Group By Operator

aggregations: count()

mode: hash

outputColumnNames: __col0

Statistics: Num rows: 1 Data size: 8 Basic stats: COMPLETE Column stats: COMPLETE

File Output Operator

compressed: false

table:

input format: org.apache.hadoop.mapred.SequenceFileInputFormat

output format: org.apache.hadoop.hive ql.io.HiveSequenceFileOutputFormat

serde: org.apache.hadoop.hive.serde2.lazybinary.LazyBinarySerDe

Stage: Stage-0

Fetch Operator

limit: -1

Processor Tree:

Group By Operator

aggregations: count(__col0)

mode: mergepartial

outputColumnNames: __col0

Statistics: Num rows: 1 Data size: 8 Basic stats: COMPLETE Column stats: COMPLETE

Select Operator

expressions: __col0 (type: bigint)

outputColumnNames: __col0

Statistics: Num rows: 1 Data size: 8 Basic stats: COMPLETE Column stats: COMPLETE

ListSink

Time taken: 0.148 seconds, Fetched: 40 row(s)

Here are the performance differences:

```
hive> set hive.fetch.task.aggr=false;
hive> select count(*) from passwords;
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_201501081639_0044, Tracking URL = http://n1a.mycluster2.com:50030/jobdetails.jsp?jobid=job_201501081639_0044
Kill Command = /opt/mapr/hadoop/hadoop-0.20.2/bin/./bin/hadoop job -kill job_201501081639_0044
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2015-01-15 12:10:15,706 Stage-1 map = 0%, reduce = 0%
2015-01-15 12:10:26,764 Stage-1 map = 47%, reduce = 0%, Cumulative CPU 6.65 sec
2015-01-15 12:10:27,776 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 8.31 sec
2015-01-15 12:10:32,808 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 9.27 sec
MapReduce Total cumulative CPU time: 9 seconds 270 msec
Ended Job = job_201501081639_0044
MapReduce Jobs Launched:
Job 0: Map: 1 Reduce: 1 Cumulative CPU: 9.27 sec MAPRFS Read: 0 MAPRFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 9 seconds 270 msec
OK
9963904
```

```
Time taken: 19.505 seconds, Fetched: 1 row(s)
hive> set hive.fetch.task.aggr=true;
hive> select count(*) from passwords;
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_201501081639_0045, Tracking URL = http://n1a.mycluster2.com:50030/jobdetails.jsp?jobid=job_201501081639_0045
Kill Command = /opt/mapr/hadoop/hadoop-0.20.2/bin/./bin/hadoop job -kill job_201501081639_0045
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2015-01-15 12:10:47,231 Stage-1 map = 0%, reduce = 0%
2015-01-15 12:10:58,301 Stage-1 map = 47%, reduce = 0%
2015-01-15 12:10:59,308 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 7.26 sec
MapReduce Total cumulative CPU time: 7 seconds 260 msec
Ended Job = job_201501081639_0045
MapReduce Jobs Launched:
Job 0: Map: 1 Cumulative CPU: 7.26 sec MAPRFS Read: 0 MAPRFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 260 msec
OK
9963904
Time taken: 13.817 seconds, Fetched: 1 row(s)
```

Note, if the query has "group-by", it can not use this feature. For example:select count(*) from passwords group by col0;

3. hive.fetch.task.conversion.threshold

This parameter controls input threshold (in bytes) for applying hive.fetch.task.conversion.It was added in Hive 0.13 per HIVE-3990, default is "-1" to disable the feature;The default value was changed in Hive 0.14 to 1GB(1073741824) per HIVE-7397.Take above example, the table size of "passwords" is 465MB:

```
[root@n3a passwords]# pwd
/mapr/my.cluster.com/user/hive/warehouse/passwords
[root@n3a passwords]# du -sh .
465M .
```

If hive.fetch.task.conversion.threshold is less than the table size, it will use MapReduce Job:

```
hive> set hive.fetch.task.conversion.threshold=100000000;
hive> select * from passwords limit 1;
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_201501081639_0046, Tracking URL = http://n1a.mycluster2.com:50030/jobdetails.jsp?jobid=job_201501081639_0046
Kill Command = /opt/mapr/hadoop/hadoop-0.20.2/bin/./bin/hadoop job -kill job_201501081639_0046
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2015-01-15 12:19:06,474 Stage-1 map = 0%, reduce = 0%
2015-01-15 12:19:11,496 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 0.85 sec
MapReduce Total cumulative CPU time: 850 msec
Ended Job = job_201501081639_0046
MapReduce Jobs Launched:
Job 0: Map: 1 Cumulative CPU: 0.85 sec MAPRFS Read: 0 MAPRFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 850 msec
OK
root x 0 0 root /root /bin/bash
Time taken: 6.698 seconds, Fetched: 1 row(s)
```

Else, it will only use fetch task:

```
hive> set hive.fetch.task.conversion.threshold=600000000;
hive> select * from passwords limit 1;
OK
root x 0 0 root /root /bin/bash
Time taken: 0.325 seconds, Fetched: 1 row(s)
```