Interactive Query on HBase

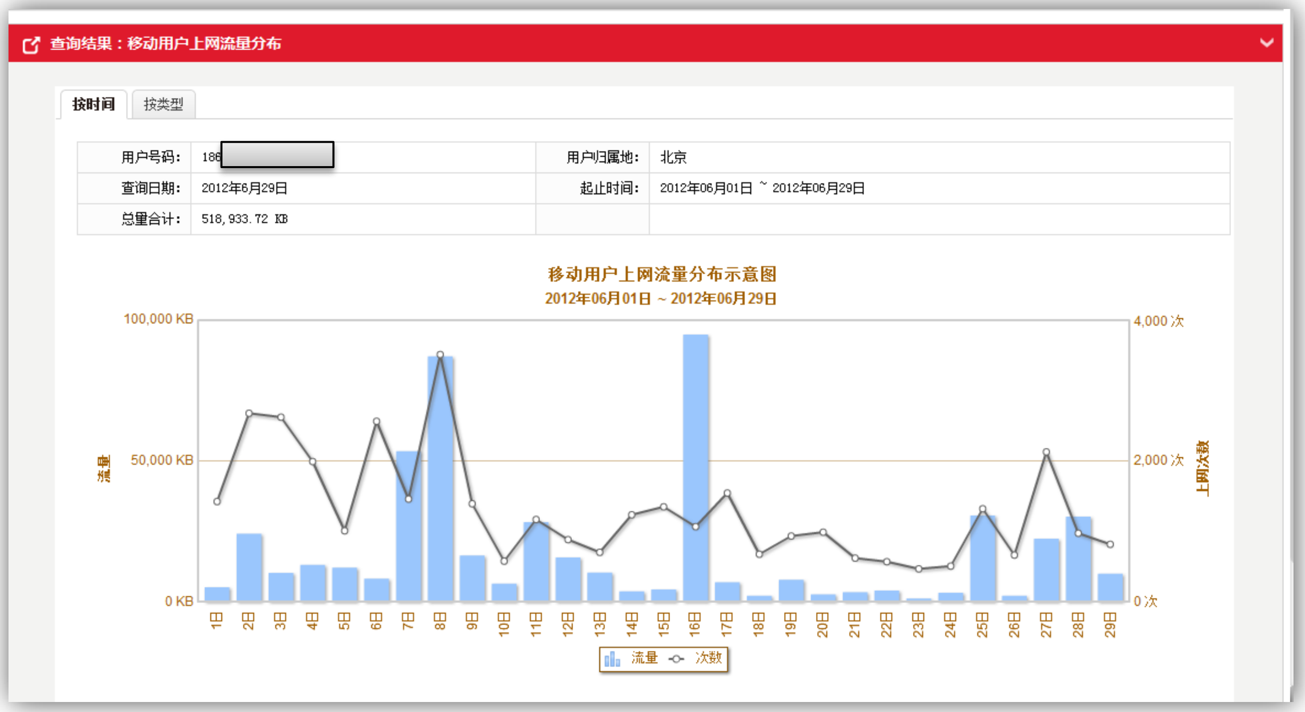
Yuanhao Sun
yuanhao.sun@gmail.com

Agenda

- Why?
- How?
- What features?
- Performance

Why we need interactive query?

- User needs interactive query and instant response, latency typically within 10ms~1s



用户号码:	186[redacted]	用户归属地:	北京
查询日期:	2012年6月29日	起止时间:	2012年06月01日 ˜ 2012年06月29日
总量合计:	518, 933. 72 KB		

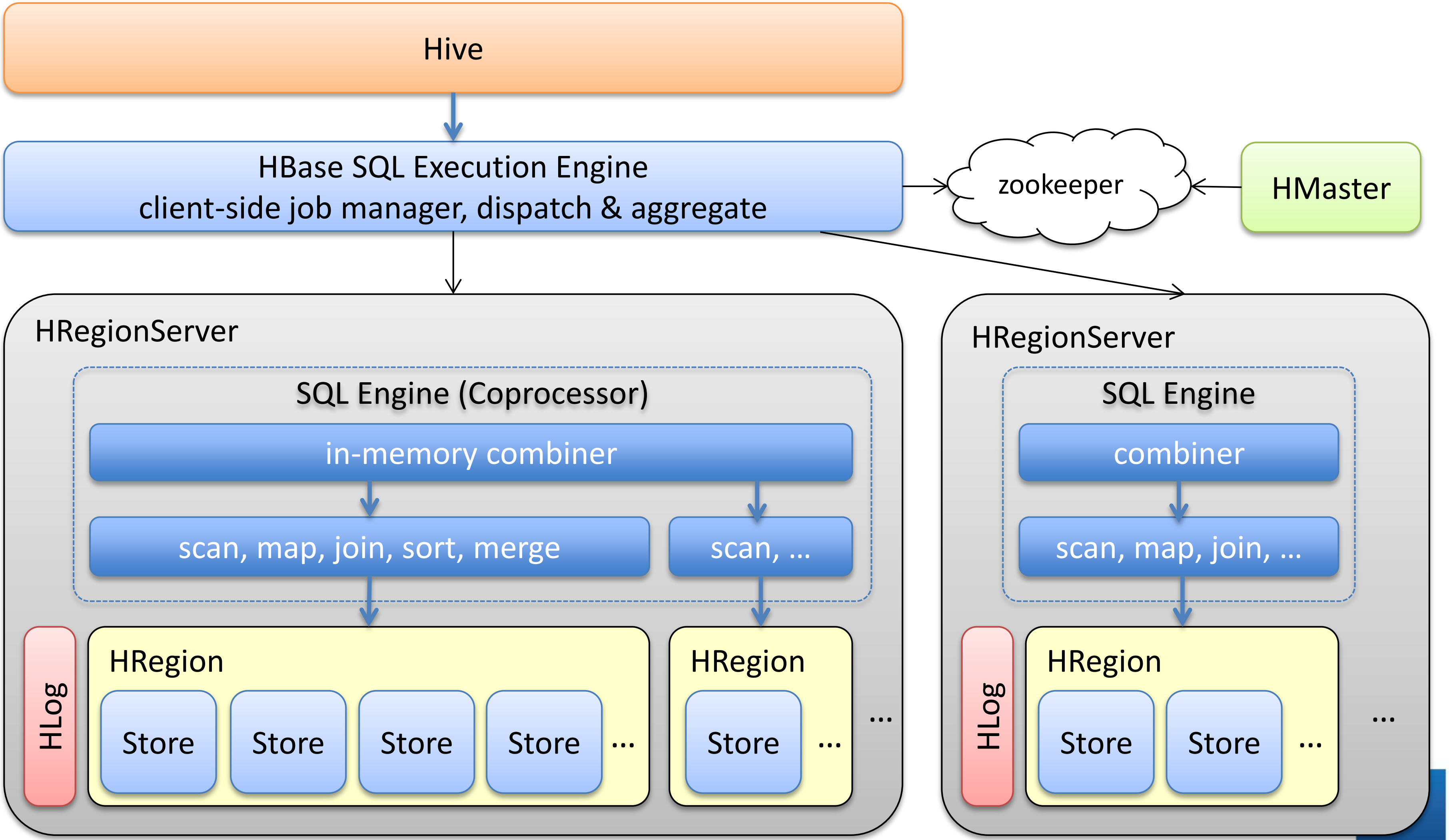
移动用户上网流量分布示意图

2012年06月01日 ~ 2012年06月29日

Why Map/Reduce is unsuitable?

- Map/Reduce is designed for batch analysis
  - M/R over HBase is 3x~4x slower than M/R on HDFS
  - M/R startup overhead is several to tens of seconds
  - On-disk computation, e.g.,
    - read from HDFS every time;
    - write immediate results to disks.
  - Lack of index, not designed for fast lookup

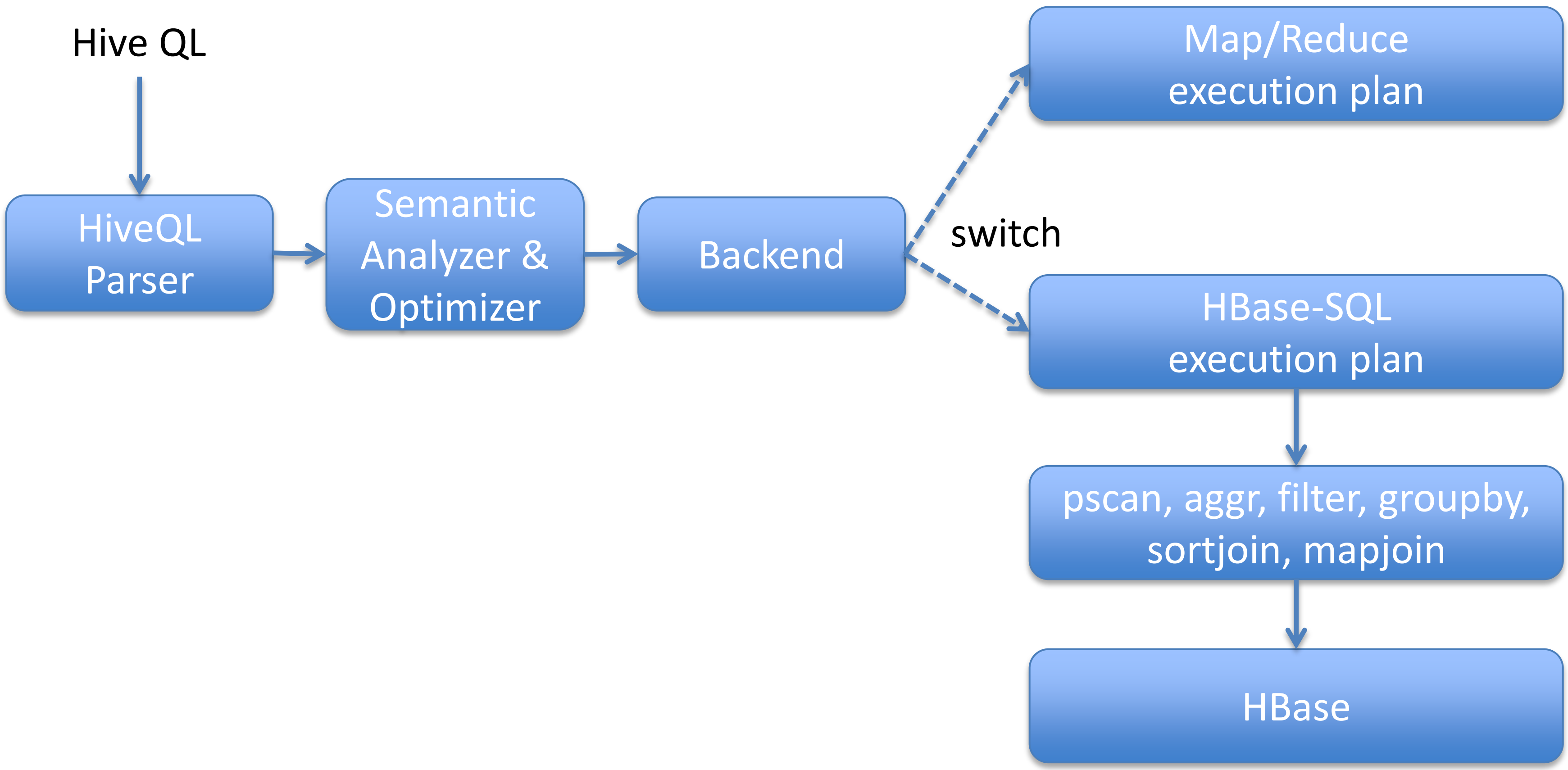
How? Architecture View



SQL Operators on HBase

- pscan
  - parallel scanner, efficient for scanning and filtering multiple regions (at different servers) in parallel
- aggr
  - distributed aggregation function support for HBase tables, ~10x faster than Map/Reduce on HBase for certain queries.
- filter
  - fast/Interactive data scan and filtering, with advanced expressions (any logical combination) for scanner and filters, also includes distributed CRC32 comparator, fuzzy row filter, etc.
- groupby
  - fast group-by aggregation, group-by key is based on any combination of part of row key and columns.
- sortjoin
  - join two large tables, where the join key is part of the row keys of two tables.
- mapjoin
  - join one large table with small table, hash table is built for small table.

Hive-HBase Architecture



Hive QL

HiveQL Parser

Semantic Analyzer & Optimizer

Backend

switch

Map/Reduce execution plan

HBase-SQL execution plan

pscan, aggr, filter, groupby, sortjoin, mapjoin

HBase

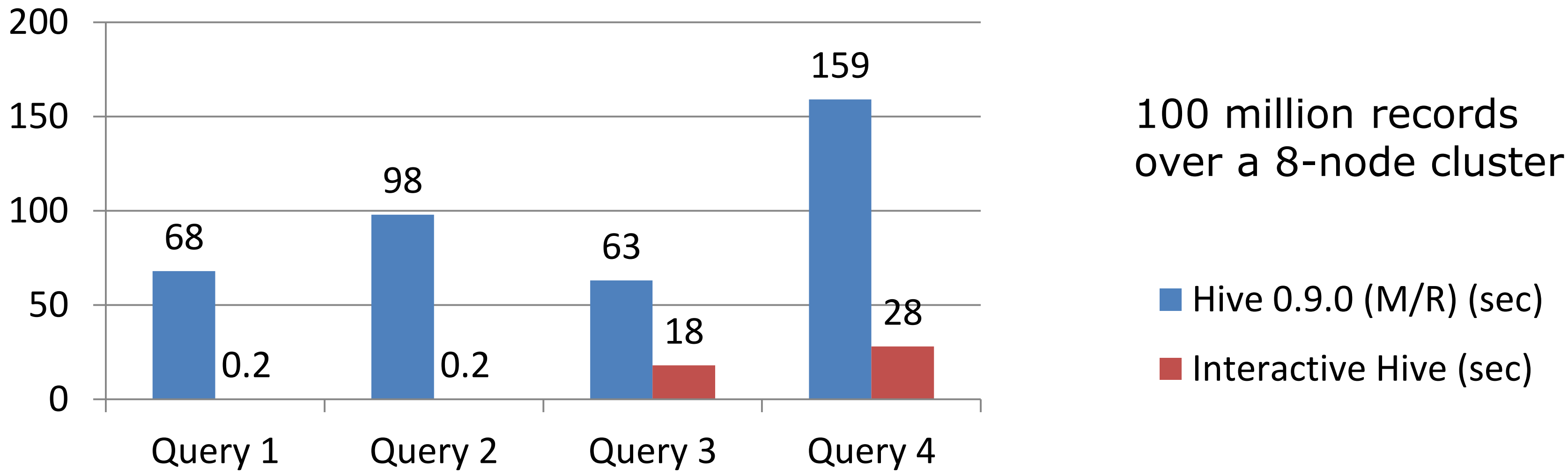
Supported HiveQL

- DDL statements
- SELECT query support:
- WHERE, GROUPBY, HAVING clauses.
- ORDER BY clause: no row key column must using limit keyword.
- JOIN: hash join and sort merge join. Types are left, right, semi, full and outer.
- DISTINCT.
- LIMIT.
- CASE WHEN clause
- LIKE operator,
- CAST clause.
- Data type: boolean, tinyint, smallint, int, bigint, float, double, string, struct.
- Aggregate function: count, max, min, sum, avg.
- Relational operators: >, >=, <=, <, =.
- Arithmetic operators: +, - , * , / , %.
- Logical operators: and, or, not (row key column does not support not).
- String function: substring, concat.

Limitations

- No support for joining multiple large tables yet
- No sub-query support yet

Interactive Hive Query over HBase



User Scenario	Query
Calculate each day's internet traffic of a specific user	SELECT sum(down+up) FROM cdr201209 WHERE number = '13300000000' GROUP BY day;
Get the 10 most heavily called numbers for a specific user	SELECT TOP(10) tonumber, sum(call_length) len FROM cdr_201209 WHERE number = '13300032810' GROUP BY tonumber ORDER BY len DESC
Get the top 1000 call length from all user phone calls	SELECT TOP(1000) number, call_length FROM cdr_201209 ORDER BY call_length DESC
Get the top 1000 users having highest total monthly charge	SELECT TOP(1000) number, sum(fee) f FROM cdr_201209 GROUP BY number order by f DESC

Join Performance

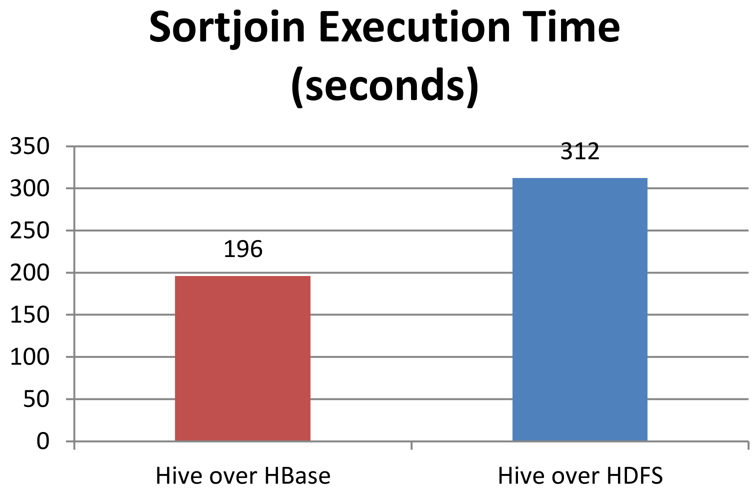
select count(*) from sm_left join sm_right on sm_left.id = sm_right.id

- Sortjoin's performance

  left table 100m records,

  right table 100m records,

  join result 1m records

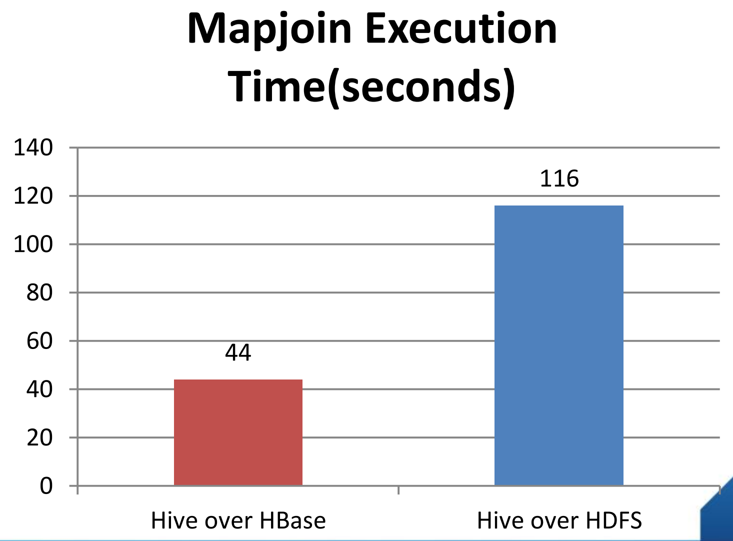


- Map-join's performance

  left table 100m records,

  right table 1m records,

  join result 1m records

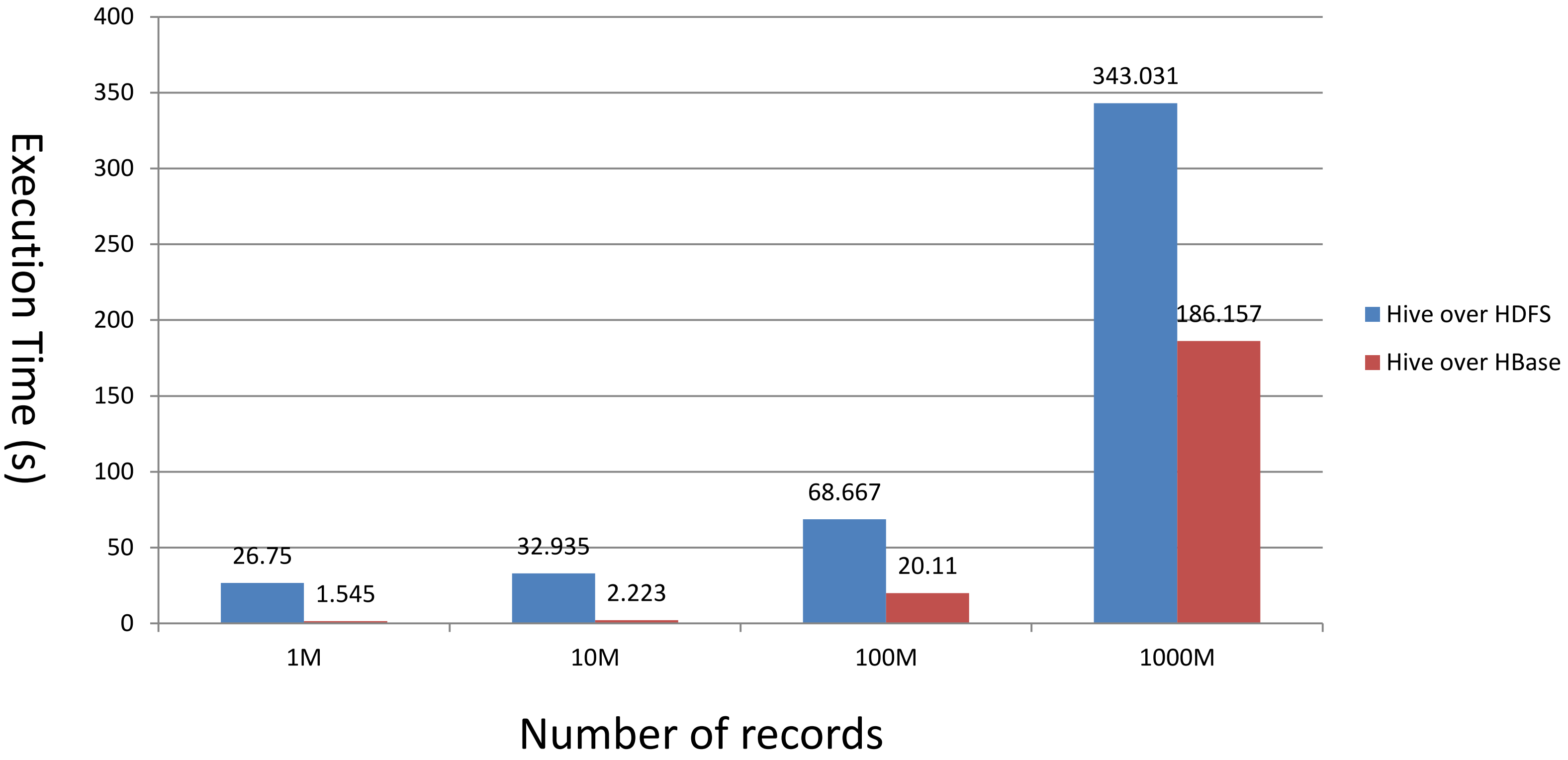


3 severs: 2 CPU x Intel® Xeon® CPU E5-2680 @ 2.70GHz Memory = 64GB Hard disk = 8 x2T

Full Table Scan

select city, sum(fee) from table group by city;

6 severs: 2 CPU x Intel® Xeon® CPU E5-2680 @ 2.70GHz Memory = 64GB Hard disk = 8 x2T



More info

- Check hadoop.intel.com or www.intel.cn/idh
- Try Intel's Distribution for Apache Hadoop*