facebook

Facebook's Petabyte Scale Data Warehouse using Hive and Hadoop

QCon San Francisco Nov 2009

facebook

Why Another Data Warehousing System?

Data, data and more data

200GB per day in March 2008

12+TB(compressed) raw data per day today

Trends Leading to More Data

Free or low cost of user services

Realization that more insights are derived from simple algorithms on more data

Deficiencies of Existing Technologies

Cost of Analysis and Storage on proprietary systems does not support trends towards more data

Limited Scalability does not support trends towards more data

Closed and Proprietary Systems

Lets try Hadoop…

- Pros
  - Superior in availability/scalability/manageability
  - Efficiency not that great, but throw more hardware
  - Partial Availability/resilience/scale more important than ACID
- Cons: Programmability and Metadata
  - Map-reduce hard to program (users know sql/bash/python)
  - Need to publish data in well known schemas
- Solution: HIVE

What is HIVE?

- A system for managing and querying structured data built on top of Hadoop
  - Map-Reduce for execution
  - HDFS for storage
  - Metadata on hdfs files

- Key Building Principles:
  - SQL as a familiar data warehousing tool
  - Extensibility – Types, Functions, Formats, Scripts
  - Scalability and Performance

facebook

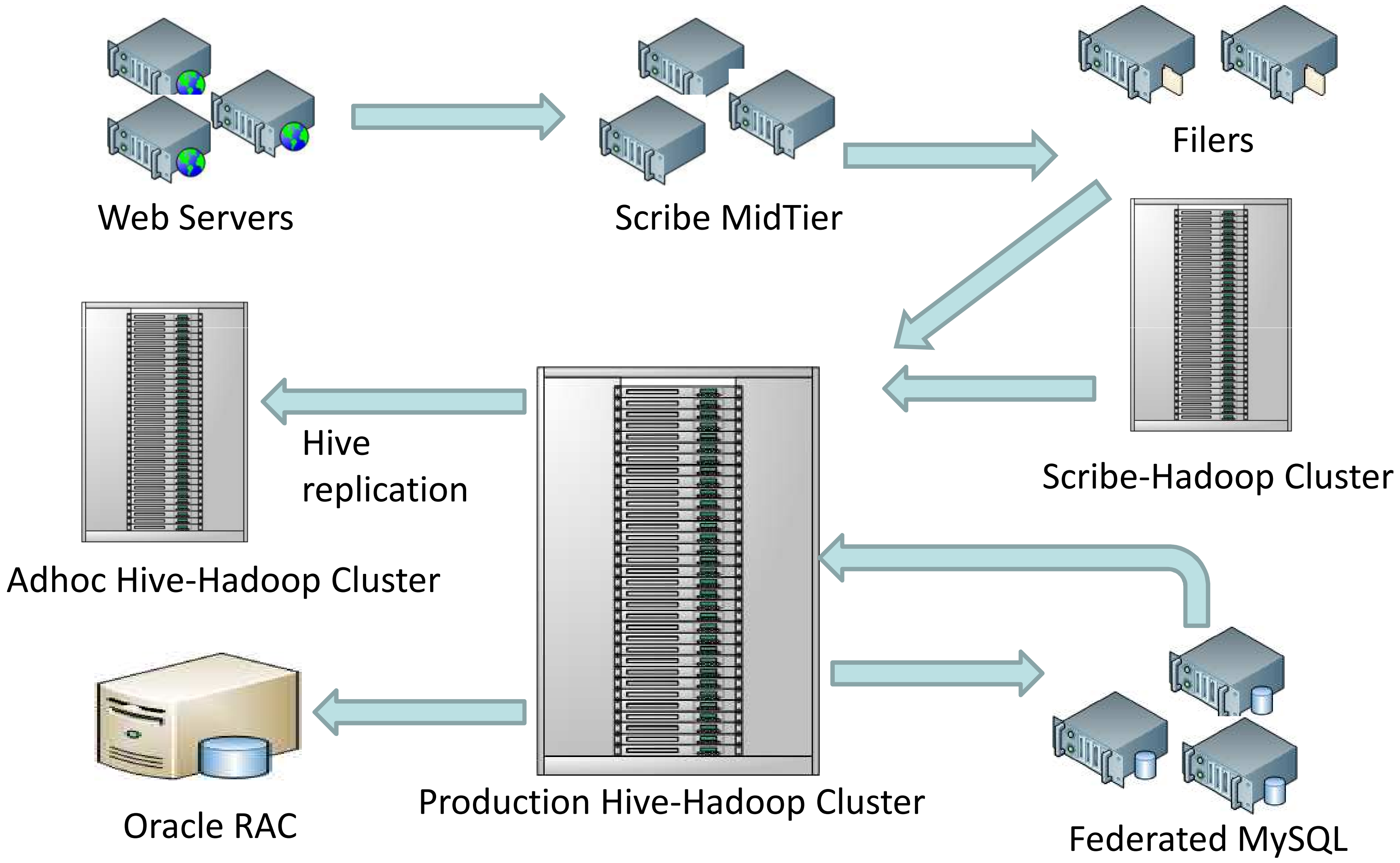
Why SQL on Hadoop?

```
hive> select key, count(1) from kv1 where key > 100 group by
   key;
```

vs.

```
$ cat > /tmp/reducer.sh
uniq -c | awk '{print $2"\t"$1}‘
$ cat > /tmp/map.sh
awk -F '\001' '{if($1 > 100) print $1}‘
$ bin/hadoop jar contrib/hadoop-0.19.2-dev-streaming.jar -input /user/hive/warehouse/kv1 -
   mapper map.sh -file /tmp/reducer.sh -file /tmp/map.sh -reducer reducer.sh -output
   /tmp/largekey -numReduceTasks 1
$ bin/hadoop dfs –cat /tmp/largekey/part*
```

Data Flow Architecture at Facebook



facebook

Hadoop & Hive Cluster @ Facebook

- **Hadoop/Hive Warehouse – the new generation**
  - 5800 cores, Raw Storage capacity of 8.7 PetaBytes
  - 12 TB per node
  - Two level network topology
    - 1 Gbit/sec from node to rack switch
    - 4 Gbit/sec to top level rack switch

Hive & Hadoop Usage @ Facebook

- **Statistics per day:**
  - 12 TB of compressed new data added per day
  - 135TB of compressed data scanned per day
  - 7500+ Hive jobs per day
  - 80K compute hours per day

- **Hive simplifies Hadoop:**
  - New engineers go though a Hive training session
  - ~200 people/month run jobs on Hadoop/Hive
  - Analysts (non-engineers) use Hadoop through Hive
  - 95% of jobs are Hive Jobs

facebook

Hive & Hadoop Usage @ Facebook

- Types of Applications:
  - Reporting
    - Eg: Daily/Weekly aggregations of impression/click counts
    - Measures of user engagement
    - Microstrategy dashboards
  - Ad hoc Analysis
    - Eg: how many group admins broken down by state/country
  - Machine Learning (Assembling training data)
    - Ad Optimization
    - Eg: User Engagement as a function of user attributes
  - Many others

facebook

Challenges

- Space Constraints
  - RAID
  - Columnar Compression
- Resource Scheduling
  - Fair Share Scheduler
- Job Isolation

facebook

More about HIVE

facebook

Data Model

	Name	**HDFS Directory**
Table	pvs	/wh/pvs
Partition	ds = 20090801, ctry = US	/wh/pvs/ds=20090801/ctry=US
Bucket	user into 32 buckets HDFS file for user hash 0	/wh/pvs/ds=20090801/ctry=US/part-00000

facebook

Data Model

- External Tables
  - Point to existing data directories in HDFS
  - Can create tables and partitions - partition columns just become annotations to external directories
  - Example: create external table with partitions

```
CREATE EXTERNAL TABLE pvs(uhash int, pageid int,
                          ds string, ctry string)
PARTITIONED ON (ds string, ctry string)
STORED AS textfile
LOCATION '/path/to/existing/table'
```

  - Example: add a partition to external table

```
ALTER TABLE pvs
ADD PARTITION (ds='20090801', ctry='US')
LOCATION '/path/to/existing/partition'
```

Hive Query Language

- SQL
  - Sub-queries in from clause
  - Equi-joins (including Outer joins)
  - Multi-table Insert
  - Multi-group-by
  - Embedding Custom Map/Reduce in SQL
- Sampling
- Primitive Types
  - integer types, float, string, date, boolean
- Nestable Collections
  - array<any-type> and map<primitive-type, any-type>
- User-defined types
  - Structures with attributes which can be of any-type

facebook

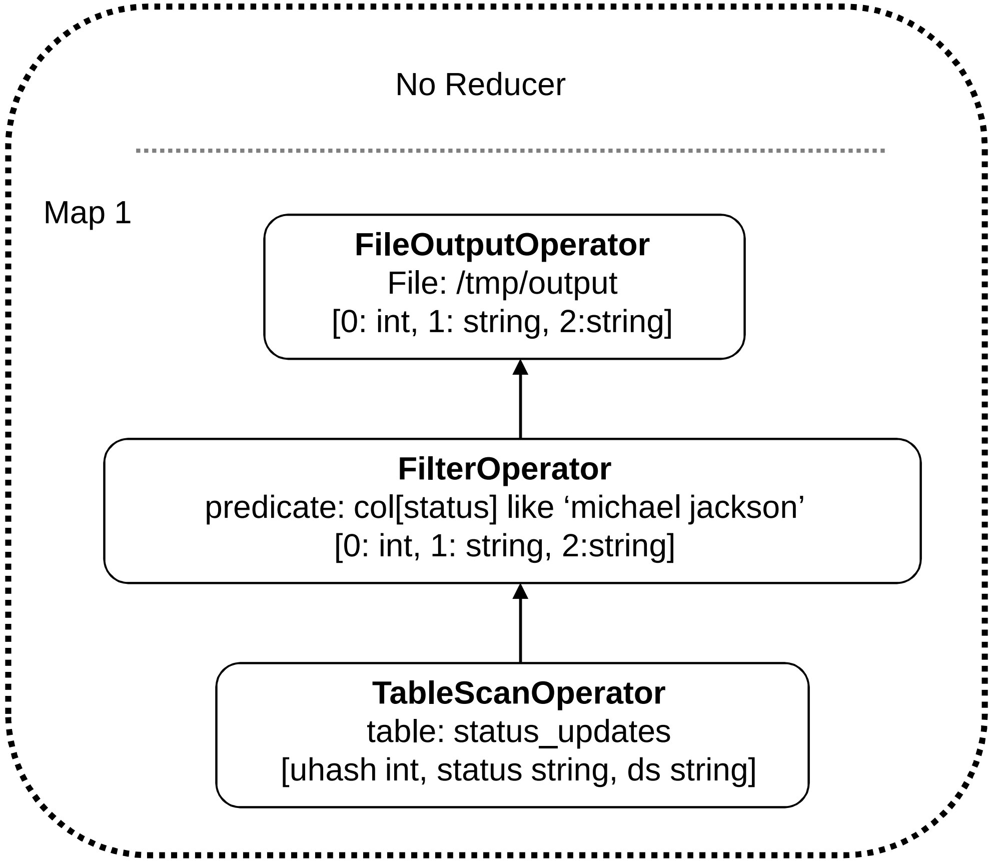
Example Application

- Status updates table:
  - `status_updates(uhash int, status string, ds string)`
- Load the data from log files:
  - `LOAD DATA LOCAL INPATH ‘/logs/status_updates’ INTO TABLE status_updates PARTITION (ds=’2009-03-20’)`
- User profile table
  - `bkt_profile(uhash int, age_bkt string, gender int)`

facebook

Example Query (Filter)

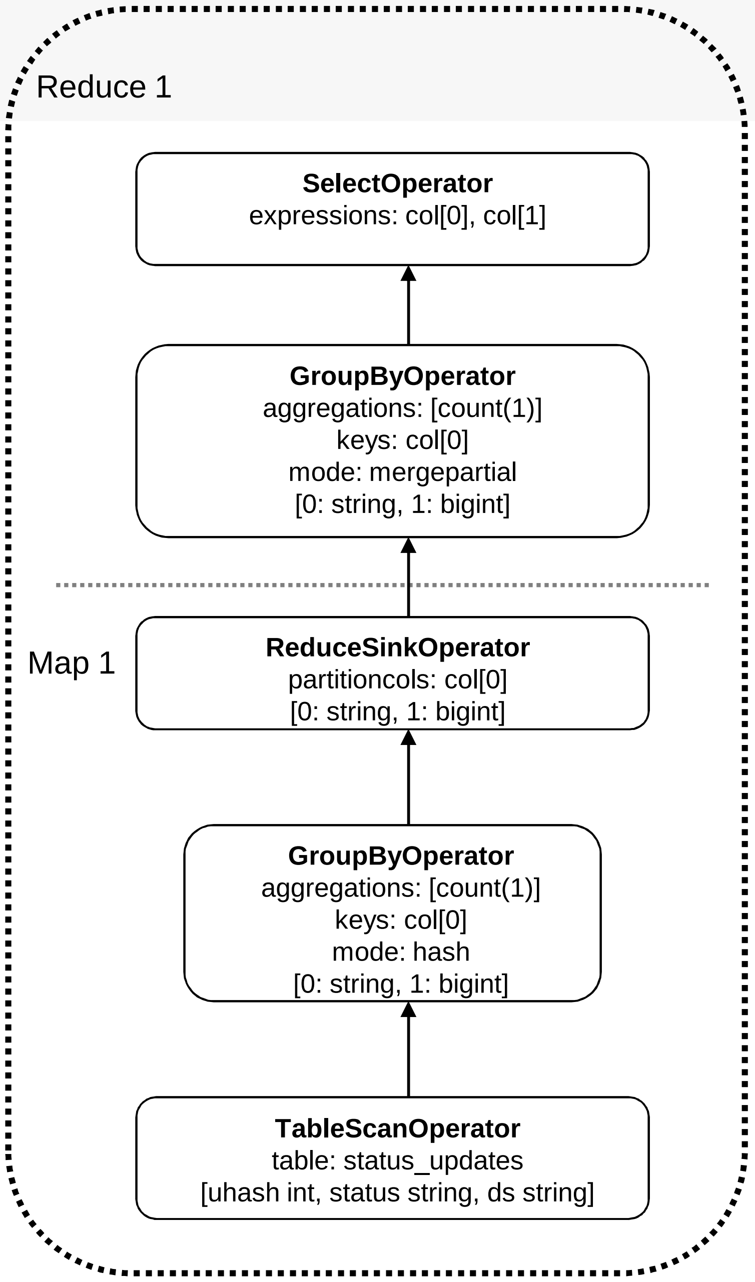
- Filter status updates containing ‘michael jackson’
  - `SELECT * FROM status_updates WHERE status LIKE ‘michael jackson’`


No Reducer
Map 1
FileOutputOperator
File: /tmp/output
[0: int, 1: string, 2:string]
FilterOperator
predicate: col[status] like ‘michael jackson’
[0: int, 1: string, 2:string]
TableScanOperator
table: status_updates
[uhash int, status string, ds string]

Example Query (Aggregation)

- Figure out total number of status_updates in a given day

  -
    ```
    SELECT ds, COUNT(1)
    FROM status_updates
    GROUP BY ds
    ```



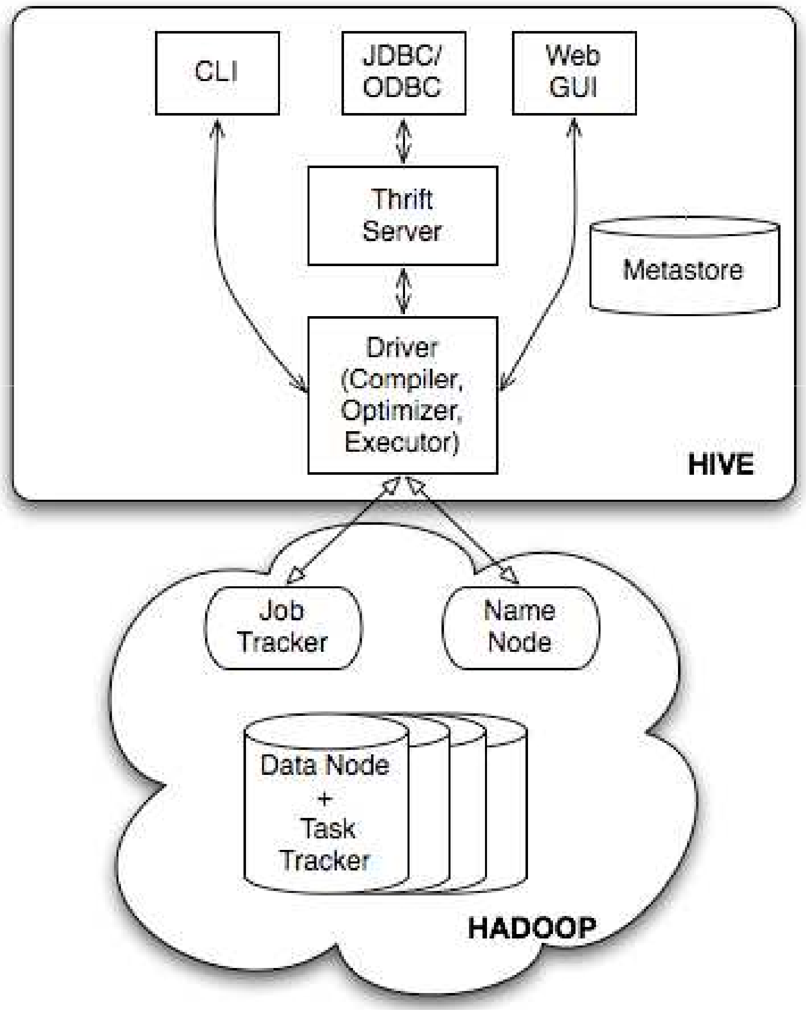
facebook

Example Query (multi-group-by)

```
FROM (SELECT a.status, b.age_bkt, b.gender
      FROM status_updates a JOIN profiles b
           ON (a.uhash = b.uhash and
               a.ds='2009-03-20' )
      ) subq1
INSERT OVERWRITE TABLE gender_summary
                      PARTITION(ds='2009-03-20')
SELECT subq1.gender, COUNT(1)
GROUP BY subq1.gender
INSERT OVERWRITE TABLE age_bkt_summary
  PARTITION(ds='2009-03-20')
SELECT subq1.age_bkt, COUNT(1)
GROUP BY subq1.age_bkt
```

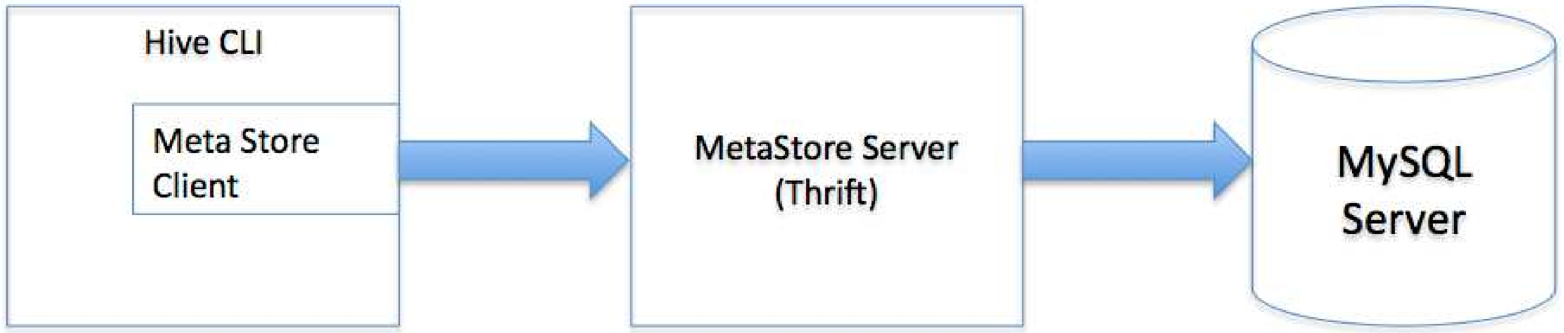
facebook

Hive Architecture



facebook

Hive Metastore



- Server Configuration same as multi user mode client config (prev slide). To run server

```
$JAVA_HOME/bin/java -Xmx1024m -Dlog4j.configuration=file://$HIVE_HOME/conf/hms-log4j.properties
    -Djava.library.path=$HADOOP_HOME/lib/native/Linux-amd64-64/ -cp $CLASSPATH
    org.apache.hadoop.hive.metastore.HiveMetaStore
```

- Client Configuration

Parameter	Description	Example
hive.metastore.uris	Location of the metastore server	thrift://<host_name>:9083
hive.metastore.local		false

facebook

Hive Optimizations

Optimizations

- Column Pruning
- Predicate Pushdown
- Partition Pruning
- Join
- Group By
- Small files

facebook

Column Pruning

- As name suggests – discard columns which are not needed
  - `SELECT a,b FROM T WHERE e < 10;`
  - `T contains 5 columns (a,b,c,d,e)`
- Columns c,d are discarded
- Select only the relevant columns

Predicate Pushdown

- Move predicate closer to the table scan.
- Predicates moved up across joins.
  - `SELECT * FROM T1 JOIN T2 ON (T1.c1=T2.c2 AND T1.c1 < 10)`
  - `SELECT * FROM T1 JOIN T2 ON (T1.c1=T2.c2) WHERE T1.c1 < 10`
- Special needs for outer joins:
  - Left outer join: predicates on the left side aliases are pushed
  - Right outer join: predicates on the right side aliases are pushed
  - Full outer join: none of the predicates are pushed
- Non-deterministic functions (eg. rand()) not pushed.
- Use annotation:
  - `@UDFType(deterministic=false)`

facebook

Hive QL - Join

```
INSERT OVERWRITE TABLE pv_users
SELECT pv.pageid, u.age_bkt
FROM page_view pv
  JOIN user u
  ON (pv.uhash = u.uhash);
```

Hive QL – Join in Map Reduce

page_view

pageid	**uhash**	time
1	**111**	9:08:01
2	**111**	9:08:13
1	**222**	9:08:14

user

uhash	age _bkt	gender
111	B3	female
222	B4	male

Map

key	value
111	<**1**,1>
111	<**1**,2>
222	<**1**,1>

key	value
111	<**2**,B3>
222	<**2**,B4>

Shuffle Sort

key	value
111	<**1**,1>
111	<**1**,2>
111	<**2**,B3>

key	value
222	<**1**,1>
222	<**2**,B4>

Reduce

facebook

Hive QL - Join

```
INSERT OVERWRITE TABLE pv_users
SELECT pv.pageid, u.age_bkt
FROM page_view p JOIN user u
  ON (pv.uhash = u.uhash)
  JOIN newuser x on (u.uhash = x.uhash);
```

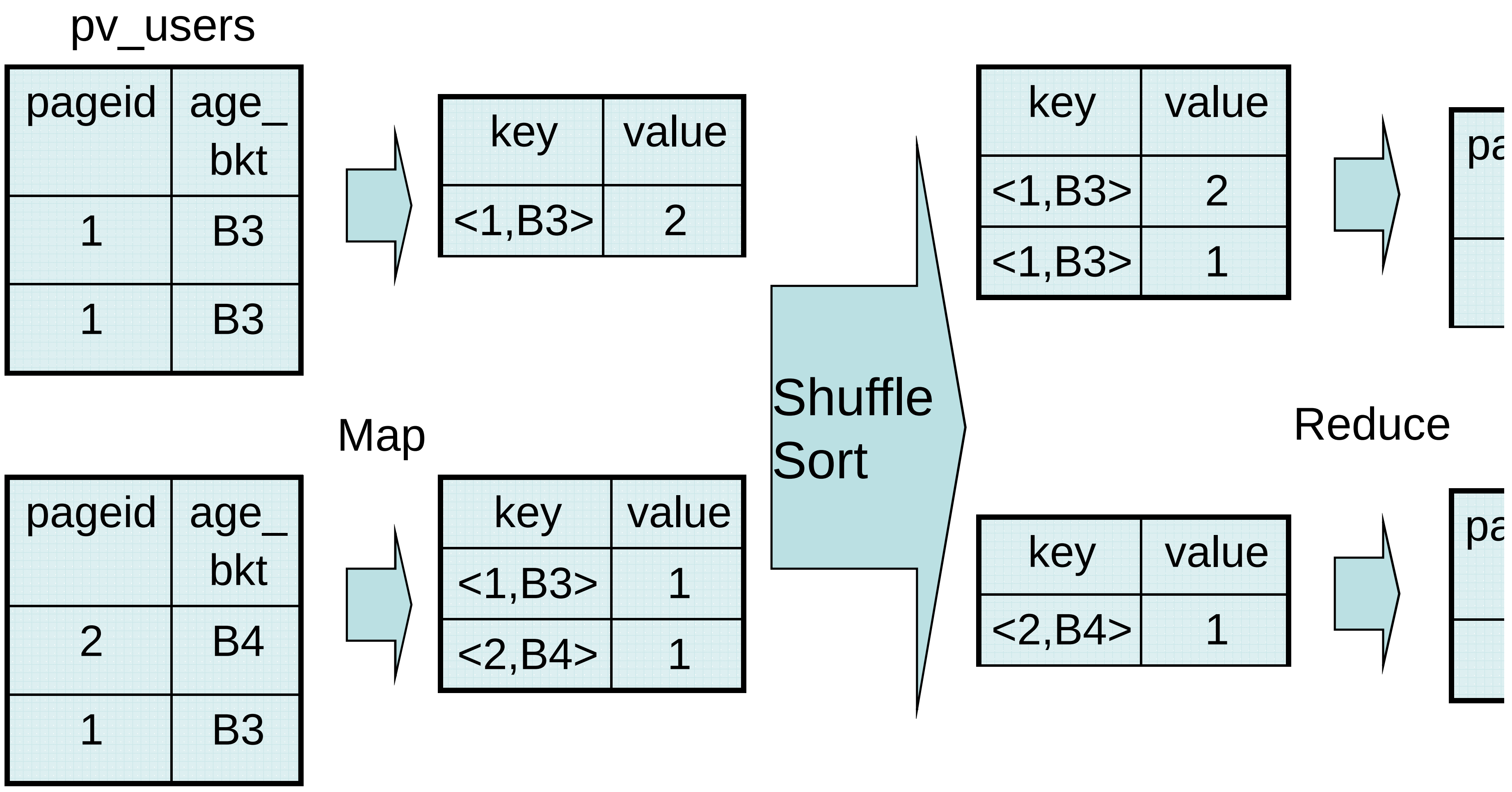
- Same join key – merge into 1 map-reduce job (OUTER joins also)
- 1 map-reduce job instead of ‘n’
- Rightmost table streamed.
- Hint for specifying the largest table
- Map Joins
  – User specified small tables stored in hash tables on the mapper
  – No reducer needed

facebook

Hive QL - Group By

```
SELECT pageid, age_bkt, count(1)
FROM pv_users
GROUP BY pageid, age_bkt;
```

Hive QL – Group By in Map Reduce



pageid	age_bkt
1	B3
1	B3

pageid	age_bkt
2	B4
1	B3

key	value
<1,B3>	2

key	value
<1,B3>	1
<2,B4>	1

key	value
<1,B3>	2
<1,B3>	1

key	value
<2,B4>	1

Group by Optimizations

- Map side partial aggregations
  - Hash-based aggregates
  - Serialized key/values in hash tables
  - 90% speed improvement on Query
    - `SELECT count(1) FROM t;`
- Load balancing for data skew

Multi GroupBy

```
FROM pv_users
  INSERT OVERWRITE TABLE pv_gender_sum
    SELECT gender, count(DISTINCT uhash), count(uhash)
      GROUP BY gender
  INSERT OVERWRITE TABLE pv_age_sum
    SELECT age_bkt, count(DISTINCT uhash)
      GROUP BY age_bkt
```

- n+1 map-reduce jobs instead of 2n
- Single scan of input table
- Same distinct key across all groupbys
- Always use multi-groupby

facebook

Hive Extensibility Features

Hive is an open system

- Different on-disk data formats
  - Text File, Sequence File, …
- Different in-memory data formats
  - Java Integer/String, Hadoop IntWritable/Text …
- User-provided map/reduce scripts
  - In any language, use stdin/stdout to transfer data …
- User-defined Functions
  - Substr, Trim, From_unixtime …
- User-defined Aggregation Functions
  - Sum, Average …
- User-define Table Functions
  - Explode …

facebook

File Format Example

-
  ```
  CREATE TABLE mylog (
    uhash BIGINT,
    page_url STRING,
    unix_time INT)
  STORED AS TEXTFILE;
  ```

-
  ```
  LOAD DATA INPATH '/user/myname/log.txt' INTO
  TABLE mylog;
  ```

facebook

Existing File Formats

	TEXTFILE	SEQUENCEFILE	RCFILE
Data type	text only	text/binary	text/binary
Internal Storage order	Row-based	Row-based	Column-based
Compression	File-based	Block-based	Block-based
Splitable*	YES	YES	YES
Splitable* after compression	NO	YES	YES

*** Splitable: Capable of splitting the file so that a single huge file can be processed by multiple mappers in parallel.**

facebook

SerDe

- SerDe is short for serialization/deserialization. It controls the format of a row.
- Serialized format:
  - Delimited format (tab, comma, ctrl-a …)
  - Thrift Protocols
- Deserialized (in-memory) format:
  - Java Integer/String/ArrayList/HashMap
  - Hadoop Writable classes
  - User-defined Java Classes (Thrift)

facebook

SerDe Examples

-
  ```
  CREATE TABLE mylog (
    uhash    BIGINT,
    page_url  STRING,
    unix_time INT)
  ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';
  ```

-
  ```
  CREATE table mylog_rc (
    uhash    BIGINT,
    page_url  STRING,
    unix_time INT)
  ROW FORMAT SERDE
    'org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe'
  STORED AS RCFILE;
  ```

facebook

Existing SerDes

	LazySimpleSerDe	**LazyBinarySerDe (HIVE-640)**	**BinarySortable SerDe**
serialized format	delimited	proprietary binary	proprietary binary sortable*
deserialized format	LazyObjects*	LazyBinaryObjects*	Writable
	ThriftSerDe (HIVE-706)	**RegexSerDe**	**ColumnarSerDe**
serialized format	Depends on the Thrift Protocol	Regex formatted	proprietary column-based
deserialized format	User-defined Classes, Java Primitive Objects	ArrayList<String>	LazyObjects*

*** LazyObjects: deserialize the columns only when accessed.**

*** Binary Sortable: binary format preserving the sort order.**

Map/Reduce Scripts Examples

- add file page_url_to_id.py;
- add file my_python_session_cutter.py;
- FROM

```
   (SELECT TRANSFORM(uhash, page_url, unix_time)
       USING 'page_url_to_id.py'
       AS (uhash, page_id, unix_time)
    FROM mylog
    DISTRIBUTE BY uhash
    SORT BY uhash, unix_time) mylog2
SELECT TRANSFORM(uhash, page_id, unix_time)
   USING 'my_python_session_cutter.py'
   AS (uhash, session_info);
```

facebook

UDF Example

- `add jar build/ql/test/test-udfs.jar;`
- `CREATE TEMPORARY FUNCTION testlength AS 'org.apache.hadoop.hive.ql.udf.UDFTestLength';`
- `SELECT testlength(page_url) FROM mylog;`
- `DROP TEMPORARY FUNCTION testlength;`

- UDFTestLength.java:

```
package org.apache.hadoop.hive.ql.udf;
public class UDFTestLength extends UDF {
  public Integer evaluate(String s) {
    if (s == null) {
      return null;
    }
    return s.length();
  }
}
```

UDAF Example

-
  ```
  SELECT page_url, count(1)
  FROM mylog;
  ```

-
  ```
  public class UDAFCount extends UDAF {
    public static class Evaluator implements UDAFEvaluator {
     private int mCount;
    public void init() {mcount = 0;}
    public boolean iterate(Object o) {
      if (o!=null) mCount++; return true;}
    public Integer terminatePartial() {return mCount;}
    public boolean merge(Integer o) {mCount += o; return
  true;}
    public Integer terminate() {return mCount;}
  }
  ```

Comparison of UDF/UDAF v.s. M/R scripts

	UDF/UDAF	M/R scripts
language	Java	any language
data format	in-memory objects	serialized streams
1/1 input/output	supported via UDF	supported
n/1 input/output	supported via UDAF	supported
1/n input/output	supported via UDTF	supported
Speed	faster	Slower

facebook

Powered by Hive

facebook

cnet

digg

Chitika
Turning page views into profits

bizo

Grooveshark

hi5

HubSpot

last.fm

Open Source Community

- Release Hive-0.4 on 10/13/2009
- 50 contributors and growing
- 11 committers
  - 3 external to Facebook
- Available as a sub project in Hadoop
  - http://wiki.apache.org/hadoop/Hive (wiki)
  - http://hadoop.apache.org/hive (home page)
  - http://svn.apache.org/repos/asf/hadoop/hive (SVN repo)
  - ##hive (IRC)
  - Works with hadoop-0.17, 0.18, 0.19, 0.20
- Mailing Lists:
  - hive-{user,dev,commits}@hadoop.apache.org

facebook

Future

- Create table as select
- Inserts without listing partitions
- Use sort properties to optimize query
- IN, exists and correlated sub-queries

- Statistics
- More join optimizations
- Persistent UDFs and UDAFs
- Better techniques for handling skews for a given key