facebook

HIVE

Data Warehousing & Analytics on Hadoop

Facebook Data Team

Why Another Data Warehousing System?

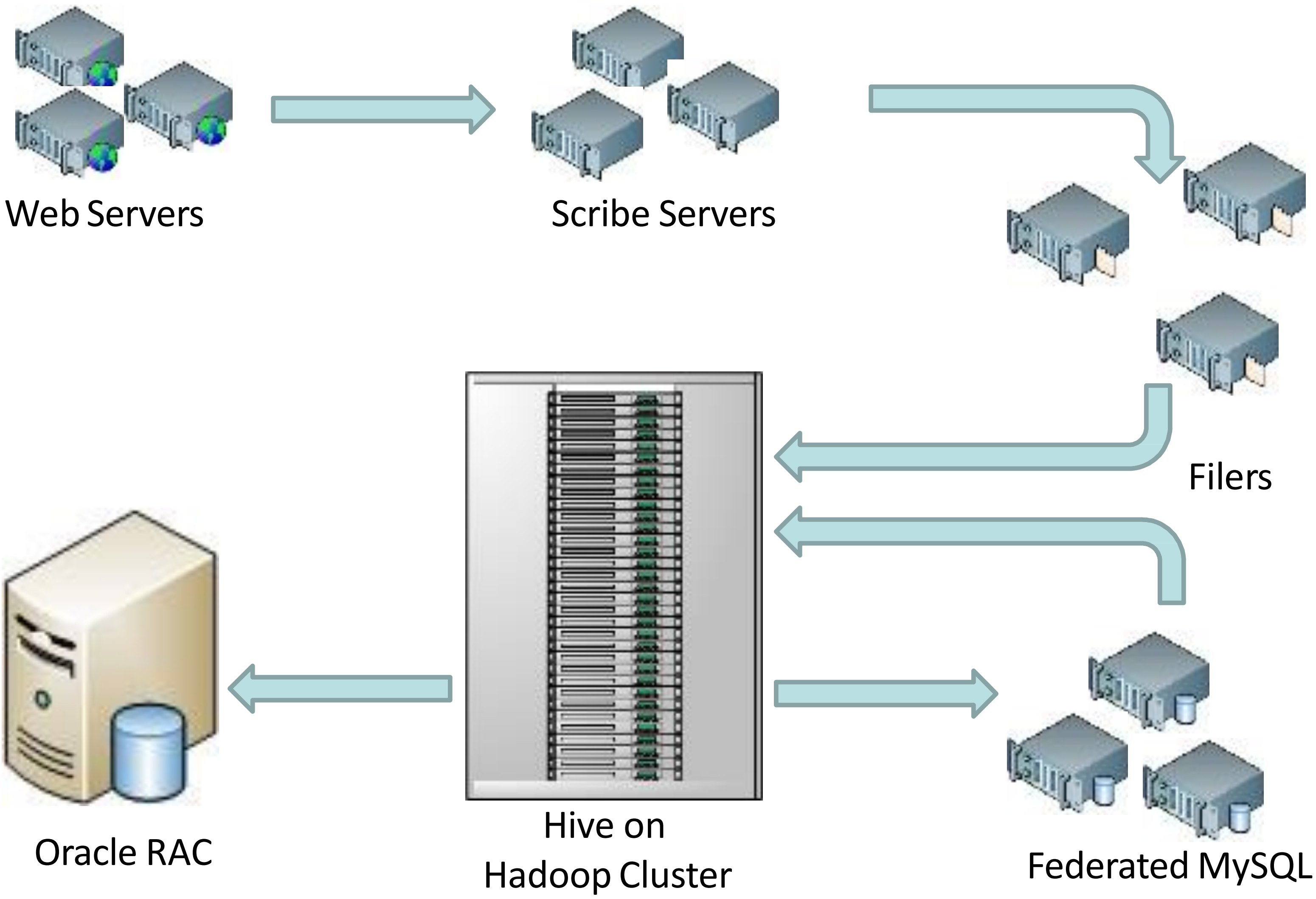
- Problem: Data, data and more data
  - 200GB per day in March 2008
  - 2+TB(compressed) raw data per day today
- The Hadoop Experiment
  - Much superior to availability and scalability of commercial DBs
  - Efficiency not that great, but throw more hardware
  - Partial Availability/resilience/scale more important than ACID
- Problem: 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 querying and managing structured data built on top of Hadoop
  - Uses Map-Reduce for execution
  - HDFS for storage – but any system that implements Hadoop FS API
- Key Building Principles:
  - Structured data with rich data types (structs, lists and maps)
  - Directly query data from different formats (text/binary) and file formats (Flat/Sequence)
  - SQL as a familiar programming tool and for standard analytics
  - Allow embedded scripts for extensibility and for non standard applications
  - Rich MetaData to allow data discovery and for optimization

facebook

Data Warehousing at Facebook Today



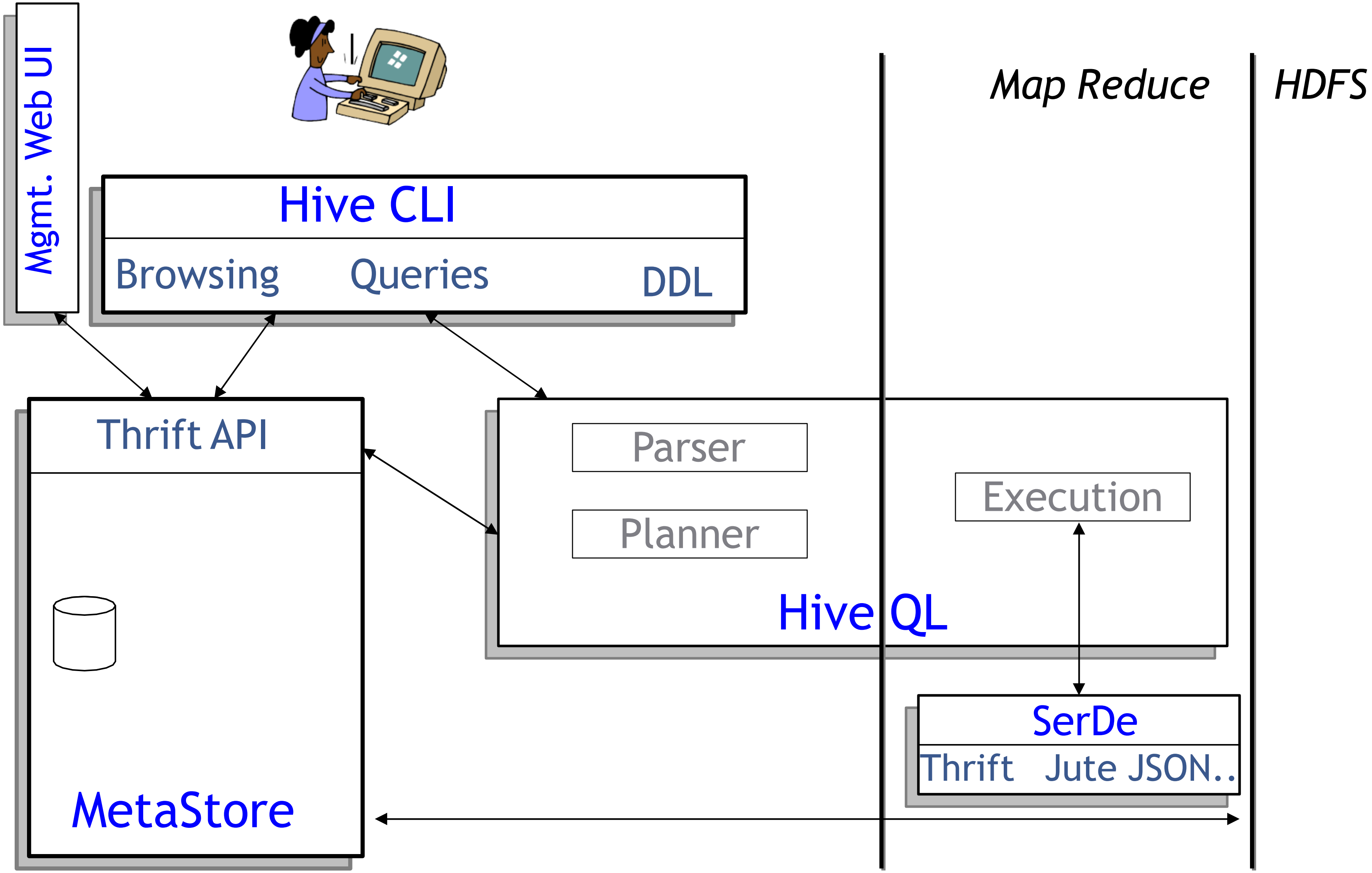
Hive/Hadoop Usage @ Facebook

- Types of Applications:
  - Summarization
    - Eg: Daily/Weekly aggregations of impression/click counts
    - Complex measures of user engagement
  - Ad hoc Analysis
    - Eg: how many group admins broken down by state/country
  - Data Mining (Assembling training data)
    - Eg: User Engagement as a function of user attributes
  - Spam Detection
    - Anomalous patterns in UGC
    - Application api usage patterns
  - Ad Optimization
  - Too many to count ..

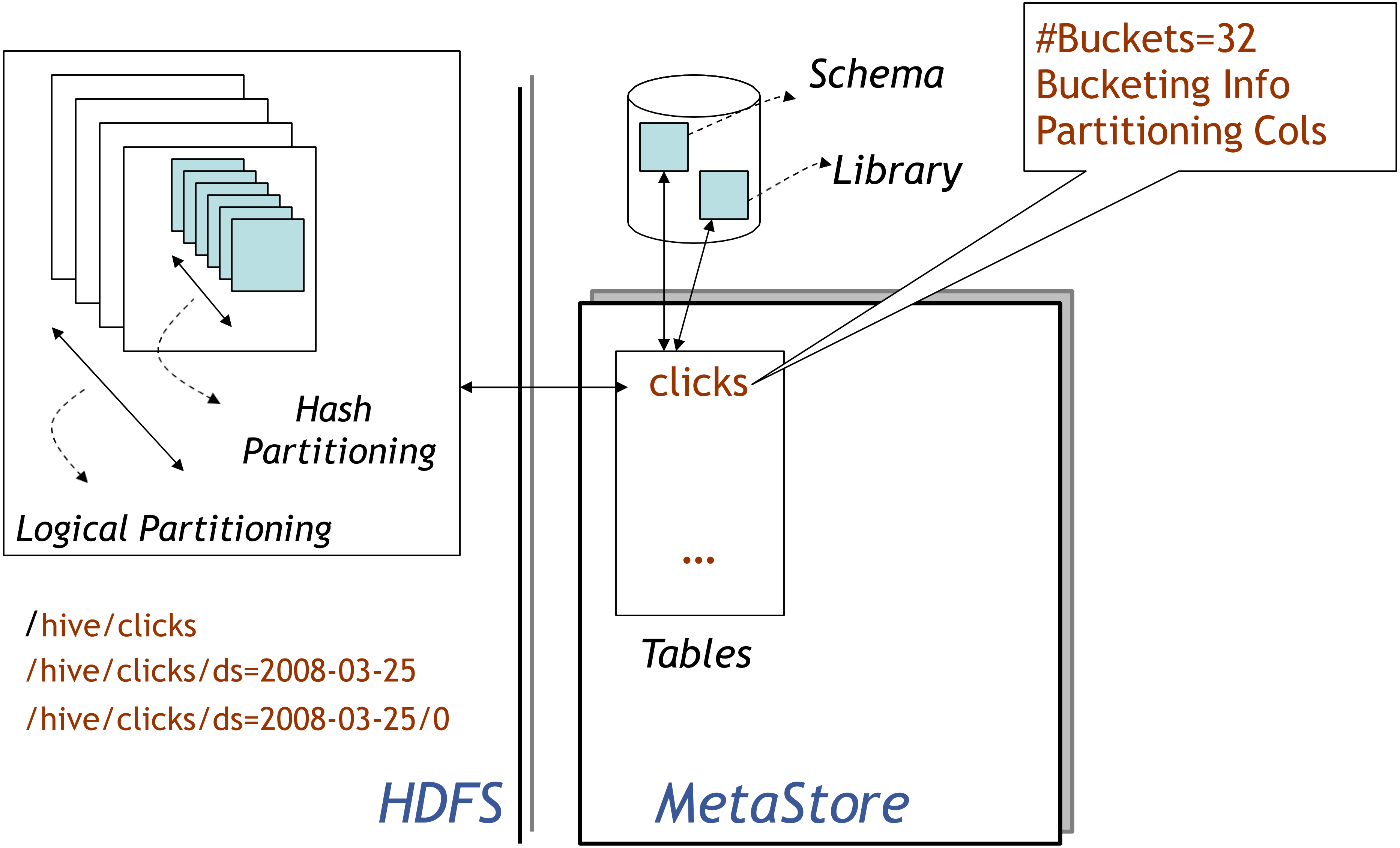
Hadoop Usage @ Facebook

- Data statistics:
  - Total Data: 180TB (mostly compressed)
  - Net Data added/day: 2+TB (compressed)
    - 6TB of uncompressed source logs
    - 4TB of uncompressed dimension data reloaded daily
- Usage statistics:
  - 3200 jobs/day with 800K tasks(map-reduce tasks)/day
  - 55TB of compressed data scanned daily
  - 15TB of compressed output data written to hdfs
  - 80 MM compute minutes/day

HIVE: Components



Data Model



/hive/clicks
/hive/clicks/ds=2008-03-25
/hive/clicks/ds=2008-03-25/0

Dealing with Structured Data

- Type system
  - Primitive types
  - Recursively build up using Composition/Maps/Lists
- ObjectInspector interface for user-defined types
  - To recursively list schema
  - To recursively access fields within a row object
- Generic (De)Serialization Interface (SerDe)
- Serialization families implement interface
  - Thrift DDL based SerDe
  - Delimited text based SerDe
  - You can write your own SerDe (XML, JSON …)

MetaStore

- Stores Table/Partition properties:
  - Table schema and SerDe library
  - Table Location on HDFS
  - Logical Partitioning keys and types
  - Partition level metadata
  - Other information
- Thrift API
  - Current clients in Php (Web Interface), Python interface to Hive, Java (Query Engine and CLI)
- Metadata stored in any SQL backend
- Future
  - Statistics
  - Schema Evolution

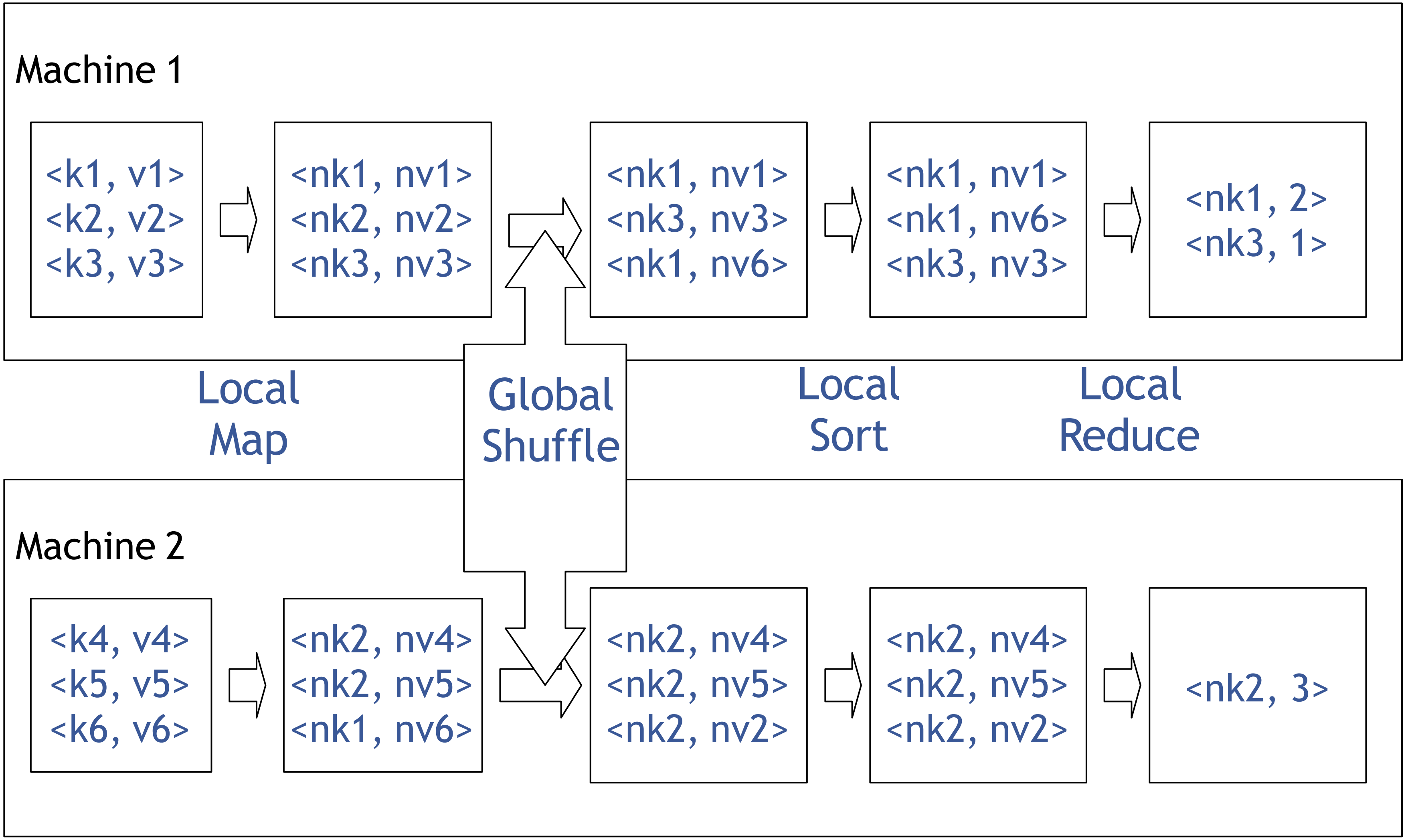
Hive Query Language

- Basic SQL
  - From clause subquery
  - ANSI JOIN (equi-join only)
  - Multi-table Insert
  - Multi group-by
  - Sampling
  - Objects traversal
- Extensibility
  - Pluggable Map-reduce scripts using TRANSFORM

Running Custom Map/Reduce Scripts

```
FROM (
    FROM pv_users
    SELECT TRANSFORM(pv_users.userid, pv_users.date) USING
      'map_script'
    AS(dt, uid)
    CLUSTER BY(dt)) map
INSERT INTO TABLE pv_users_reduced
    SELECT TRANSFORM(map.dt, map.uid) USING 'reduce_script'
      AS (date, count);
```

(Simplified) Map Reduce Review



Machine 1

<k1, v1>
<k2, v2>
<k3, v3>

<nk1, nv1>
<nk2, nv2>
<nk3, nv3>

<nk1, nv1>
<nk3, nv3>
<nk1, nv6>

<nk1, nv1>
<nk1, nv6>
<nk3, nv3>

<nk1, 2>
<nk3, 1>

Local Map

Global Shuffle

Local Sort

Local Reduce

Machine 2

<k4, v4>
<k5, v5>
<k6, v6>

<nk2, nv4>
<nk2, nv5>
<nk1, nv6>

<nk2, nv4>
<nk2, nv5>
<nk2, nv2>

<nk2, nv4>
<nk2, nv5>
<nk2, nv2>

<nk2, 3>

Hive QL - Join

page_view

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

X

user

userid	age	gender
111	25	female
222	32	male

=

pv_users

pageid	age
1	25
2	25
1	32

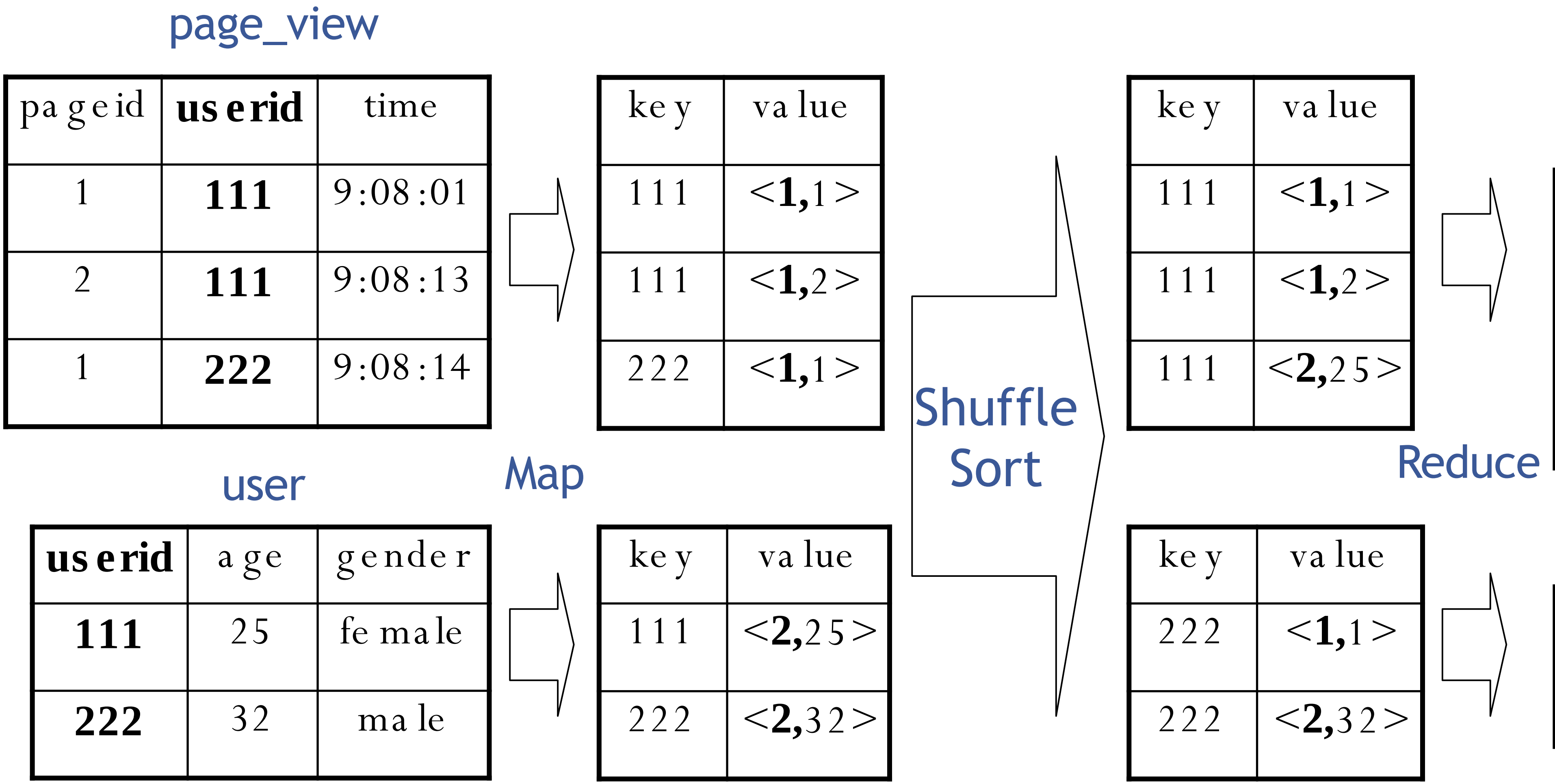
- SQL:

  INSERT INTO TABLE pv_users

  SELECT pv.pageid, u.age

  FROM page_view pv JOIN user u ON (pv.userid = u.userid);

Hive QL – Join in Map Reduce



page_view

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

user

userid	age	gender
111	25	female
222	32	male

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

key	value
111	<**2,**25>
222	<**2,**32>

key	value
111	<**1,**1>
111	<**1,**2>
111	<**2,**25>

key	value
222	<**1,**1>
222	<**2,**32>

Joins

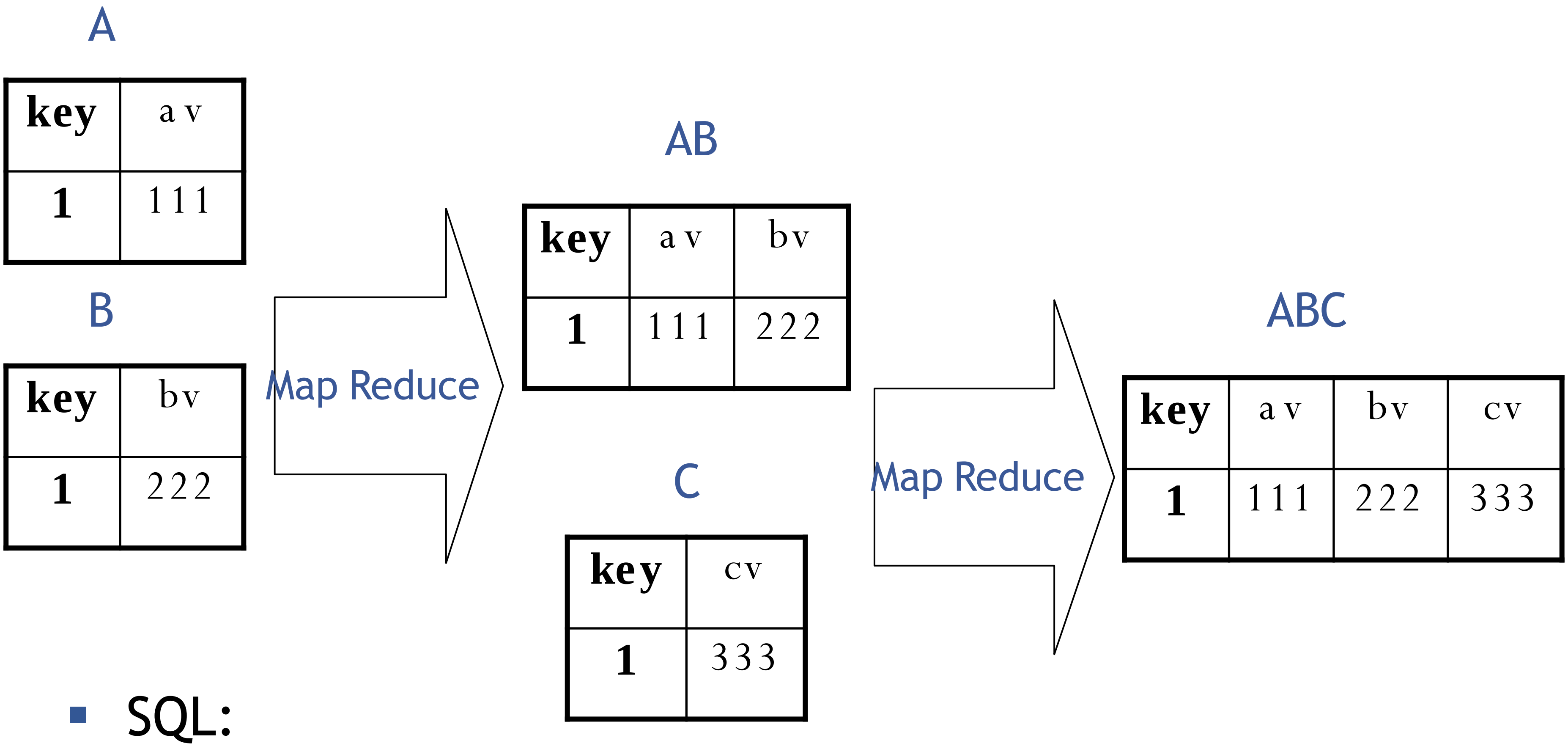
- Outer Joins

```
INSERT INTO TABLE pv_users
SELECT pv.*, u.gender, u.age
 FROM page_view pv FULL OUTER JOIN user u ON (pv.userid = u.id)
 WHERE pv.date = 2008-03-03;
```

Join To Map Reduce

- Only Equality Joins with conjunctions supported
- Future
  - Pruning of values send from map to reduce on the basis of projections
  - Make Cartesian product more memory efficient
  - Map side joins
    - Hash Joins if one of the tables is very small
    - Exploit pre-sorted data by doing map-side merge join

Hive Optimizations
– Merge Sequential Map Reduce Jobs



A

key	a v
1	111

B

key	b v
1	222

AB

key	a v	b v
1	111	222

C

key	c v
1	333

ABC

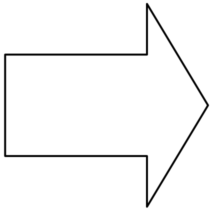
key	a v	b v	c v
1	111	222	333

- SQL:
  - FROM (a join b on a.key = b.key) join c on a.key = c.key SELECT …

Hive QL – Group By

pv_users

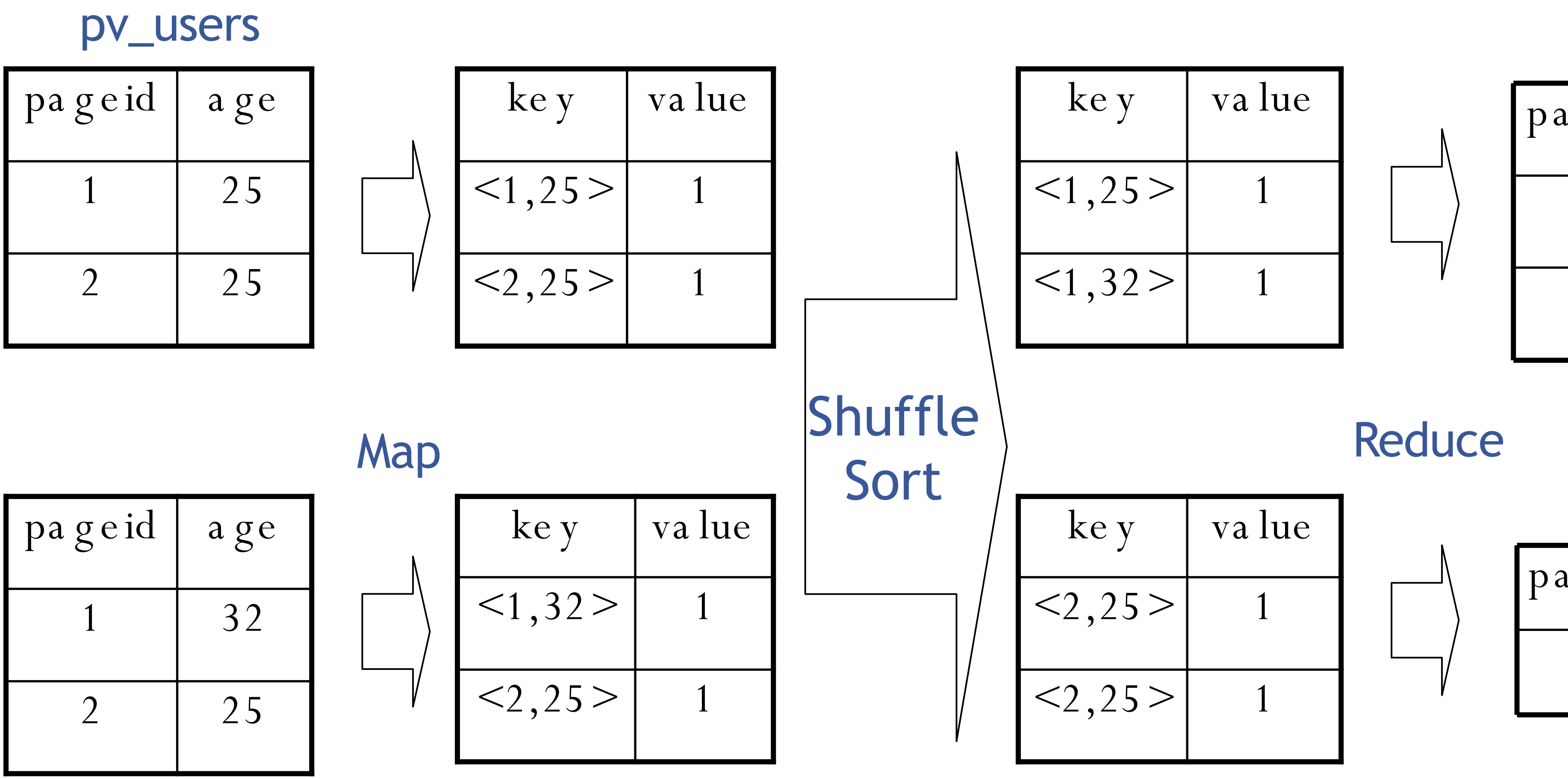
pageid	age
1	25
2	25
1	32
2	25



pageid	age	count
1	25	1
2	25	2
1	32	1

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

Hive QL – Group By in Map Reduce



pv_users

pageid	age
1	25
2	25

pageid	age
1	32
2	25

Map

key	value
<1,25>	1
<2,25>	1

key	value
<1,32>	1
<2,25>	1

Shuffle Sort

key	value
<1,25>	1
<1,32>	1

key	value
<2,25>	1
<2,25>	1

Reduce

pa

pa

Hive QL – Group By with Distinct

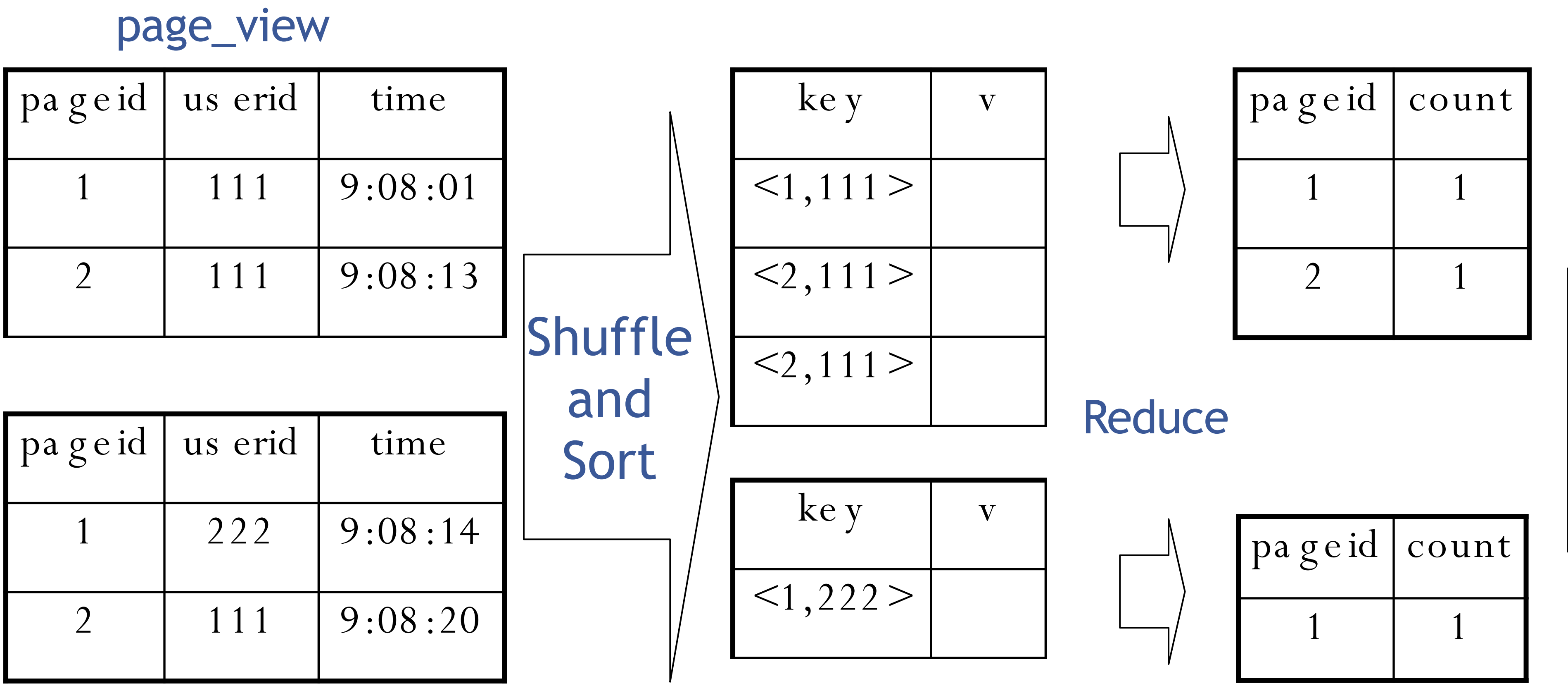
page_view

pageid	userid	time
1	111	9:08:01
2	111	9:08:13
1	222	9:08:14
2	111	9:08:20

pageid	count_distinct_userid
1	2
2	1

```
SELECT pageid, COUNT(DISTINCT userid)
FROM page_view GROUP BY pageid
```

Hive QL – Group By with Distinct in Map Reduce



page_view

pageid	userid	time
1	111	9:08:01
2	111	9:08:13

pageid	userid	time
1	222	9:08:14
2	111	9:08:20

Shuffle and Sort

key	v
<1,111>	
<2,111>	
<2,111>	

key	v
<1,222>	

Reduce

pageid	count
1	1
2	1

pageid	count
1	1

Group by Future optimizations

- Map side partial aggregations
  - Hash Based aggregates
  - Exploit pre-sorted data for distinct counts
- Partial aggregations in Combiner
- Be smarter about how to avoid multiple stage
- Exploit table/column statistics for deciding strategy

Inserts into Files, Tables and Local Files

```
FROM pv_users
INSERT INTO TABLE pv_gender_sum
    SELECT pv_users.gender, count_distinct(pv_users.userid)
    GROUP BY(pv_users.gender)
INSERT INTO DIRECTORY ‘/user/facebook/tmp/pv_age_sum.dir’
    SELECT pv_users.age, count_distinct(pv_users.userid)
    GROUP BY(pv_users.age)
INSERT INTO LOCAL DIRECTORY ‘/home/me/pv_age_sum.dir’
    FIELDS TERMINATED BY ‘,’ LINES TERMINATED BY \013
    SELECT pv_users.age, count_distinct(pv_users.userid)
    GROUP BY(pv_users.age);
```

Future Work

- Cost-based optimization
- Multiple interfaces (JDBC…)
- Performance Comparisons with similar work (PIG)
- SQL Compliance (order by, nested queries…)
- Integration with BI tools
- Data Compression
    - Columnar storage schemes
    - Exploit lazy/functional Hive field retrieval interfaces
- Better data locality
    - Co-locate hash partitions on same rack
    - Exploit high intra-rack bandwidth for merge joins

Hive Performance

- full table aggregate (not grouped)
- Input data size: 1,407,867,660 (32 files)
- count in mapper and 2 map-reduce jobs for sum
  - time taken 30 seconds
  - Test cluster: 10 nodes

```
from (
        from test t select transform (t.userid) as (cnt) using myCount'
        ) mout
select sum(mout.cnt);
```

Hadoop Challenges @ Facebook

- QOS/Isolation:
  - Big jobs can hog the cluster
  - JobTracker memory as limited resource
  - Limit memory impact of runaway tasks

  ➔ Fair Scheduler (Matei)

- Protection
  - What if a software bug corrupts the NameNode transaction log/image?

  ➔ HDFS SnapShots (Dhruba)

- Data Archival
  - Not all data is hot and needs colocation with Compute

  ➔ HDFS Symlinks (Dhruba) Data Archival

- Performance
  - Really hard to understand what bottlenecks are

facebook

Conclusion

- Available as a contrib project in hadoop
  - http://svn.apache.org/repos/asf/hadoop/core/
  - Checkout src/contrib/hive from trunk (works against 0.19 onwards)
- Latest distributions (including for hadoop-0.17) at: http://mirror.facebook.com/facebook/hive/
- People:
  - Suresh Anthony
  - Zheng Shao
  - Prasad Chakka
  - Pete Wyckoff
  - Namit Jain
  - Raghu Murthy
  - Joydeep Sen Sarma
  - Ashish Thusoo