Introduction to cloud computing

Jiaheng Lu

Department of Computer Science

Renmin University of China

www.jiahenglu.net

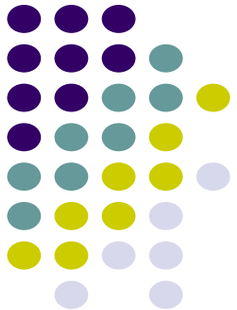
Hadoop/Hive

Open-Source Solution for Huge Data Sets

Data Scalability Problems

- **Search Engine**
  - 10KB / doc * 20B docs = 200TB
  - Reindex every 30 days: 200TB/30days = 6 TB/day
- **Log Processing / Data Warehousing**
  - 0.5KB/events * 3B pageview events/day = 1.5TB/day
  - 100M users * 5 events * 100 feed/event * 0.1KB/feed = 5TB/day
- **Multipliers:** 3 copies of data, 3-10 passes of raw data
- **Processing Speed (Single Machine)**
  - 2-20MB/second * 100K seconds/day = 0.2-2 TB/day

Google's Solution

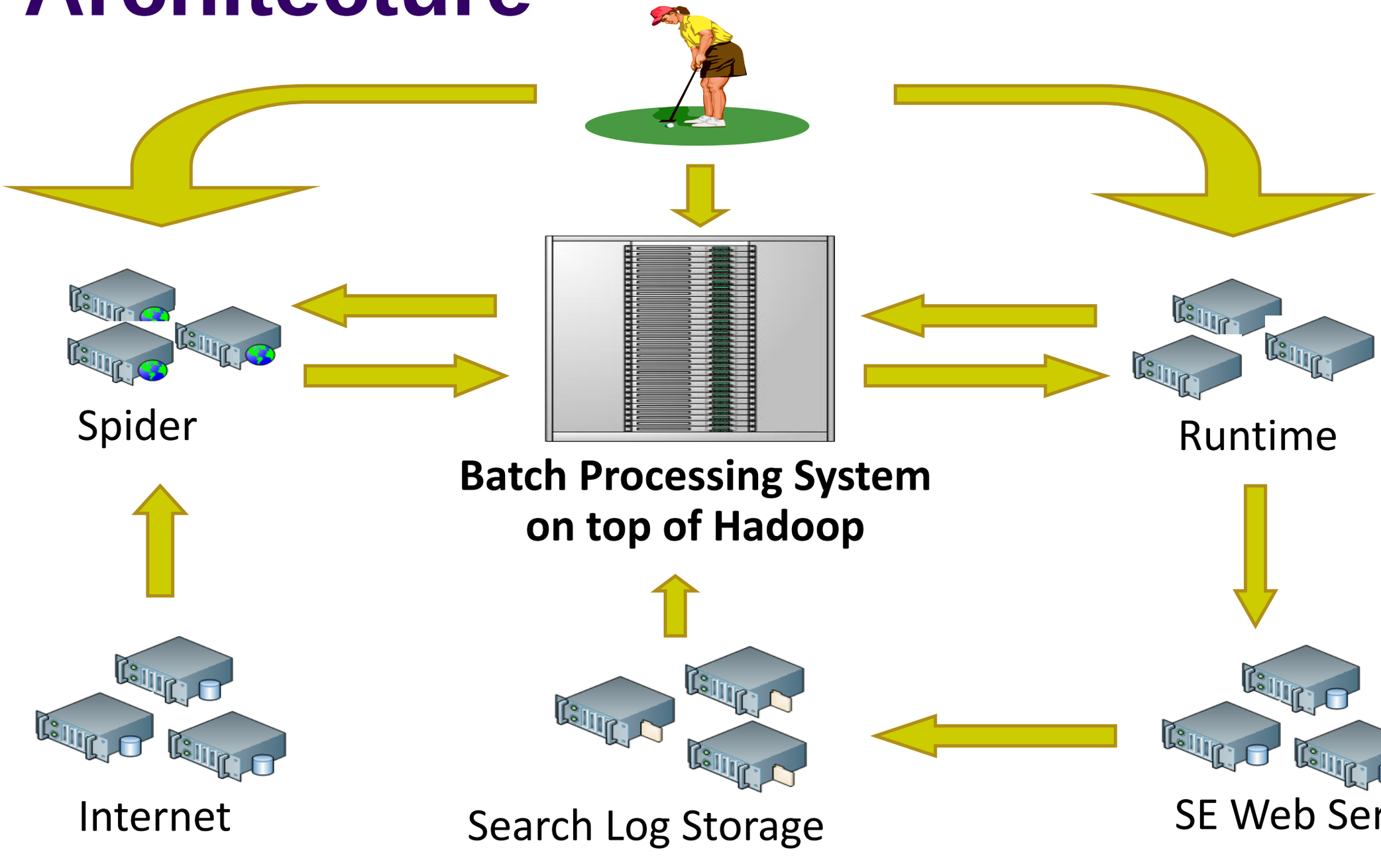


- **Google File System** – *SOSP'2003*
- **Map-Reduce** – *OSDI'2004*
- **Sawzall** – *Scientific Programming Journal'2005*
- **Big Table** – *OSDI'2006*
- **Chubby** – *OSDI'2006*

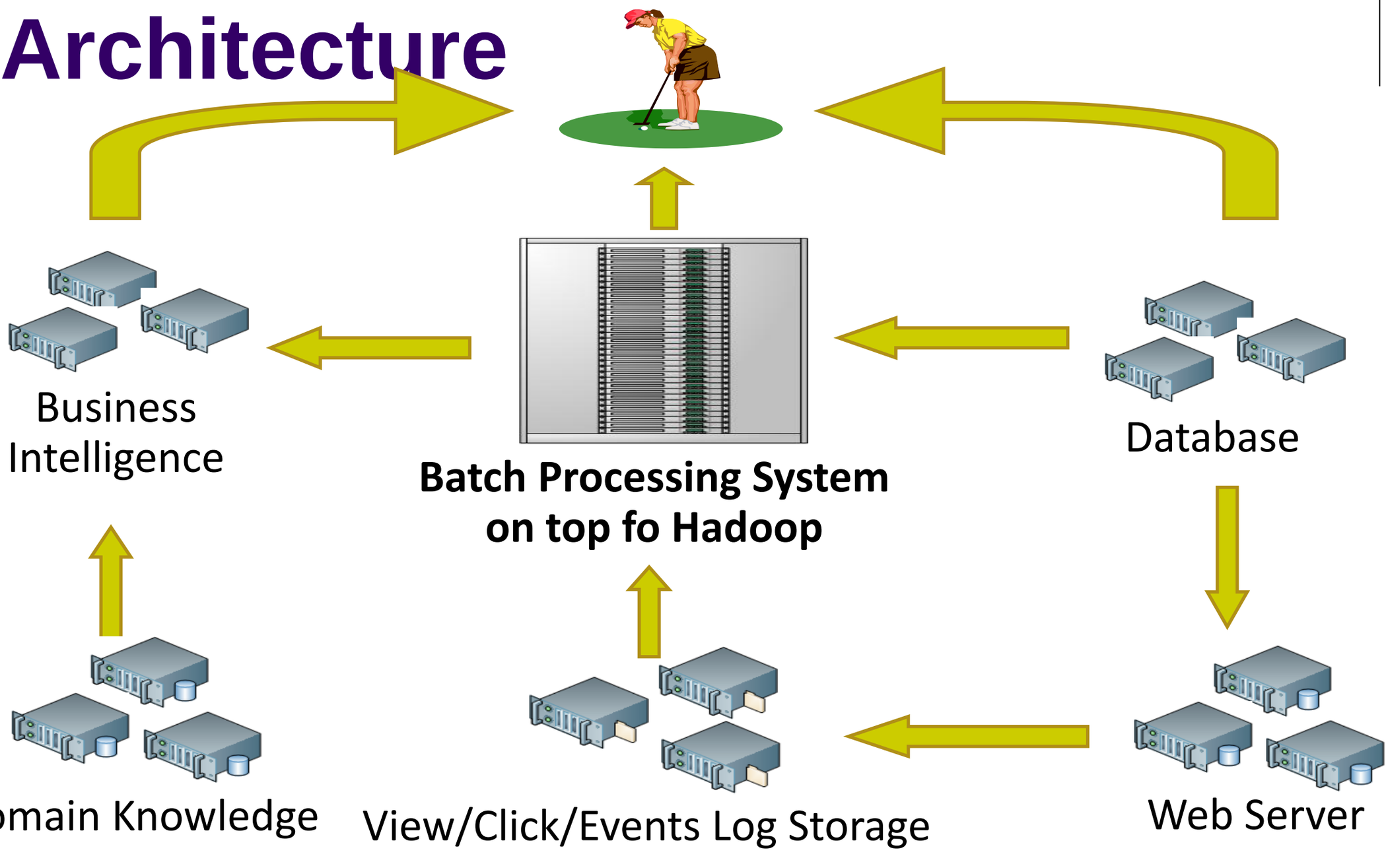
Open Source World's Solution

- **Google File System –** *Hadoop Distributed FS*
- **Map-Reduce –** *Hadoop Map-Reduce*
- **Sawzall –** *Pig, Hive, JAQL*
- **Big Table –** *Hadoop HBase*, *Cassandra*
- **Chubby** – *Zookeeper*

Simplified Search Engine Architecture



Simplified Data Warehouse Architecture



Hadoop History

- **Jan 2006 – Doug Cutting joins Yahoo**
- Feb 2006 – Hadoop splits out of Nutch and Yahoo starts using it.
- **Dec 2006 – Yahoo creating 100-node Webmap with Hadoop**
- Apr 2007 – Yahoo on 1000-node cluster
- Jan 2008 – Hadoop made a top-level Apache project
- **Dec 2007 – Yahoo creating 1000-node Webmap with Hadoop**
- Sep 2008 – Hive added to Hadoop as a contrib project

Hadoop Introduction

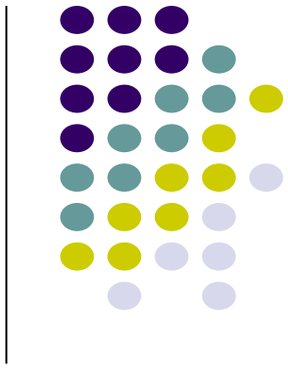
- **Open Source Apache Project**
  - http://hadoop.apache.org/
  - Book: http://oreilly.com/catalog/9780596521998/index.html
- **Written in Java**
  - Does work with other languages
- **Runs on**
  - Linux, Windows and more
  - Commodity hardware with high failure rate

Current Status of Hadoop

- **Largest Cluster**
  - 2000 nodes (8 cores, 4TB disk)
- **Used by 40+ companies / universities over the world**
  - Yahoo, Facebook, etc
  - Cloud Computing Donation from Google and IBM
- **Startup focusing on providing services for hadoop**
  - Cloudera

Hadoop Components

- **Hadoop Distributed File System (HDFS)**
- **Hadoop Map-Reduce**
- **Contributes**
  - Hadoop Streaming
  - Pig / JAQL / Hive
  - HBase



Hadoop Distributed File System

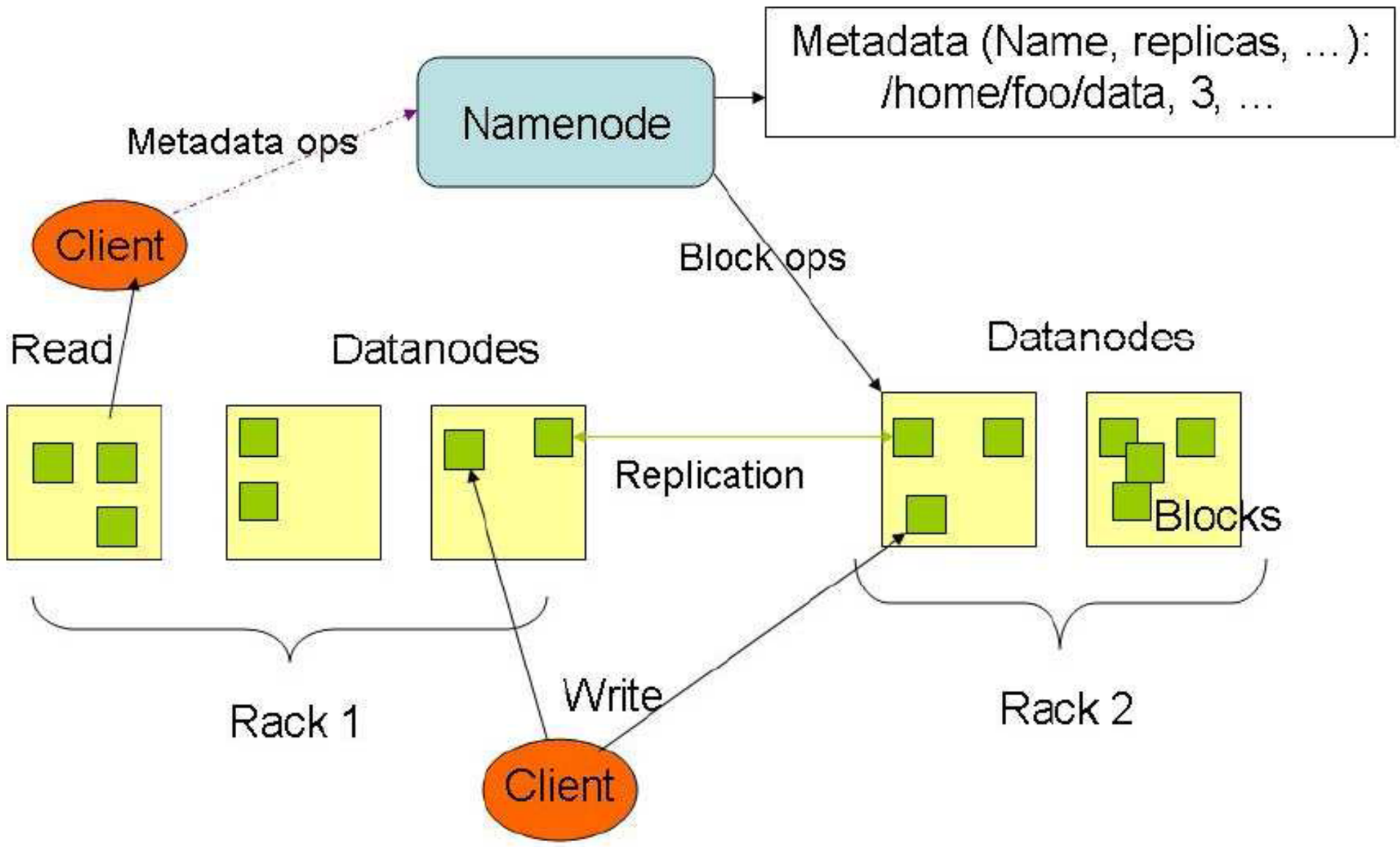
Goals of HDFS

- **Very Large Distributed File System**
  - 10K nodes, 100 million files, 10 PB
- **Convenient Cluster Management**
  - Load balancing
  - Node failures
  - Cluster expansion
- **Optimized for Batch Processing**
  - Allow move computation to data
  - Maximize throughput

HDFS Details

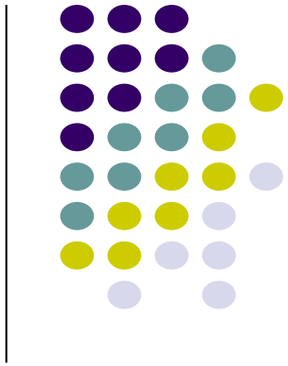
- **Data Coherency**
  - Write-once-read-many access model
  - Client can only append to existing files
- **Files are broken up into blocks**
  - Typically 128 MB block size
  - Each block replicated on multiple DataNodes
- **Intelligent Client**
  - Client can find location of blocks
  - Client accesses data directly from DataNode

HDFS Architecture



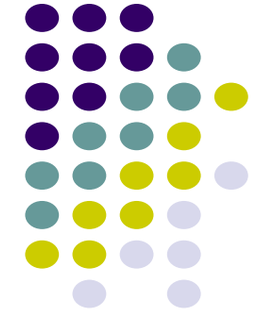
HDFS User Interface

- **Java API**
- **Command Line**
  - hadoop dfs -mkdir /foodir
  - hadoop dfs -cat /foodir/myfile.txt
  - hadoop dfs -rm /foodir myfile.txt
  - hadoop dfsadmin -report
  - hadoop dfsadmin -decommission datanodename
- **Web Interface**
  - http://host:port/dfshealth.jsp

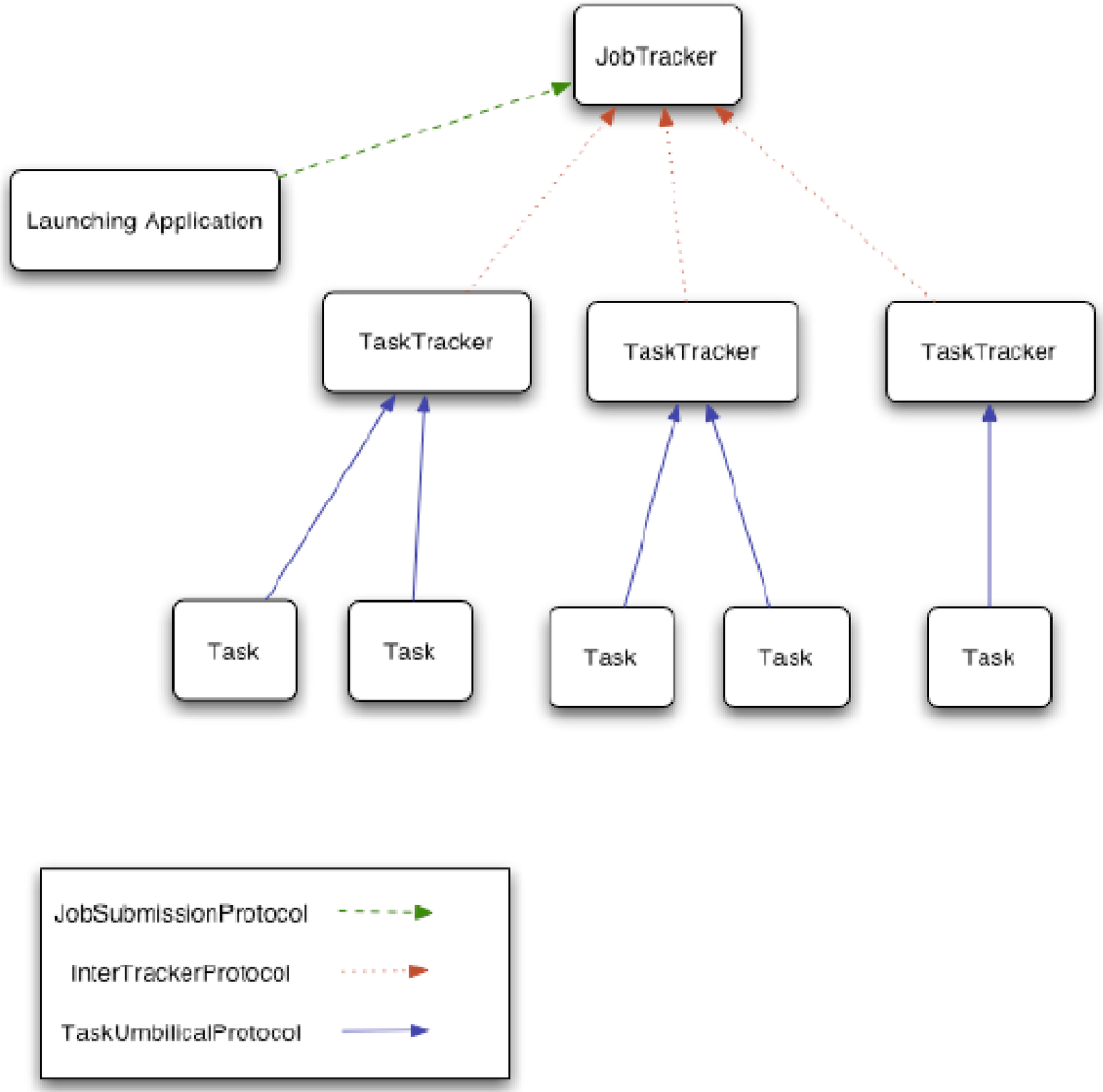


Hadoop Map-Reduce and Hadoop Streaming

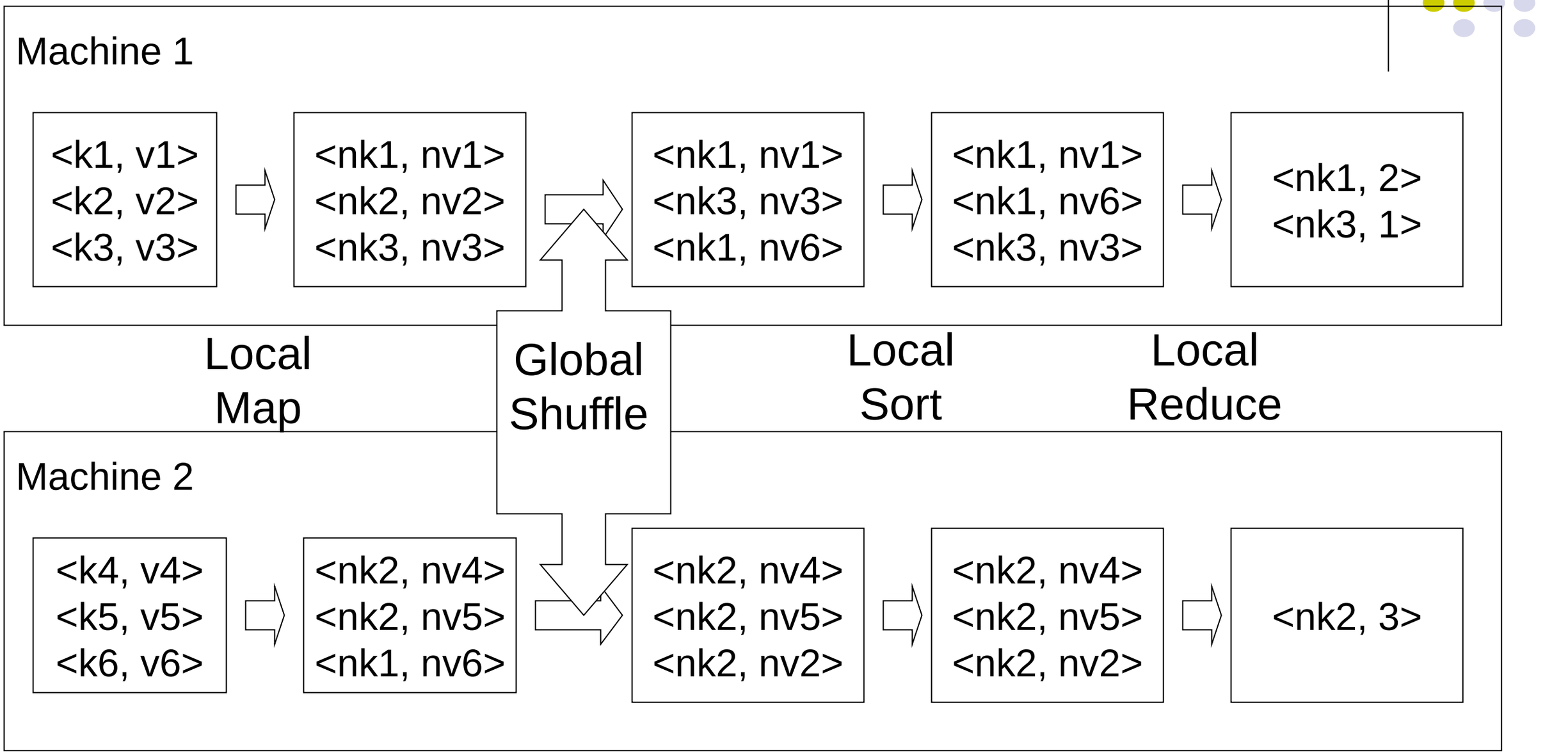
Hadoop Map-Reduce Introduction



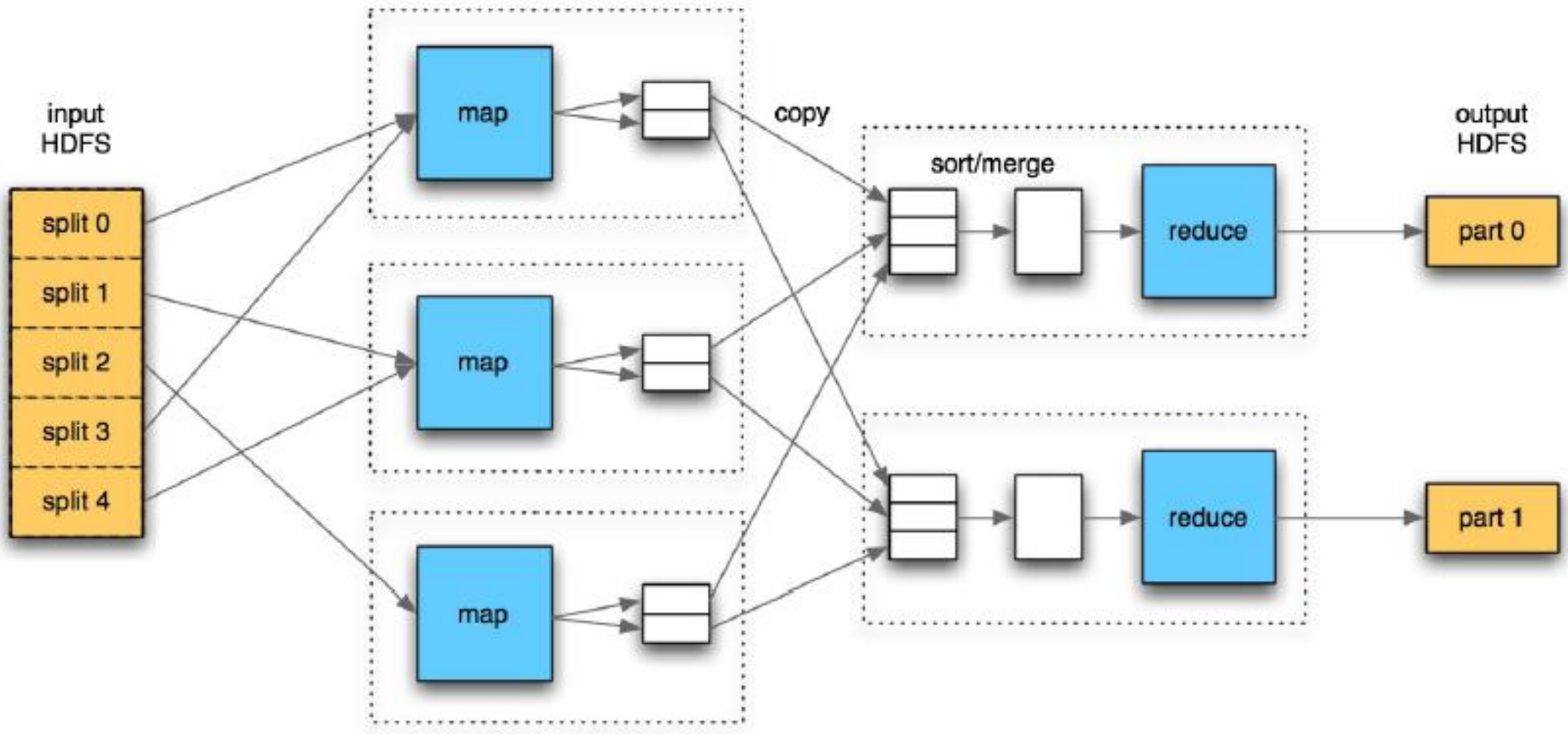
- **Map/Reduce works like a parallel Unix pipeline:**
  - cat input | grep | sort | uniq -c | cat > output
  - Input | Map | Shuffle & Sort | Reduce | Output
- **Framework does inter-node communication**
  - Failure recovery, consistency etc
  - Load balancing, scalability etc
- **Fits a lot of batch processing applications**
  - Log processing
  - Web index building


JobTracker
Launching Application
TaskTracker
TaskTracker
TaskTracker
Task
Task
Task
Task
Task
JobSubmissionProtocol
InterTrackerProtocol
TaskUmbilicalProtocol

(Simplified) Map Reduce Review



Physical Flow



Example Code

```
public void map(LongWritable key, Text val,
    OutputCollector<Text, IntWritable> output,
    Reporter reporter) throws IOException {

  if (pattern.matcher(val.toString()).matches()) {
    output.collect(val, new IntWritable(1));
  }
}

public void reduce(Text key, Iterator<IntWritable> vals,
    OutputCollector<Text, IntWritable> output,
    Reporter reporter) throws IOException {

  int sum = 0;
  while (vals.hasNext()) {
    sum += vals.next().get();
  }
  output.collect(key, new IntWritable(sum));
}
```

Hadoop Streaming

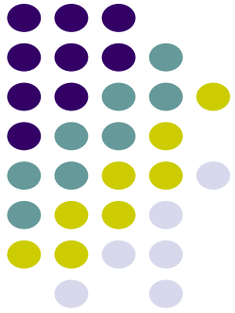
- **Allow to write Map and Reduce functions in any languages**
  - Hadoop Map/Reduce only accepts Java
- **Example: Word Count**
  - hadoop streaming
    -input /user/zshao/articles
    -mapper ‘tr “ ” “\n”’
    -reducer ‘uniq -c‘
    -output /user/zshao/
    -numReduceTasks 32

Hive - SQL on top of Hadoop

Map-Reduce and SQL

- **Map-Reduce is scalable**
  - SQL has a huge user base
  - SQL is easy to code
- **Solution: Combine SQL and Map-Reduce**
  - Hive on top of Hadoop (open source)
  - Aster Data (proprietary)
  - Green Plum (proprietary)

Hive



- **A database/data warehouse on top of Hadoop**
  - Rich data types (structs, lists and maps)
  - Efficient implementations of SQL filters, joins and group-by's on top of map reduce
- **Allow users to access Hive data without using Hive**
- **Link:**
  - **http://svn.apache.org/repos/asf/hadoop/hive/trunk/**

Dealing with Structured Data

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

MetaStore

- Stores Table/Partition properties:
  - Table schema and SerDe library
  - Table Location on HDFS
  - Logical Partitioning keys and types
  - Other information
- Thrift API
  - Current clients in Php (Web Interface), Python (old CLI), Java (Query Engine and CLI), Perl (Tests)
- Metadata can be stored as text files or even in a SQL backend

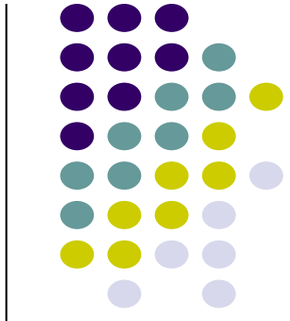
Hive CLI

- DDL:
  - create table/drop table/rename table
  - alter table add column
- Browsing:
  - show tables
  - describe table
  - cat table
- Loading Data
- Queries

Web UI for Hive

- MetaStore UI:
  - Browse and navigate all tables in the system
  - Comment on each table and each column
  - Also captures data dependencies
- HiPal:
  - Interactively construct SQL queries by mouse clicks
  - Support projection, filtering, group by and joining
  - Also support

Hive Query Language

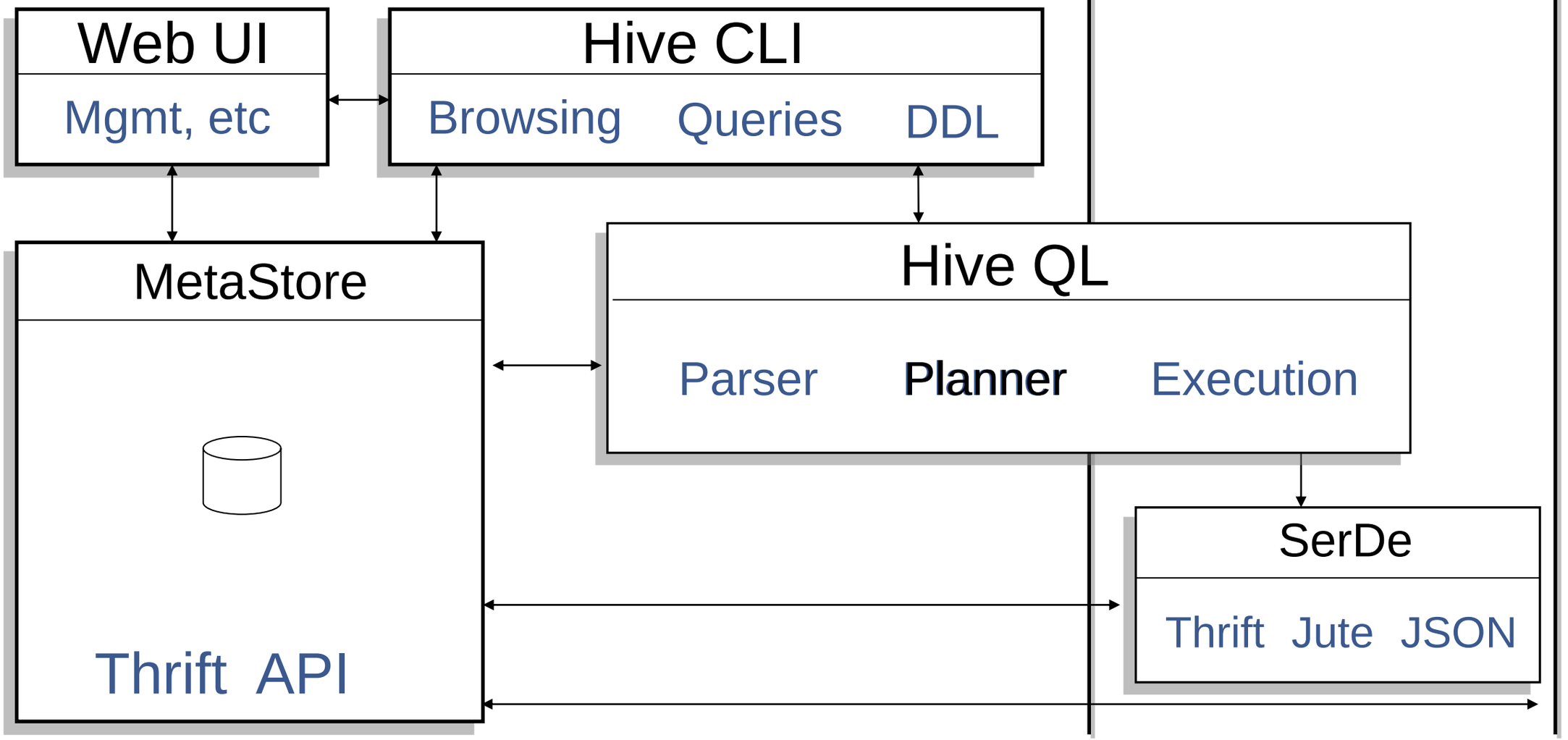


- Philosophy
  - SQL
  - Map-Reduce with custom scripts (hadoop streaming)
- Query Operators
  - Projections
  - Equi-joins
  - Group by
  - Sampling
  - Order By

Hive QL – Custom Map/Reduce Scripts

- Extended SQL:
  - FROM (
    - FROM pv_users
    - **MAP** pv_users.userid, pv_users.date
    - **USING** 'map_script' AS (dt, uid)
    - **CLUSTER BY** dt) map
  - INSERT INTO TABLE pv_users_reduced
    - **REDUCE** map.dt, map.uid
    - **USING** 'reduce_script' AS (date, count);
- Map-Reduce: similar to hadoop streaming

Hive Architecture



Hive QL – Join

page_view

page id	**user id**	time
1	**111**	9:08:01
2	**111**	9:08:13
1	**222**	9:08:14

X

user

user id	age	gender
111	25	female
222	32	male

=

pv_users

page id	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

page id	**user id**	time
1	**111**	9:08:01
2	**111**	9:08:13
1	**222**	9:08:14

user

user id	age	gender
111	25	female
222	32	male

Map

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

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

Shuffle Sort

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

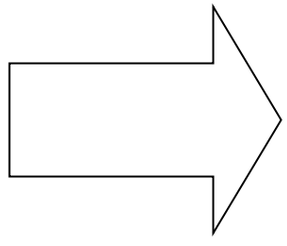
key	value
222	<**1**,1>
222	<**2**,32>

Reduce

Hive QL – Group By

pv_users

page id	age
1	25
2	25
1	32
2	25



pageid_age_sum

pageid	age	Count
1	25	1
2	25	2
1	32	1

- SQL:
  - INSERT INTO TABLE pageid_age_sum
  - SELECT pageid, age, count(1)
  - FROM pv_users
  - GROUP BY pageid, age;

Hive QL – Group By in Map Reduce

pv_users

page id	age
1	25
2	25

page id	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

p

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

result

pageid	count_distinct_userid
1	2
2	1

- SQL
  - 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
1	222	9:08:14
2	111	9:08:20

Shuffle and Sort

key	v
<1,111>	
<1,222>	
<2,111>	
<2,111>	

Reduce

pageid	count
1	2
2	1

Shuffle key is a prefix of the sort key.

Hive QL: Order By

page_view

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

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

Shuffle randomly.

Shuffle and Sort

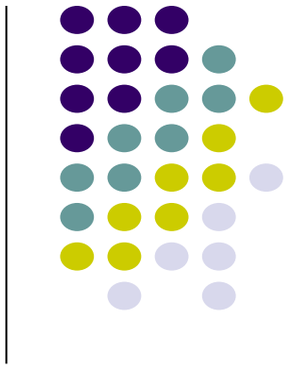
key	v
<1,111>	9:08:01
<2,111>	9:08:13

key	v
<1,222>	9:08:14
<2,111>	9:08:20

Reduce

pageid	userid	t
1	111	9
2	111	9

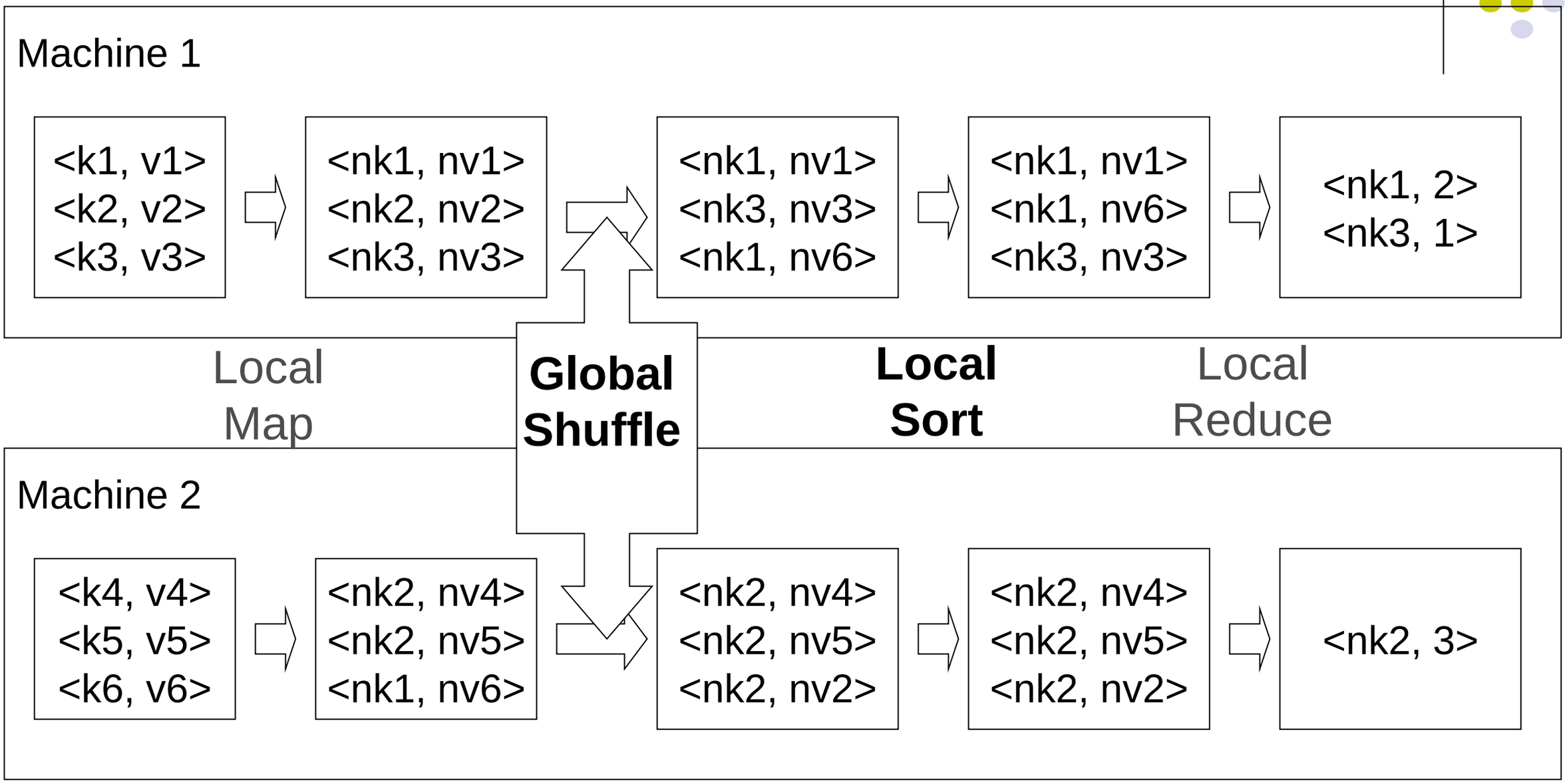
pageid	userid	t
1	222	9
2	111	9



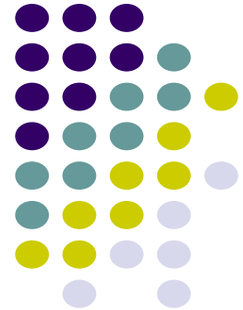
Hive Optimizations

Efficient Execution of SQL on top of Map-Reduce

(Simplified) Map Reduce Revisit



Merge Sequential Map Reduce Jobs



A

key	av
1	111

B

key	bv
1	222

Map Reduce

AB

ke y	av	bv
1	111	222

C

key	cv
1	333

Map Reduce

ABC

ke y	av	bv	cv
1	111	222	333

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

Share Common Read Operations

page id	age
1	25
2	32

Map Reduce

page id	count
1	1
2	1

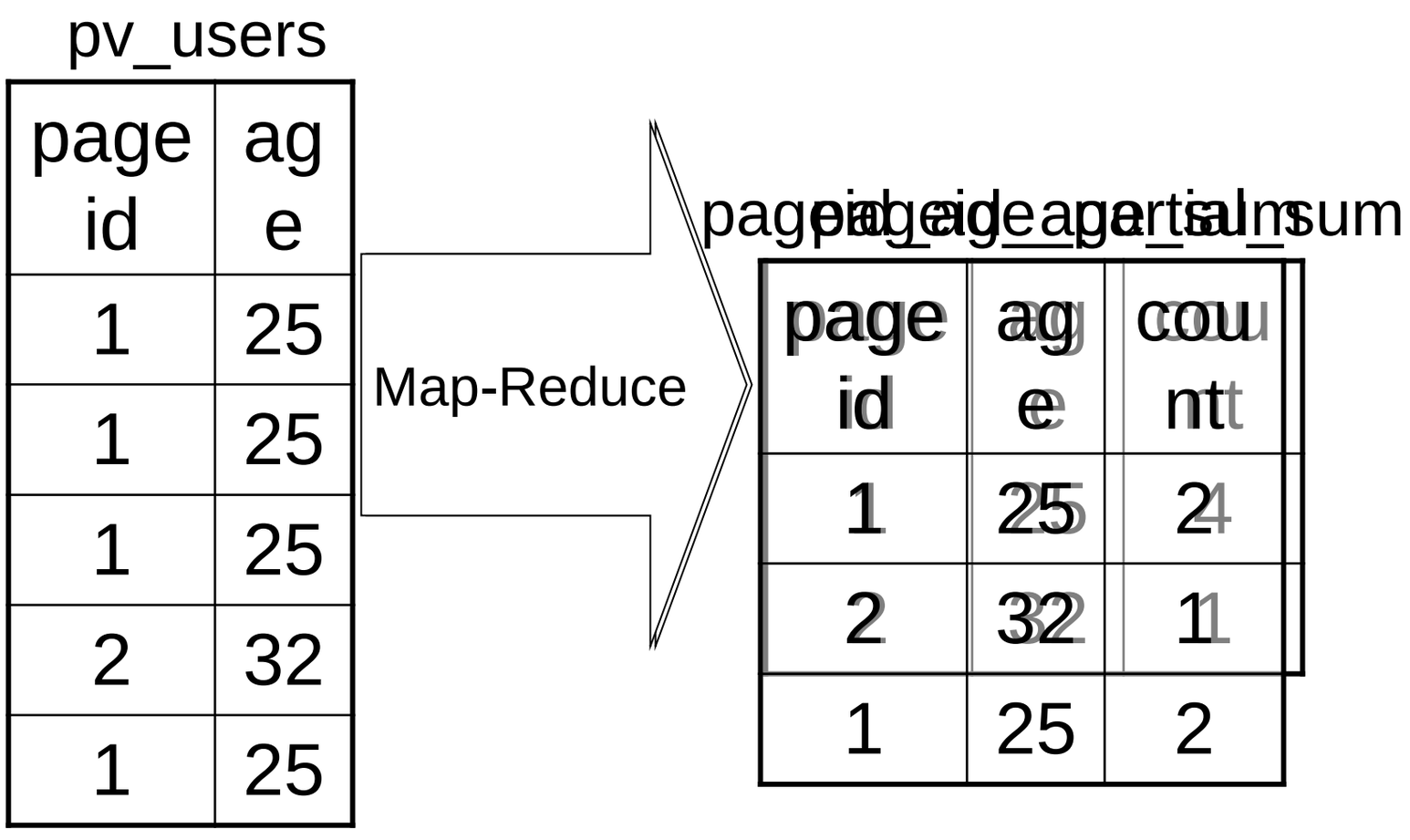
page id	age
1	25
2	32

Map Reduce

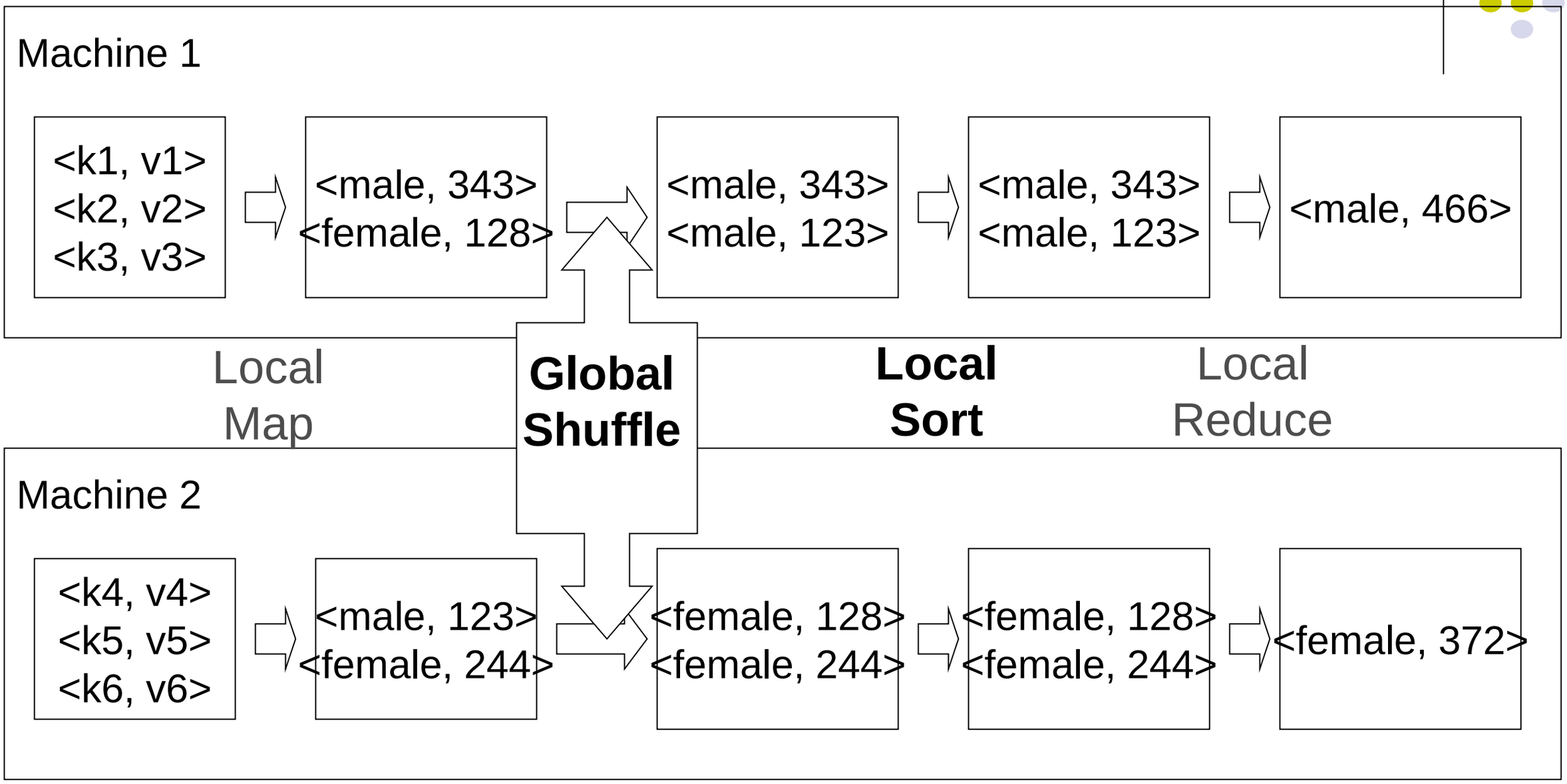
age	count
25	1
32	1

- Extended SQL
  - **FROM pv_users**
  - **INSERT** INTO TABLE pv_pageid_sum
    - SELECT pageid, count(1)
    - GROUP BY pageid
  - **INSERT** INTO TABLE pv_age_sum
    - SELECT age, count(1)
    - GROUP BY age;

Load Balance Problem



Map-side Aggregation / Combiner



Query Rewrite

- **Predicate Push-down**
  - select * from (select * from t) where col1 = ‘2008’;
- **Column Pruning**
  - select col1, col3 from (select * from t);