Big Data Analytics

Introduction

Theme of this Course

Large-Scale Data Management

Big Data Analytics

Data Science and Analytics

- How to manage very large amounts of data and extract value and knowledge from them

Introduction to Big Data

What is Big Data?

What makes data, "Big" Data?

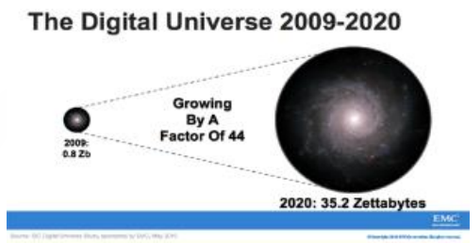
Big Data Definition

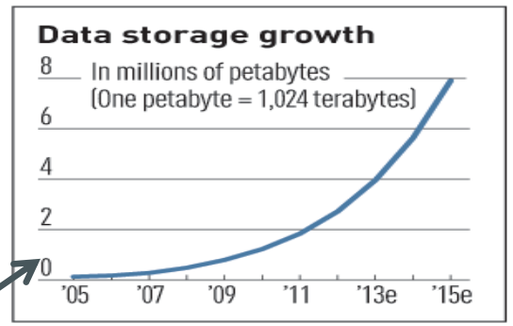
- No single standard definition…

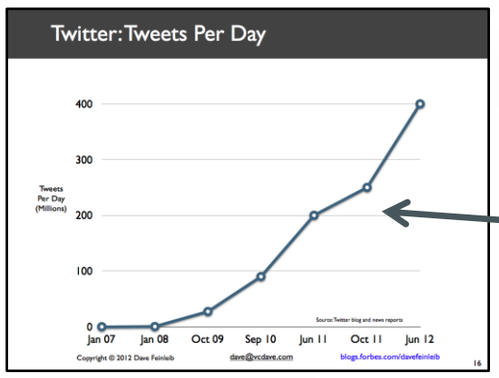
"***Big Data***" is data whose scale, diversity, and complexity require new architecture, techniques, algorithms, and analytics to manage it and extract value and hidden knowledge from it…

Characteristics of Big Data:
1-Scale (Volume)

- **Data Volume**
  - 44x increase from 2009 2020
  - From 0.8 zettabytes to 35zb
- Data volume is increasing exponentially



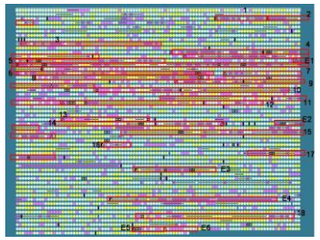


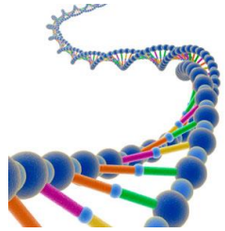


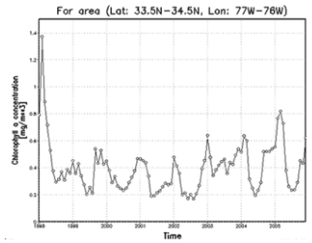
Exponential increase in collected/generated data

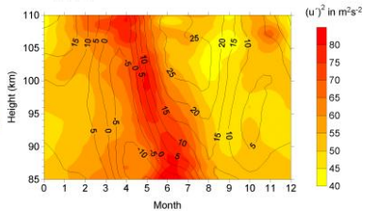
Characteristics of Big Data:
2-Complexity (Varity)

- Various formats, types, and structures
- Text, numerical, images, audio, video, sequences, time series, social media data, multi-dim arrays, etc…
- Static data vs. streaming data
- A single application can be









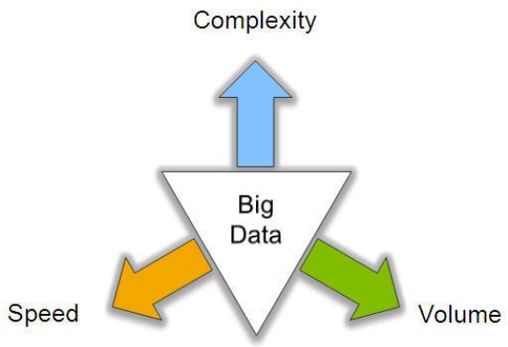
To extract knowledge➔ all these types of data need to linked together

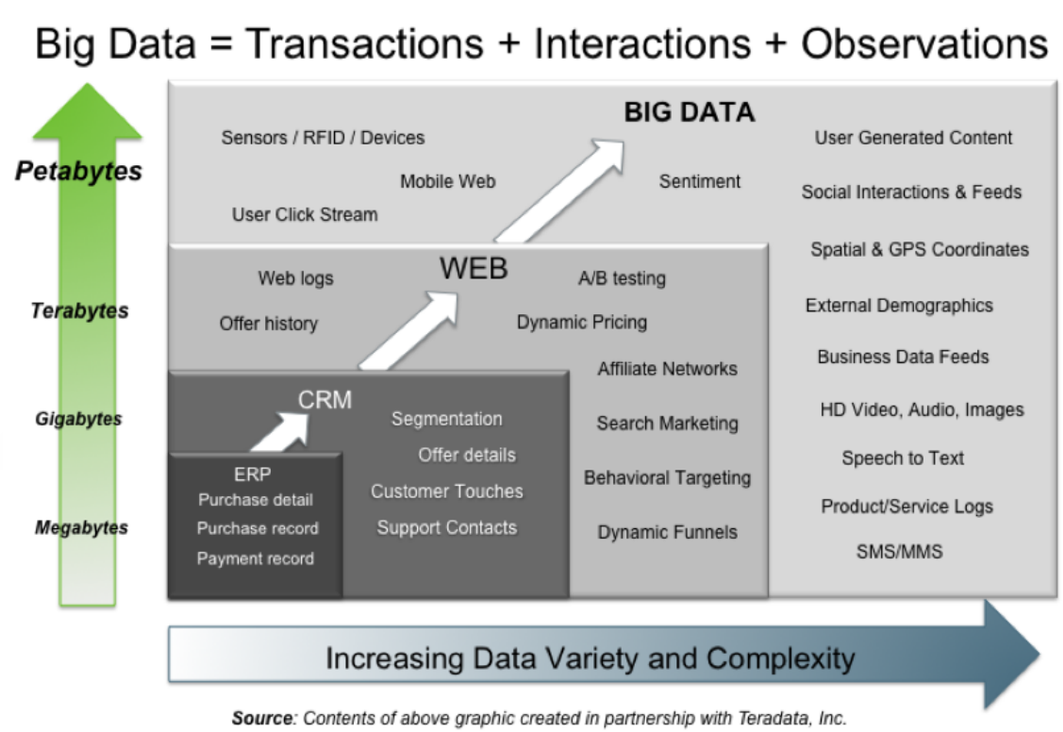
Characteristics of Big Data:
3-Speed (Velocity)

- Data is begin generated fast and need to be processed fast
- Online Data Analytics
- Late decisions ➔ missing opportunities

- **Examples**
  - **E-Promotions:** Based on your current location, your purchase history, what you like ➔ send promotions right now for store next to you
  - **Healthcare monitoring:** sensors monitoring your activities and body ➔ any abnormal measurements require immediate reaction

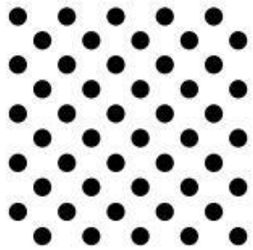
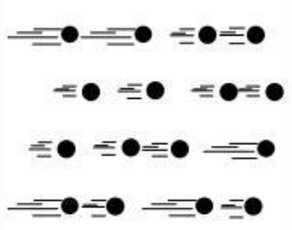
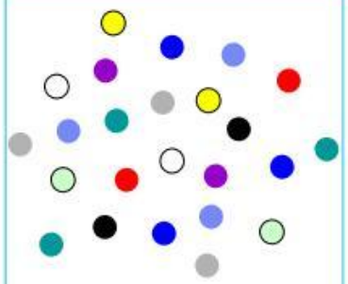
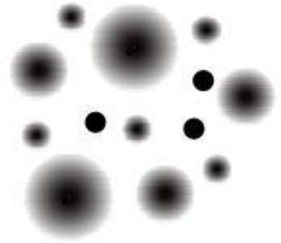
Big Data: 3V's



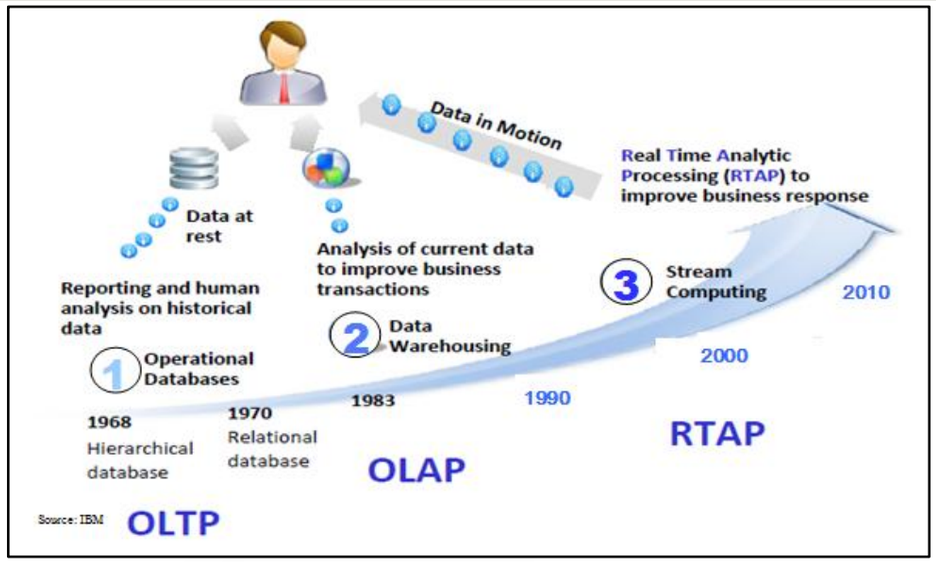


***Source**: Contents of above graphic created in partnership with Teradata, Inc.*

Some Make it 4V's

Volume	Velocity	Variety	Veracity*
			
Data at Rest	**Data in Motion**	**Data in Many Forms**	**Data in Doubt**
Terabytes to exabytes of existing data to process	Streaming data, milliseconds to seconds to respond	Structured, unstructured, text, multimedia	Uncertainty due to data inconsistency & incompleteness, ambiguities, latency, deception, model approximations

Harnessing Big Data



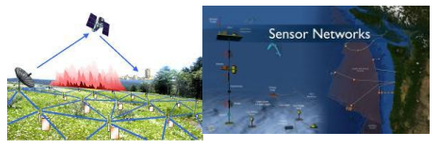
- **OLTP:** Online Transaction Processing (DBMSs)
- **OLAP:** Online Analytical Processing (Data Warehousing)
- **RTAP:** Real-Time Analytics Processing (Big Data Architecture & technology)

Who's Generating Big Data

Social media and networks
(all of us are generating data)

Scientific instruments
(collecting all sorts of data)

Mobile devices
(tracking all objects all the time)



Sensor technology and networks
(measuring all kinds of data)

- The progress and innovation is no longer hindered by the ability to collect data
- But, by the ability to manage, analyze, summarize, visualize, and discover knowledge from the collected data in a timely manner and in a scalable fashion

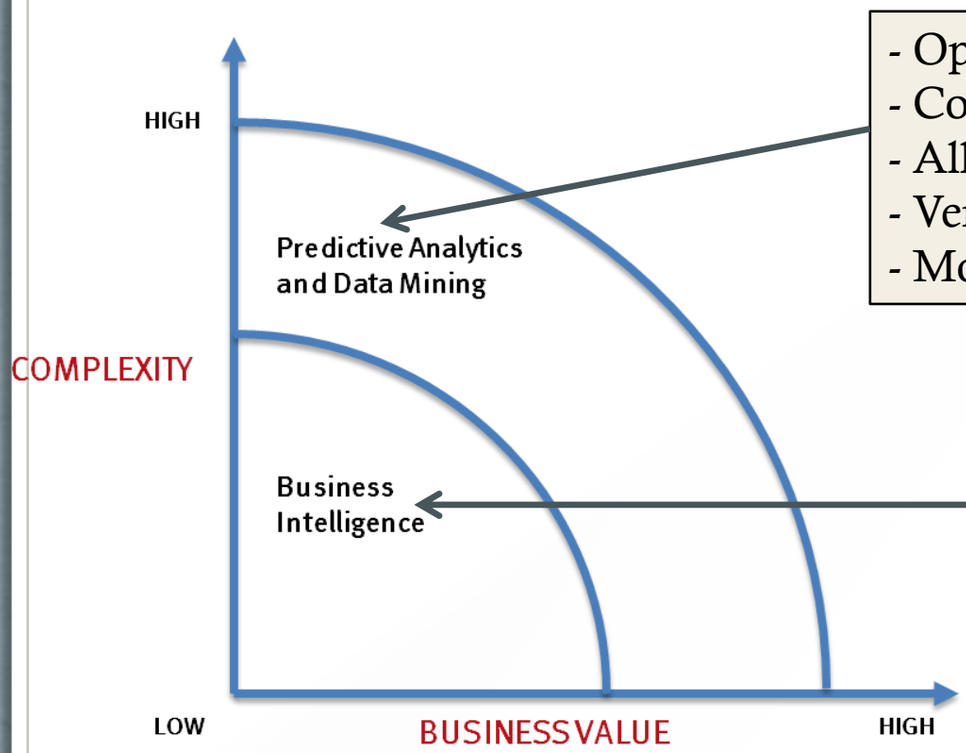
The Model Has Changed…

- **The Model of Generating/Consuming Data has Changed**

Old Model: Few companies are generating data, all others are consuming data

New Model: all of us are generating data, and all of us are consuming data

What's driving Big Data

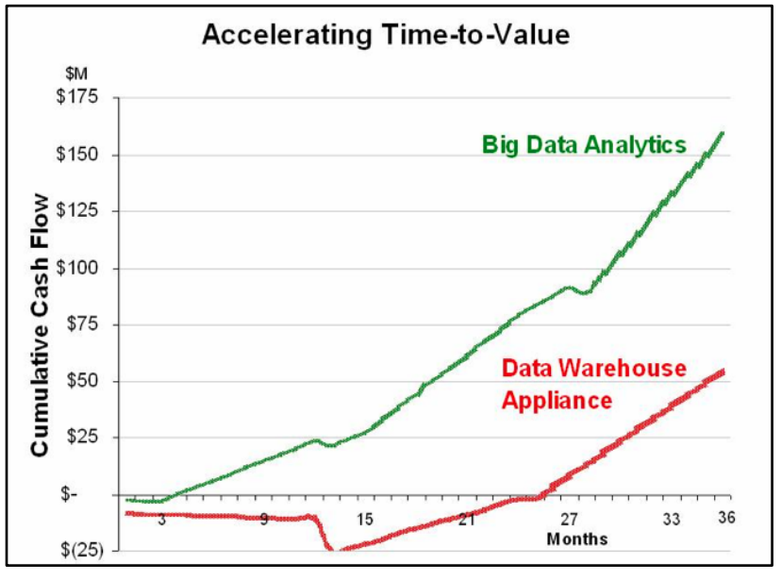


- Optimizations and predictive analytics
- Complex statistical analysis
- All types of data, and many sources
- Very large datasets
- More of a real-time

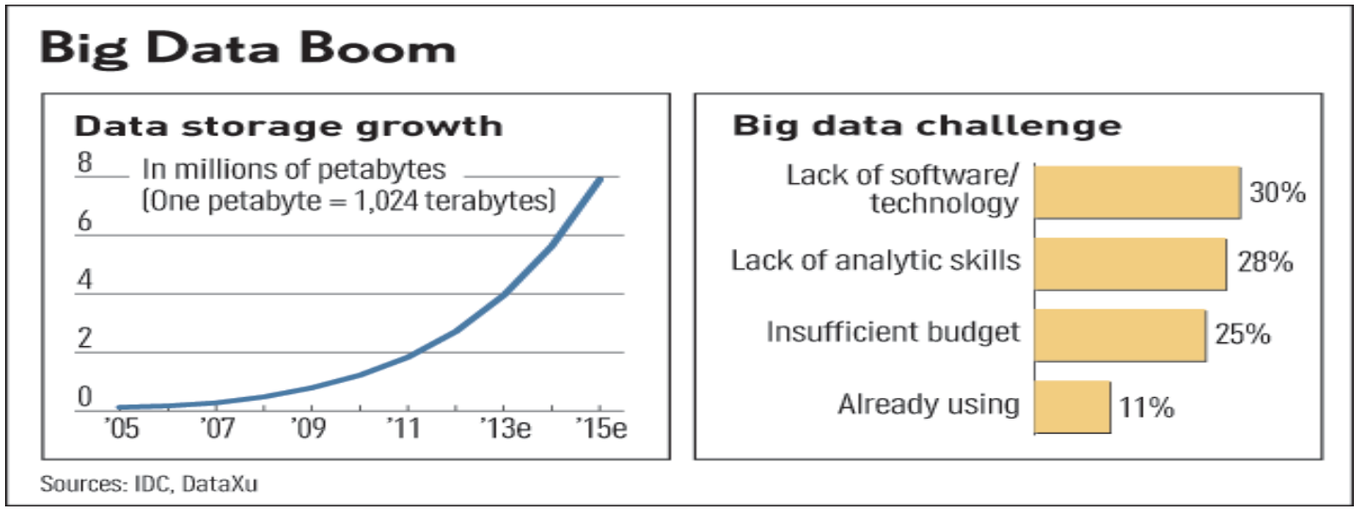
- Ad-hoc querying and reporting
- Data mining techniques
- Structured data, typical sources
- Small to mid-size datasets

Value of Big Data Analytics

- Big data is more real-time in nature than traditional DW applications
- Traditional DW architectures (e.g. Exadata, Teradata) are not well-suited for big data apps
- Shared nothing, massively parallel processing, scale out architectures are well-suited for big data apps



Challenges in Handling Big Data



- **The Bottleneck is in technology**
  - New architecture, algorithms, techniques are needed
- **Also in technical skills**
  - Experts in using the new technology and dealing with big data

What Technology Do We Have For Big Data ??

Big Data Landscape

Vertical Apps

PREDICTIVE POLICING
bloomreach
MYRRIX

Log Data Apps

splunk> loggly sumologic

Ad/Media Apps

rocketfuel
collective[i]
bluefin
Recorded Future
LuckySort
Media Science
TURN
DataXu

Business Intelligence

ORACLE | Hyperion
SAP BusinessObjects RJMetrics
Microsoft | Business Intelligence
IBM COGNOS birst
Autonomy MicroStrategy
QlikView bime
Chart.io GoodData

Analytics and Visualization

tableau Palantir
OPERA metaLayer
METAMARKETS dataspora
TERADATA ASTER centrifuge
SAS TIBCO KARMASPHERE
panopticon pentaho
Datameer
platfora ClearStory CIRRO
alteryx visual.ly AYATA

Data As A Service

factual
kaggle
knoema
GNIP DATASIFT Windows Azure INRIX LexisNexis LOQATE

Analytics Infrastructure

Hortonworks VERTICA MAPR
cloudera INFOBRIGHT PARACCEL
EMC² GREENPLUM
NETEZZA kognitio
DATASTAX EXASOL

Operational Infrastructure

Couchbase 10gen
TERADATA HADAPT
TERRACOTTA VoltDB
MarkLogic INFORMATICA

Infrastructure As A Service

amazon web services
Windows Azure
infochimps
Google BigQuery

Structured Databases

ORACLE MySQL
Microsoft SQL Server PostgreSQL
IBM DB2 SYBASE
memsql

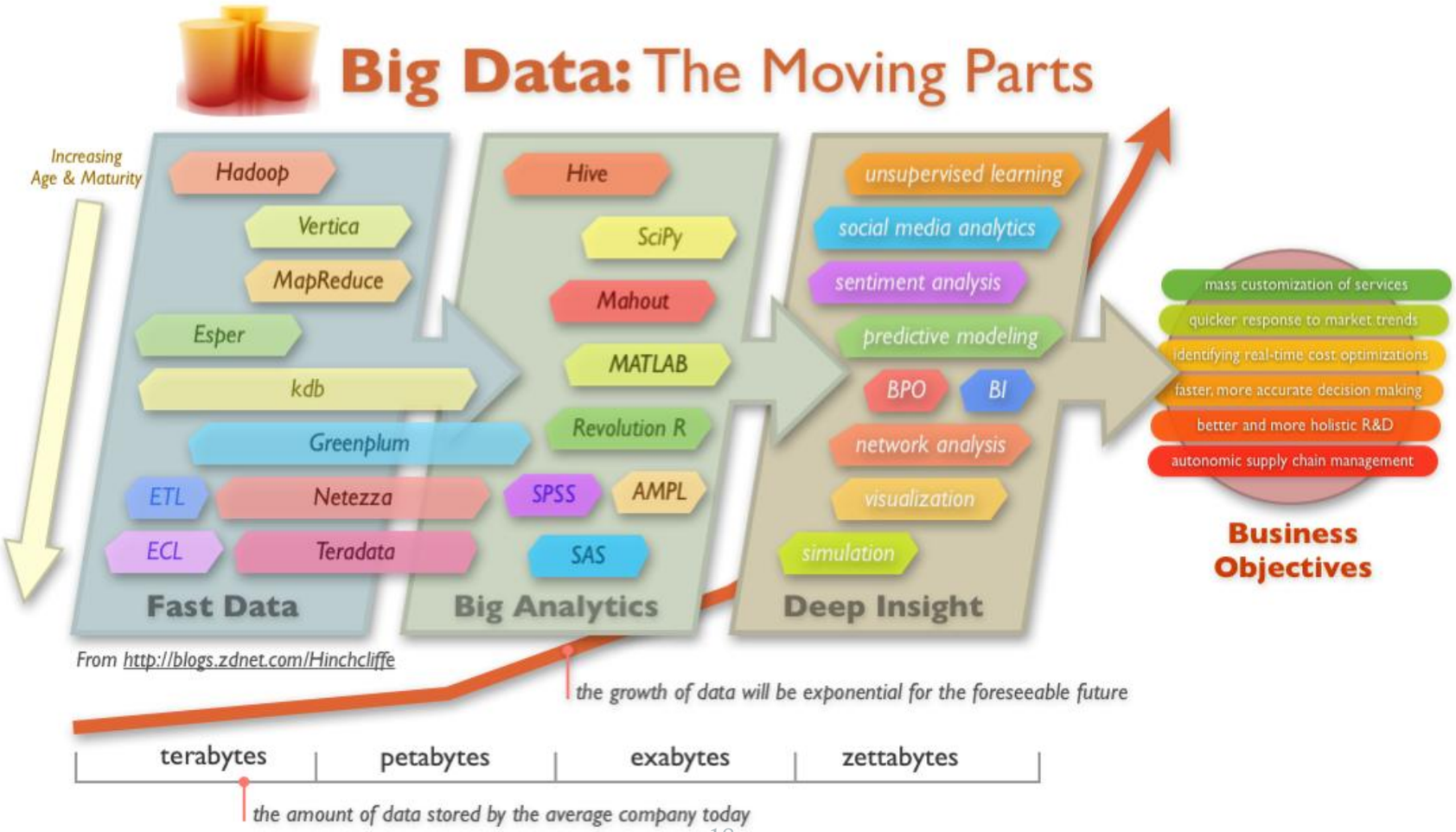
Technologies

hadoop
hadoop MapReduce
mahout
APACHE HBASE
Cassandra

dave@vcdave.com

blogs.forbes.com/davefeinleib

Big Data Technology



What You Will Learn…

- We focus on ***Hadoop / MapReduce technology***
- **Learn the platform (how it is designed and works)**
  - How big data are managed in a scalable, efficient way
- **Learn writing Hadoop jobs in different languages**
  - Programming Languages: Java, C, Python
  - High-Level Languages: Apache Pig, Hive
- **Learn advanced analytics tools on top of Hadoop**
  - RHadoop: Statistical tools for managing big data
  - Mahout: Data mining and machine learning tools over big data
- **Learn state-of-art technology from recent research papers**
  - Optimizations, indexing techniques, and other extensions to Hadoop

Course Logistics

Course Logistics

- Web Page: http://web.cs.wpi.edu/~cs525/s13-MYE/
- Electronic WPI system: blackboard.wpi.edu
- Lectures
  - Tuesday, Thursday: (4:00pm - 5:20pm)

Textbook & Reading List

- **No specific textbook**
  - Big Data is a relatively new topic (so no fixed syllabus)

- **Reading List**
  - We will cover the state-of-art technology from research papers in big conferences
  - Many Hadoop-related papers are available on the course website

- **Related books:**
  - Hadoop, The Definitive Guide [pdf]

Requirements & Grading

- **Seminar-Type Course**
  - Students will read research papers and present them (Reading List)
- **Hands-on Course**
  - No written homework or exams
  - Several coding projects covering the entire semester

Done in teams of two

Course grades are divided as follows:

Item	Percentage	Notes
Projects (6 or 7)	50%	Each project will be done in teams of two.
Presentations (1 or 2)	25%	Each presentation will be done in teams of two. If the number of teams is large, some teams may do one presentation + an extra project.
Reviews	15%	Reviews are done individually. Whenever a team is presenting a paper, other students are expected to read the presented paper and submit a review on it.
Class Participation	10%	Includes discussions in class and attendance.

Requirements & Grading (Cont'd)

- **Reviews**
  - When a team is presenting (*not the instructor*), the other students should prepare a review on the presented paper
  - Course website gives guidelines on how to make good reviews
  - ***Reviews are done individually***

Course grades are divided as follows:

Item	Percentage	Notes
Projects (6 or 7)	50%	Each project will be done in teams of two.
Presentations (1 or 2)	25%	Each presentation will be done in teams of two. If the number of teams is large, some teams may do one presentation + an extra project.
Reviews	15%	Reviews are done individually. Whenever a team is presenting a paper, other students are expected to read the presented paper and submit a review on it.
Class Participation	10%	Includes discussions in class and attendance.

Late Submission Policy

- **For Projects**
  - One-day late → 10% off the max grade
  - Two-day late → 20% off the max grade
  - Three-day late → 30% off the max grade
  - Beyond that, no late submission is accepted
  - **Submissions:**
    - Submitted via blackboard system by the due date
    - Demonstrated to the instructor within the week after
- **For Reviews**
  - No late submissions
  - Student may skip at most 4 reviews
  - **Submissions:**
    - Given to the instructor at the beginning of class

More about Projects

- **A virtual machine is created including the needed platform for the projects**
  - Ubuntu OS (Version 12.10)
  - Hadoop platform (Version 1.1.0)
  - Apache Pig (Version 0.10.0)
  - Mahout library (Version 0.7)
  - Rhadoop
  - In addition to other software packages
- **Download it from the course website (link)**
  - Username and password will be sent to you
- **Need Virtual Box (Vbox)** [free]

Next Step from You…

1. Form teams of two
2. Visit the course website (Reading List), each team selects its first paper to present (1st come 1st served)
   - Send me your choices top 2/3 choices
3. You have until Jan 20th
   - Otherwise, I'll randomly form teams and assign papers
4. Use Blackboard "Discussion" forum for posts or for searching for teammates

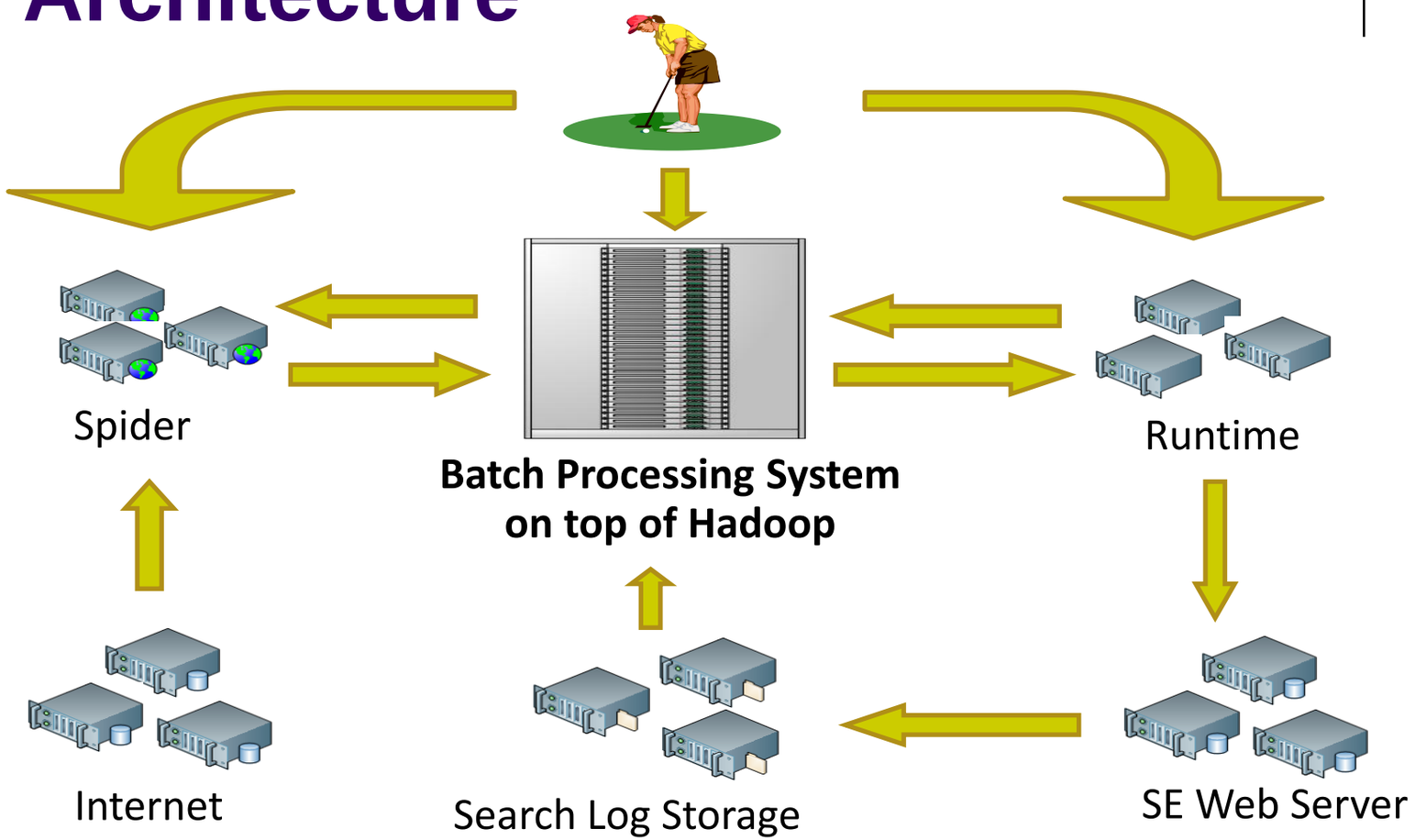
Course Output: What You Will Learn…

- We focus on ***Hadoop / MapReduce technology***
- **Learn the platform (how it is designed and works)**
  - How big data are managed in a scalable, efficient way
- **Learn writing Hadoop jobs in different languages**
  - Programming Languages: Java, C, Python
  - High-Level Languages: Apache Pig, Hive
- **Learn advanced analytics tools on top of Hadoop**
  - RHadoop: Statistical tools for managing big data
  - Mahout: Analytics and data mining tools over big data
- **Learn state-of-art technology from recent research papers**
  - Optimizations, indexing techniques, and other extensions to Hadoop

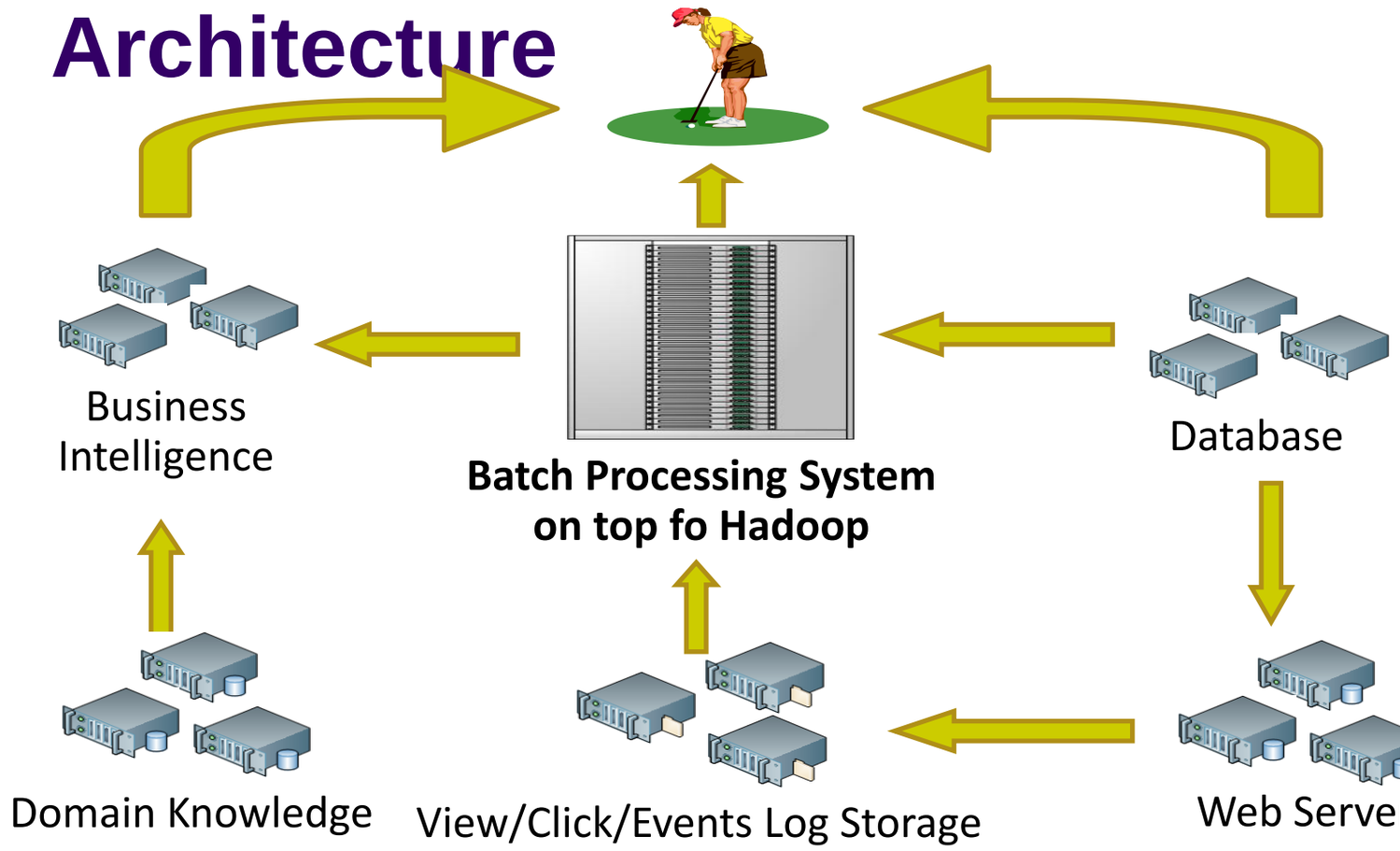
Open Source World's Solution

- **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

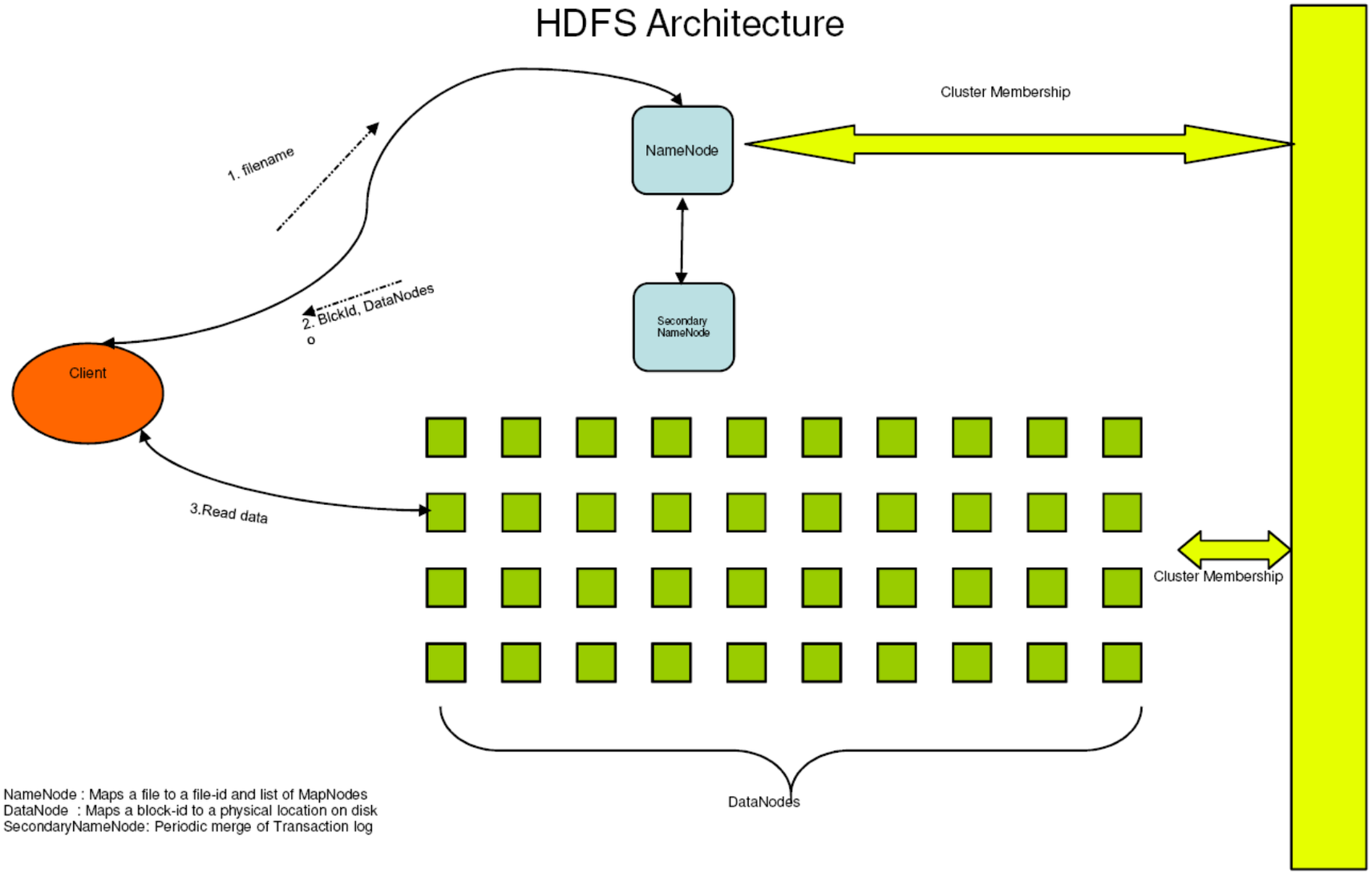
Hadoop Components

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

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

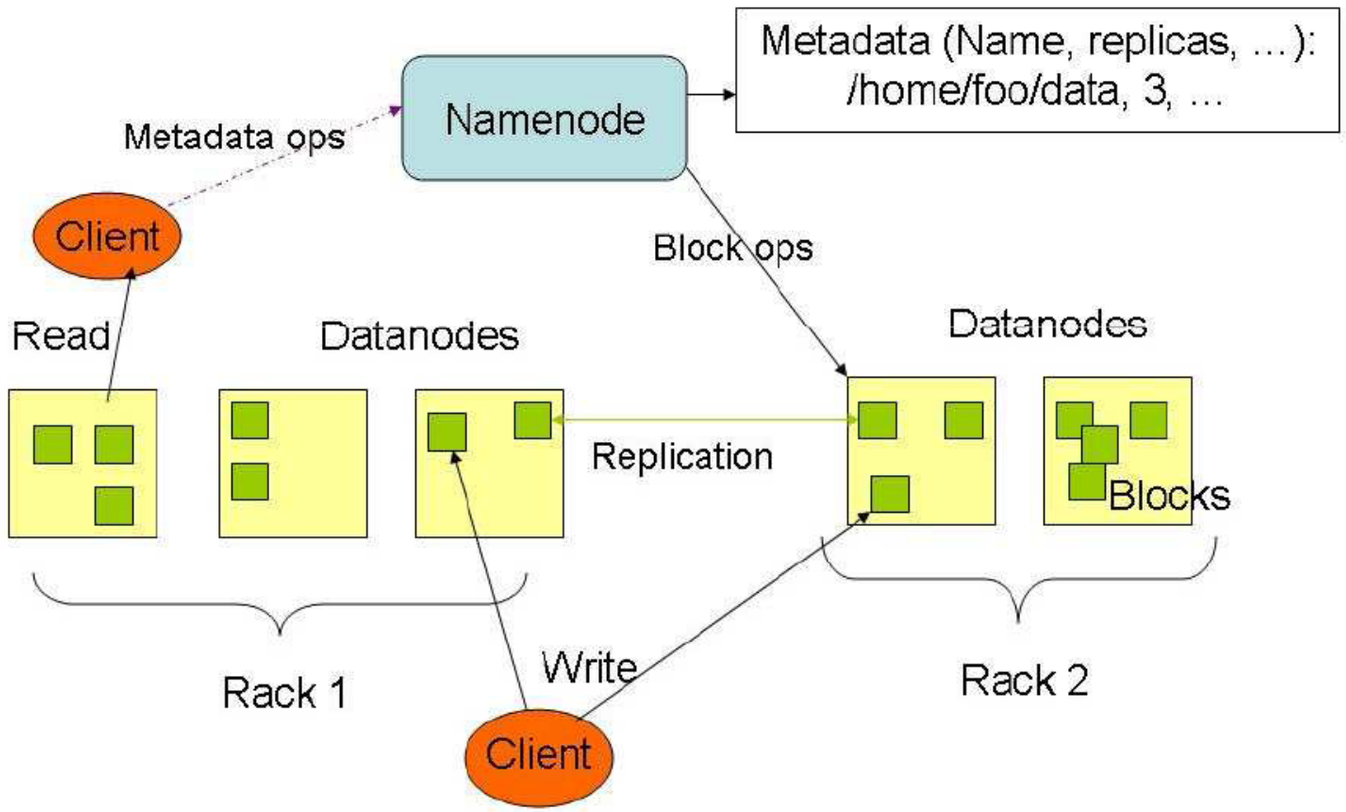


NameNode : Maps a file to a file-id and list of MapNodes
DataNode : Maps a block-id to a physical location on disk
SecondaryNameNode: Periodic merge of Transaction log

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

More about HDFS

- http://hadoop.apache.org/core/docs/current/hdfs_design.html

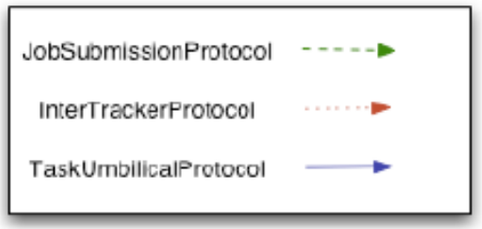
- Hadoop FileSystem API
  - HDFS
  - Local File System
  - Kosmos File System (KFS)
  - Amazon S3 File System

Hadoop Map-Reduce and Hadoop Streaming

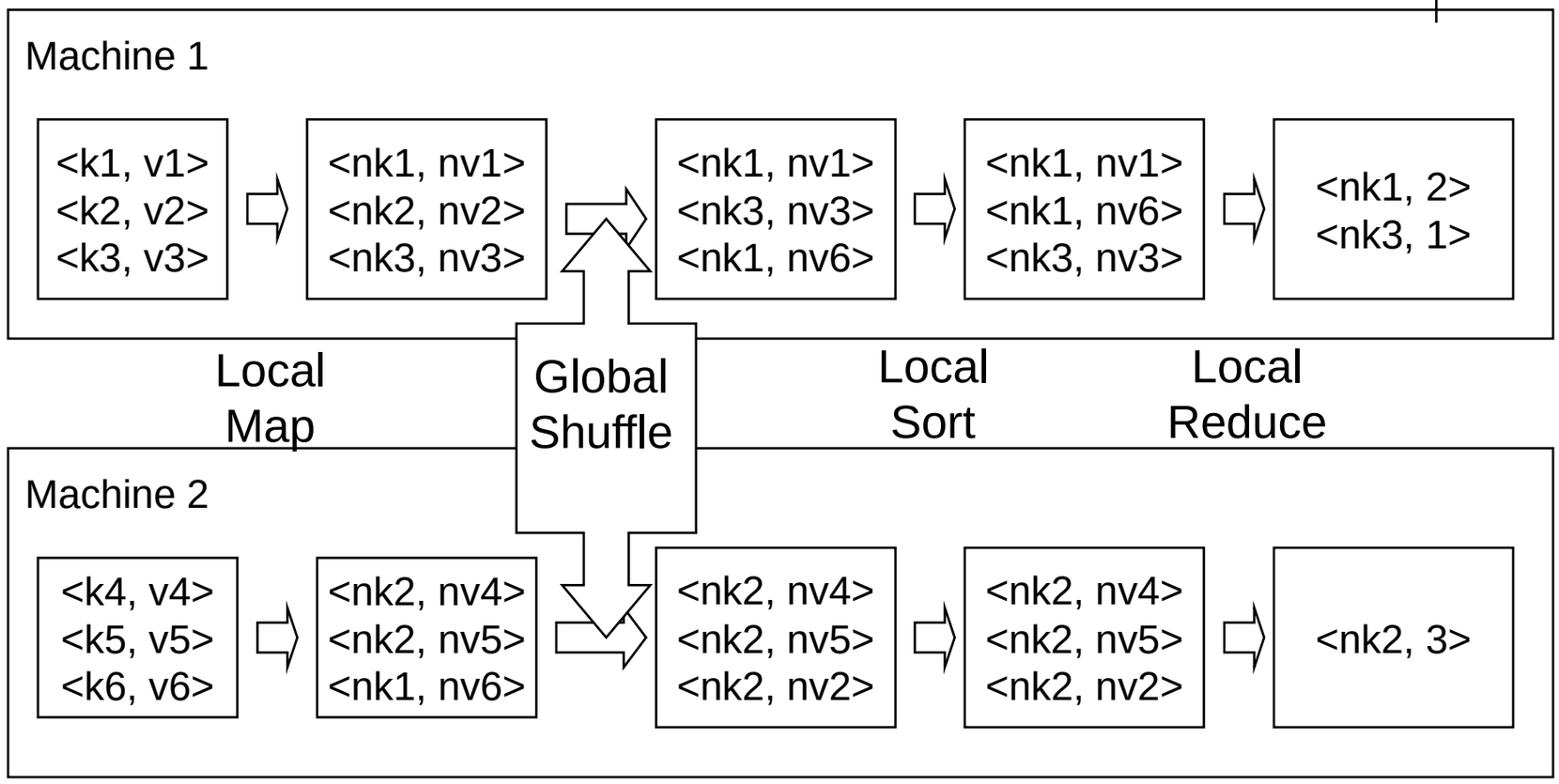
Hadoop Map-Reduce Introduction

- **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

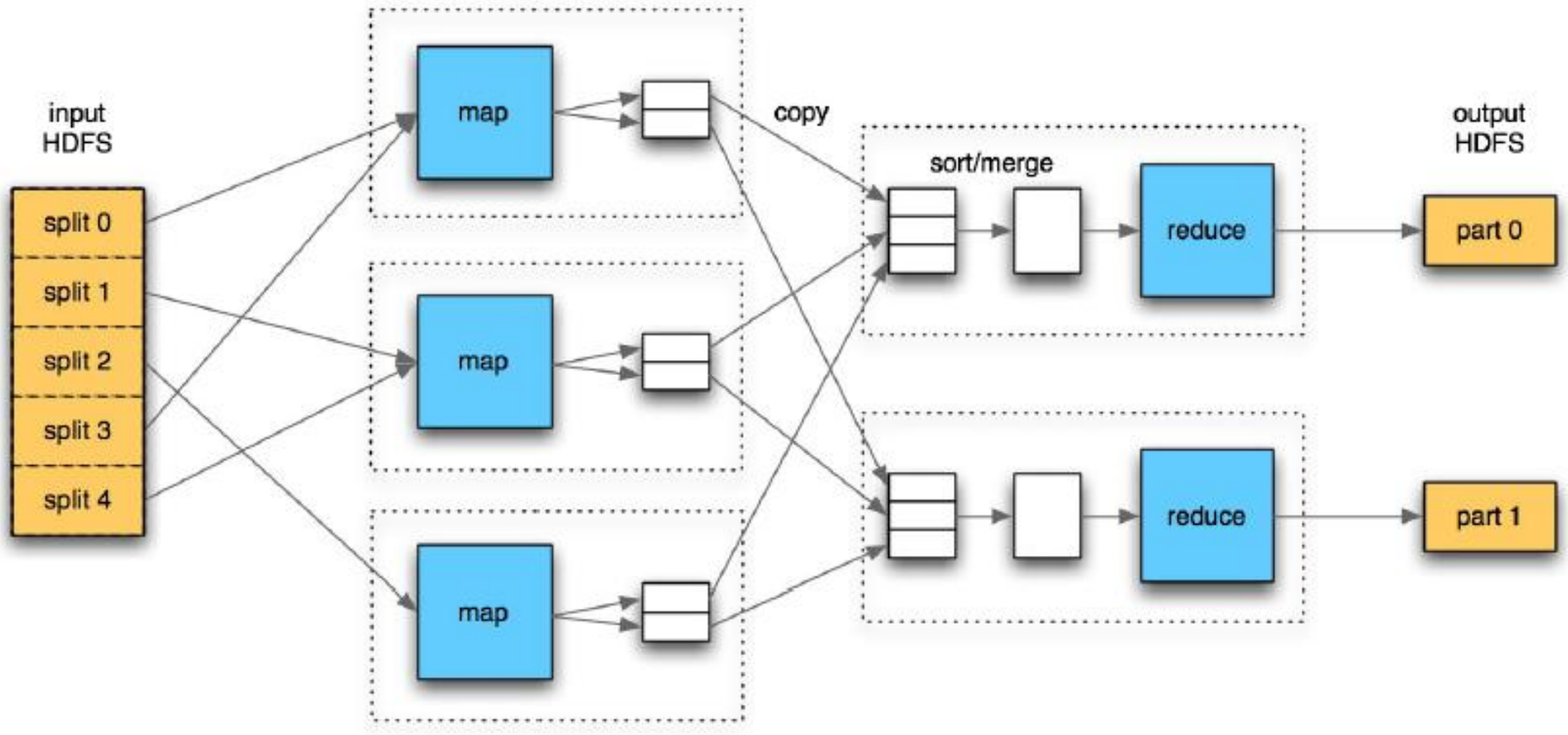
JobTracker
Launching Application
TaskTracker
TaskTracker
TaskTracker
Task
Task
Task
Task
Task


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
```

Example: Log Processing

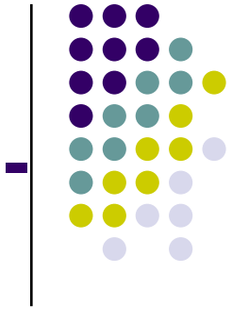
- **Generate #pageview and #distinct users for each page each day**
  - Input: timestamp url userid
- **Generate the number of page views**
  - Map: emit < <date(timestamp), url>, 1>
  - Reduce: add up the values for each row
- **Generate the number of distinct users**
  - Map: emit < <date(timestamp), url, userid>, 1>
  - Reduce: For the set of rows with the same <date(timestamp), url>, count the number of distinct users by “uniq –c"

Example: Page Rank

- **In each Map/Reduce Job:**
  - **Map:** emit <link, eigenvalue(url)/#links>
    for each input: <url, <eigenvalue, vector<link>> >
  - **Reduce:** add all values up for each link, to generate the new eigenvalue for that link.

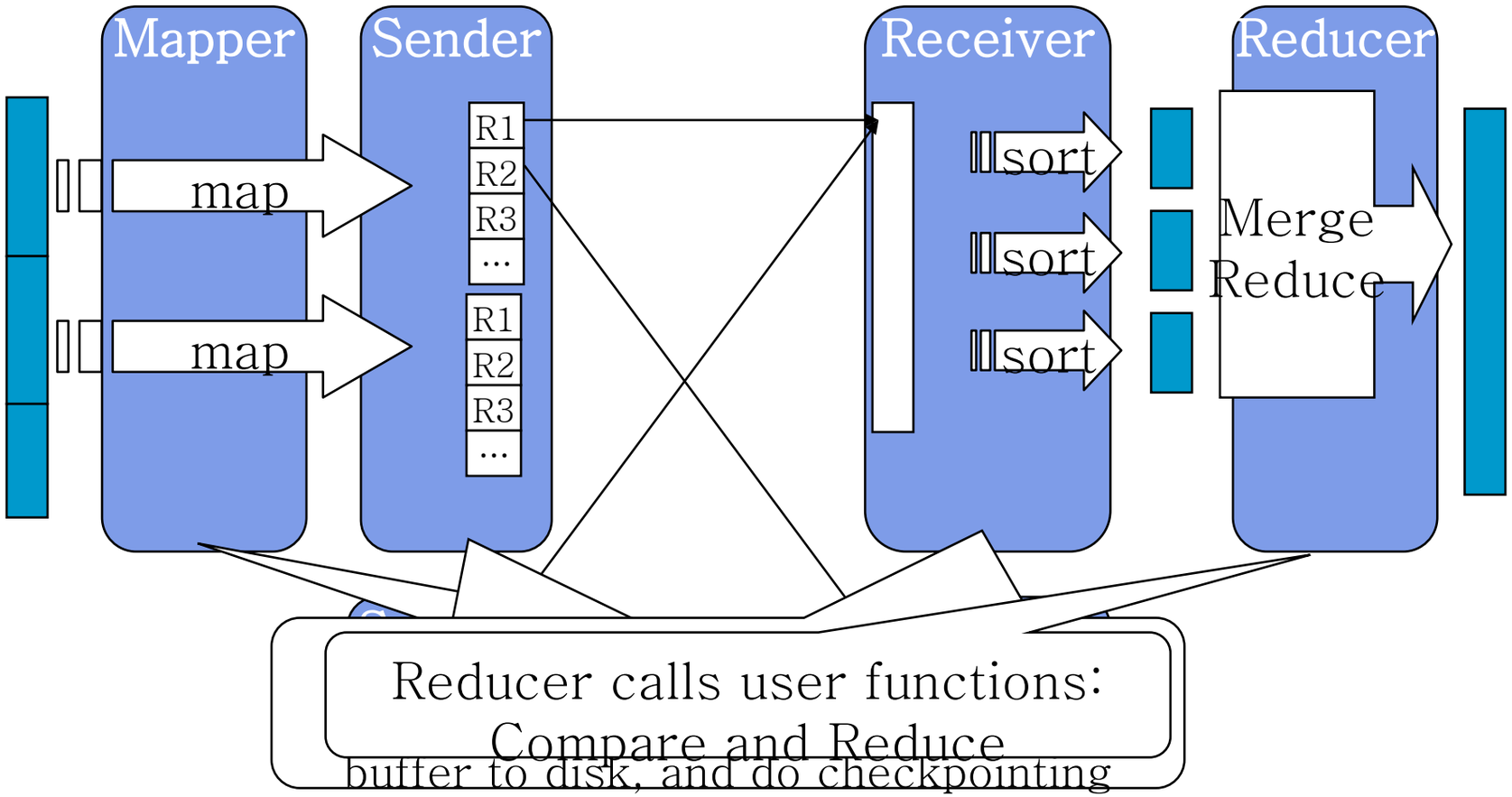
- Run 50 map/reduce jobs till the eigenvalues are stable.

TODO: Split Job Scheduler and Map-Reduce



- **Allow easy plug-in of different scheduling algorithms**
  - Scheduling based on job priority, size, etc
  - Scheduling for CPU, disk, memory, network bandwidth
  - Preemptive scheduling
- **Allow to run MPI or other jobs on the same cluster**
  - PageRank is best done with MPI

TODO: Faster Map-Reduce



MapReduce and Hadoop Distributed File System

The Context: Big-data

- Man on the moon with 32KB (1969); my laptop had 2GB RAM (2009)
- Google collects 270PB data in a month (2007), 20000PB a day (2008)
- 2010 census data is expected to be a huge gold mine of information
- Data mining huge amounts of data collected in a wide range of domains from astronomy to healthcare has become essential for planning and performance.
- We are in a knowledge economy.
  - Data is an important asset to any organization
  - Discovery of knowledge; Enabling discovery; annotation of data
- We are looking at newer
  - programming models, and
  - Supporting algorithms and data structures.
- NSF refers to it as “data-intensive computing” and industry calls it “big-data” and “cloud computing”

Purpose of this talk

- To provide a simple introduction to:
  - "The big-data computing" : An important advancement that has a potential to impact significantly the CS and undergraduate curriculum.
  - A programming model called MapReduce for processing "big-data"
  - A supporting file system called Hadoop Distributed File System (HDFS)
- To encourage educators to explore ways to infuse relevant concepts of this emerging area into their curriculum.

The Outline

- Introduction to MapReduce
- From CS Foundation to MapReduce
- MapReduce programming model
- Hadoop Distributed File System
- Relevance to Undergraduate Curriculum
- Demo (Internet access needed)
- Our experience with the framework
- Summary
- References

MapReduce

What is MapReduce?

- MapReduce is a programming model Google has used successfully is processing its "big-data" sets (~ 20000 peta bytes per day)
  - Users specify the computation in terms of a *map* and a *reduce* function,
  - Underlying runtime system automatically parallelizes the computation across large-scale clusters of machines, and
  - Underlying system also handles machine failures, efficient communications, and performance issues.

-- Reference: Dean, J. and Ghemawat, S. 2008. **MapReduce: simplified data processing on large clusters.** *Communication of ACM* 51, 1 (Jan. 2008), 107-113.

From CS Foundations to MapReduce

Consider a large data collection:

{web, weed, green, sun, moon, land, part, web, green,…}

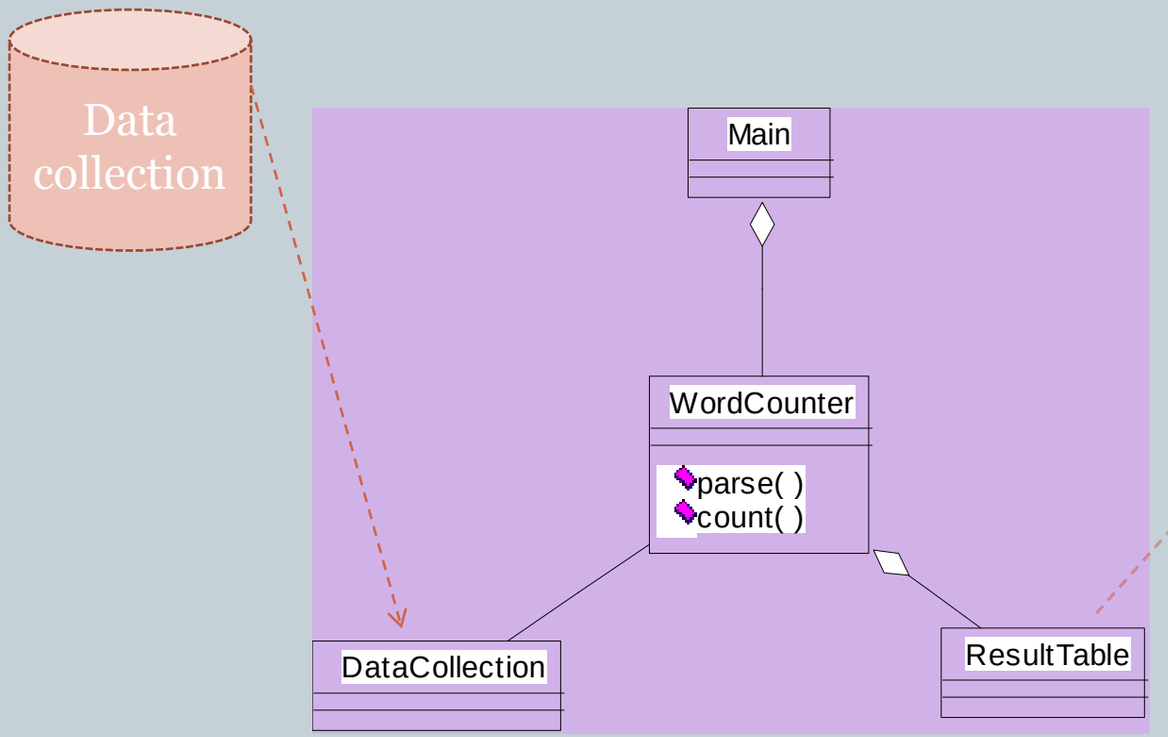
Problem: Count the occurrences of the different words in the collection.

Lets design a solution for this problem;

- We will start from scratch
- We will add and relax constraints
- We will do incremental design, improving the solution for performance and scalability

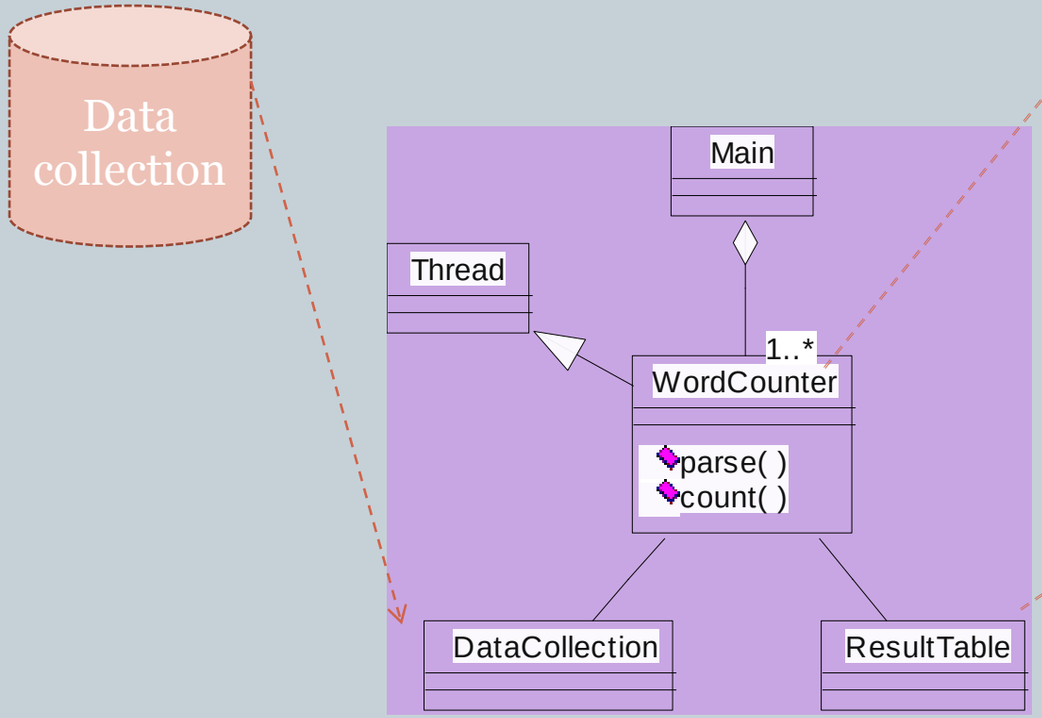
Word Counter and Result Table

{web, weed, green, sun, moon, land, part, web, green,...}



web	**2**
weed	**1**
green	**2**
sun	**1**
moon	**1**
land	**1**
part	**1**

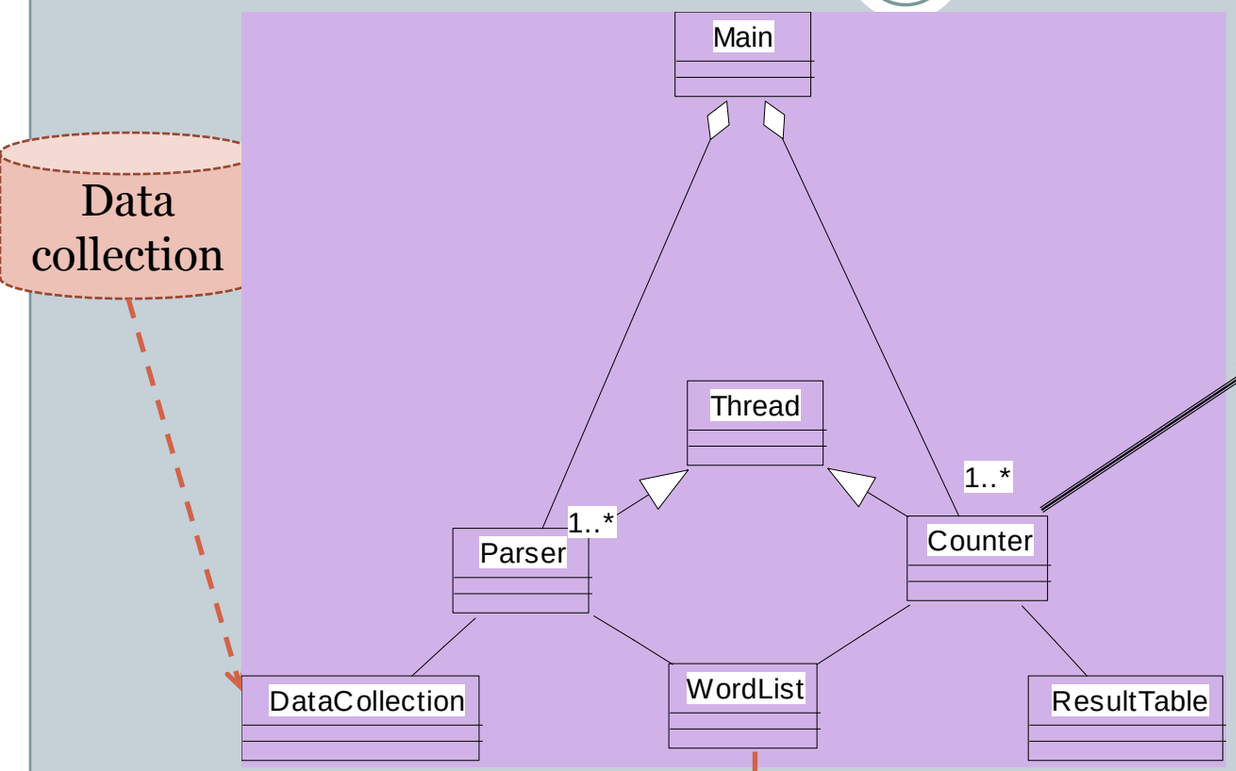
Multiple Instances of Word Counter



web	2
weed	1
green	2
sun	1
moon	1
land	1
part	1

Observe:
Multi-thread
Lock on shared data

Improve Word Counter for Performance



No need for lock

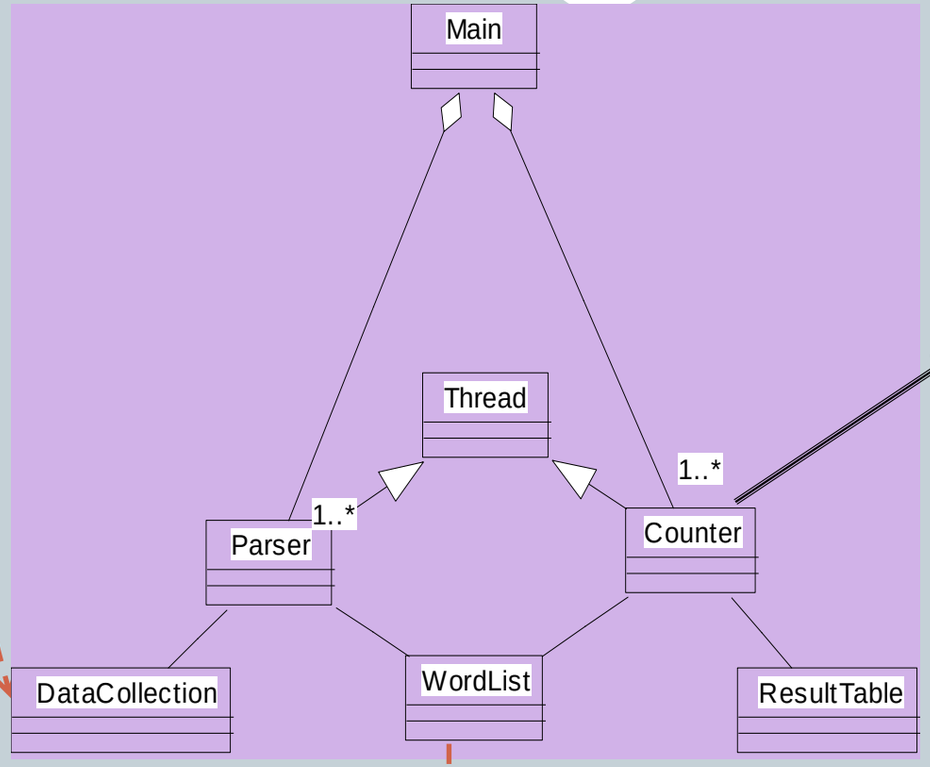
web	2
weed	1
green	2
sun	1
moon	1
land	1
part	1

Separate counters

KEY	web	weed	green	sun	moon	land	part	web	green	.....
VALUE										

Peta-scale Data

Data collection



web	2
weed	1
green	2
sun	1
moon	1
land	1
part	1

KEY	web	weed	green	sun	moon	land	part	web	green	.....
VALUE										

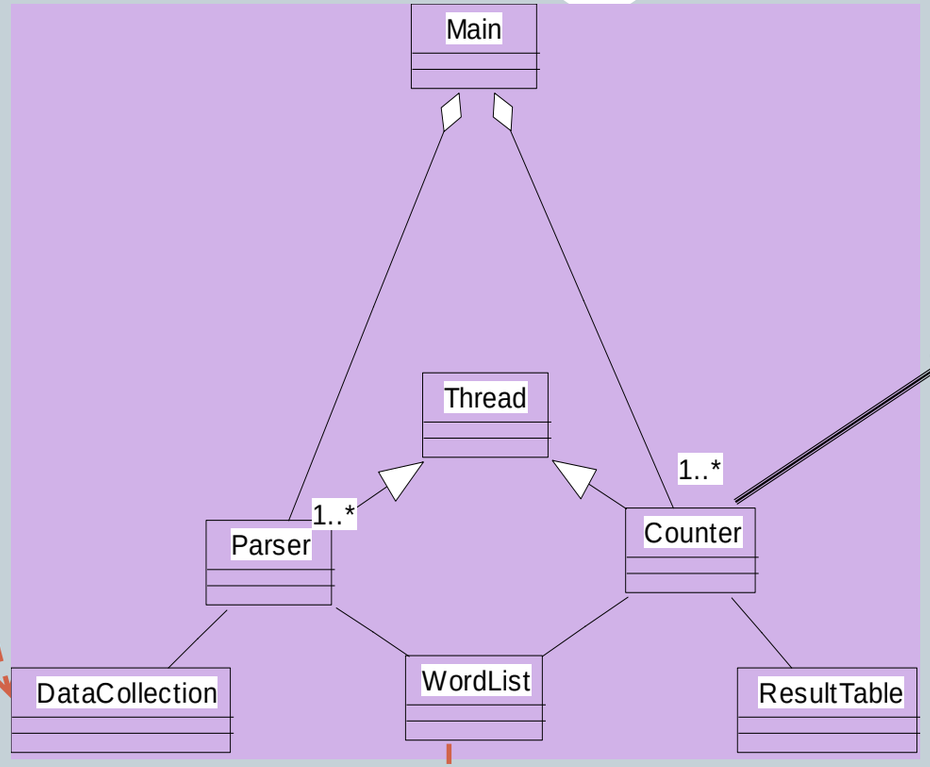
Addressing the Scale Issue

- Single machine cannot serve all the data: you need a distributed special (file) system
- Large number of commodity hardware disks: say, 1000 disks 1TB each
  - Issue: With Mean time between failures (MTBF) or failure rate of 1/1000, then at least 1 of the above 1000 disks would be down at a given time.
  - Thus failure is norm and not an exception.
  - File system has to be fault-tolerant: replication, checksum
  - Data transfer bandwidth is critical (location of data)

- Critical aspects: fault tolerance + replication + load balancing, monitoring
- Exploit parallelism afforded by splitting parsing and counting
- Provision and locate computing at data locations

Peta-scale Data

Data collection



web	2
weed	1
green	2
sun	1
moon	1
land	1
part	1

KEY	web	weed	green	sun	moon	land	part	web	green	.....
VALUE										

Peta Scale Data is Commonly Distributed

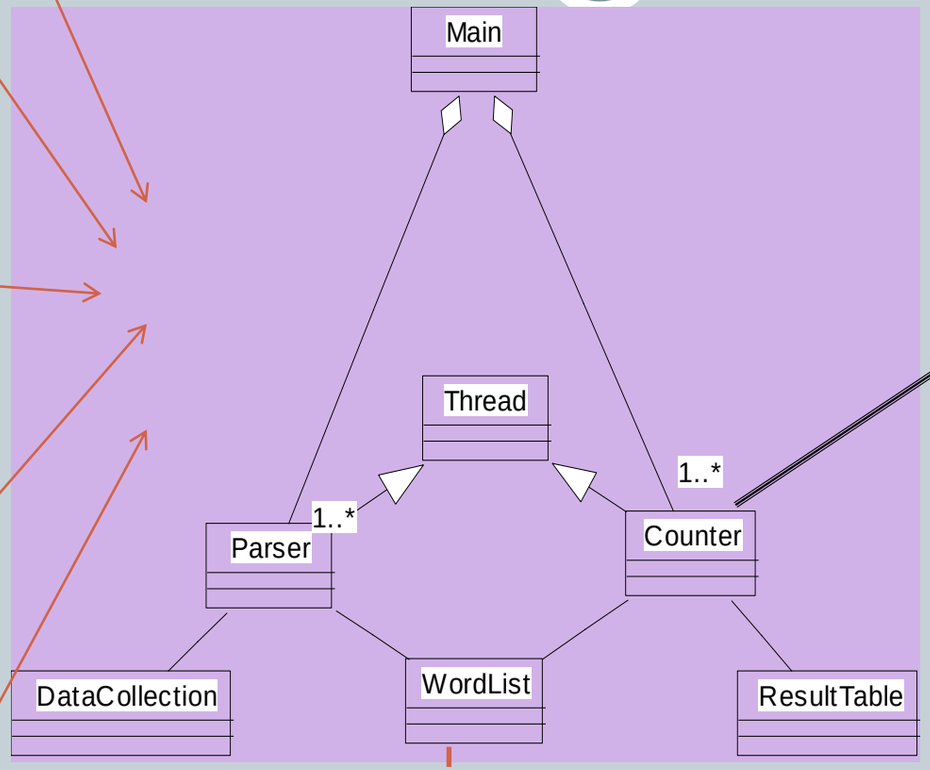
Data collection

Data collection

Data collection

Data collection

Data collection

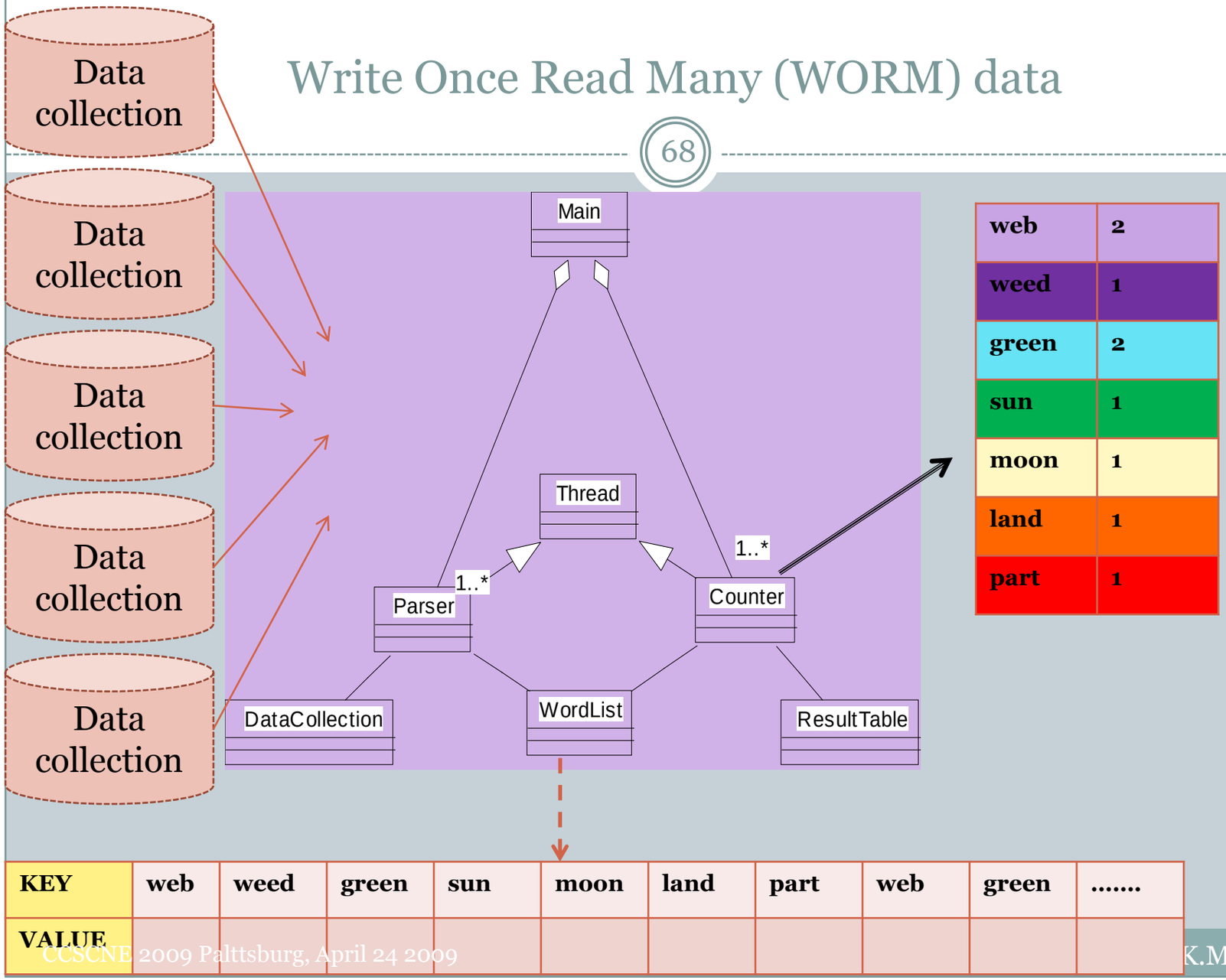


web	2
weed	1
green	2
sun	1
moon	1
land	1
part	1

Issue: managing the large scale data

KEY	web	weed	green	sun	moon	land	part	web	green	.....
VALUE										

Write Once Read Many (WORM) data



web	2
weed	1
green	2
sun	1
moon	1
land	1
part	1

KEY	web	weed	green	sun	moon	land	part	web	green	.....
VALUE										

WORM Data is Amenable to Parallelism

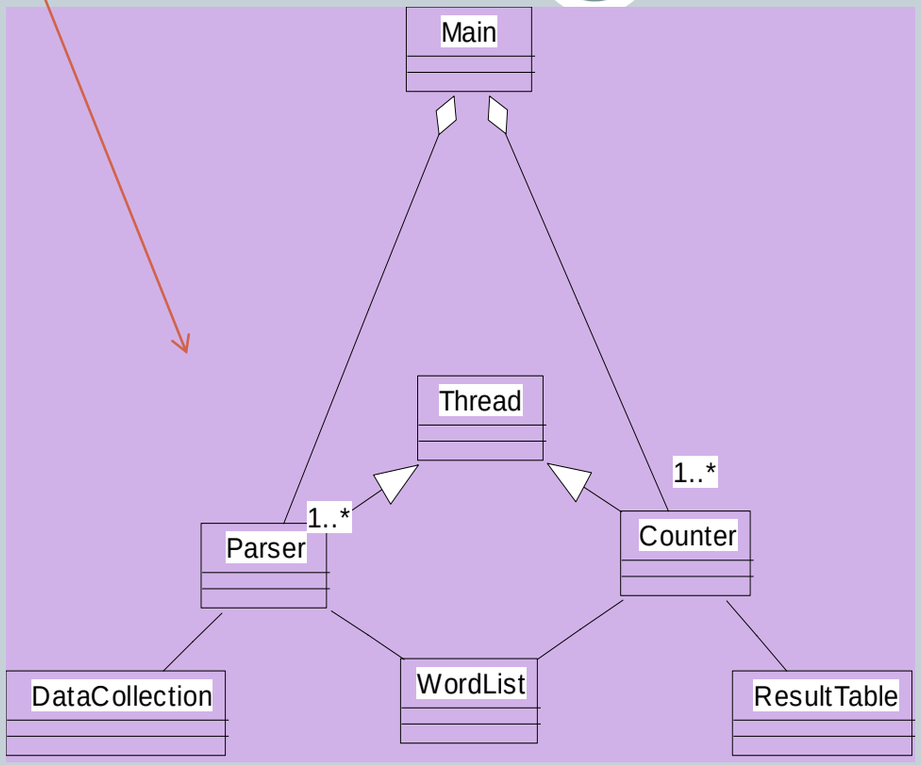
Data collection

Data collection

Data collection

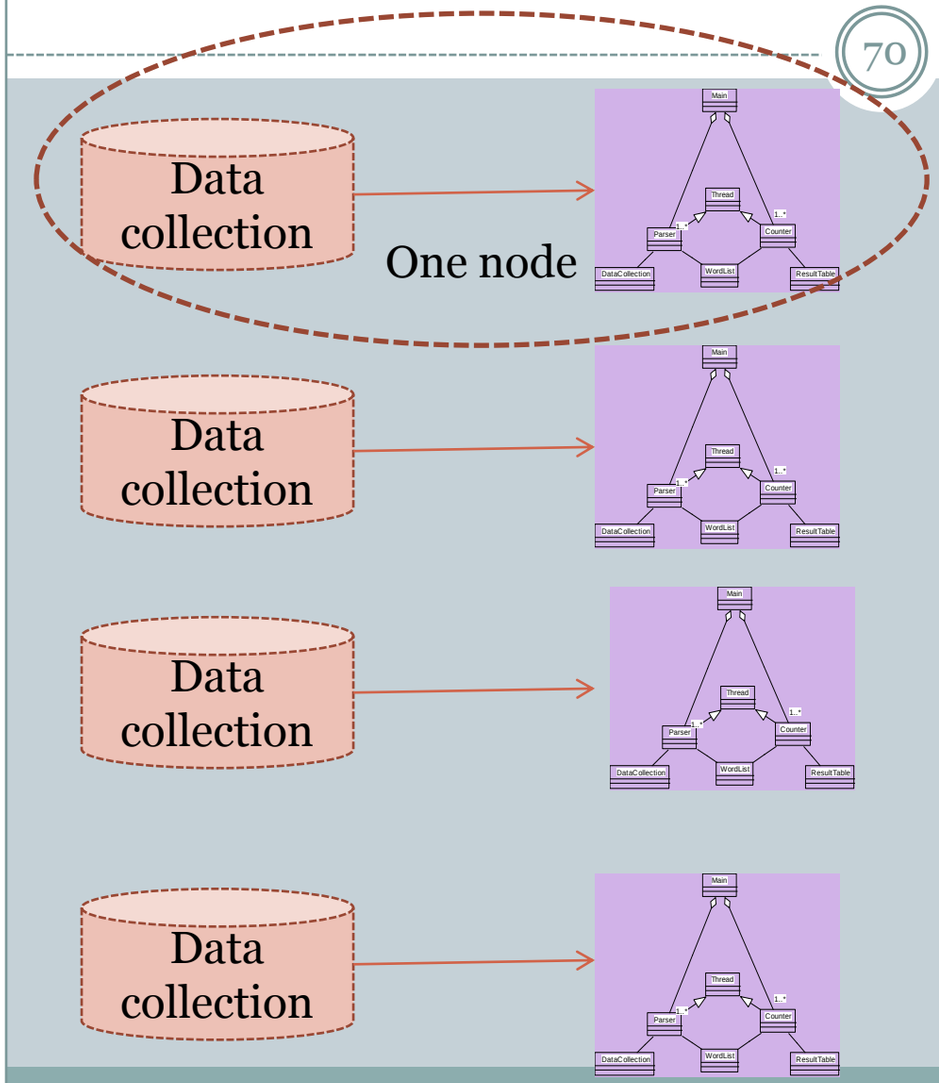
Data collection

Data collection



1. Data with WORM characteristics : yields to parallel processing;
2. Data without dependencies: yields to out of order processing

Divide and Conquer: Provision Computing at Data Location



For our example,
#1: Schedule parallel parse tasks
#2: Schedule parallel count tasks

This is a particular solution;
Lets generalize it:

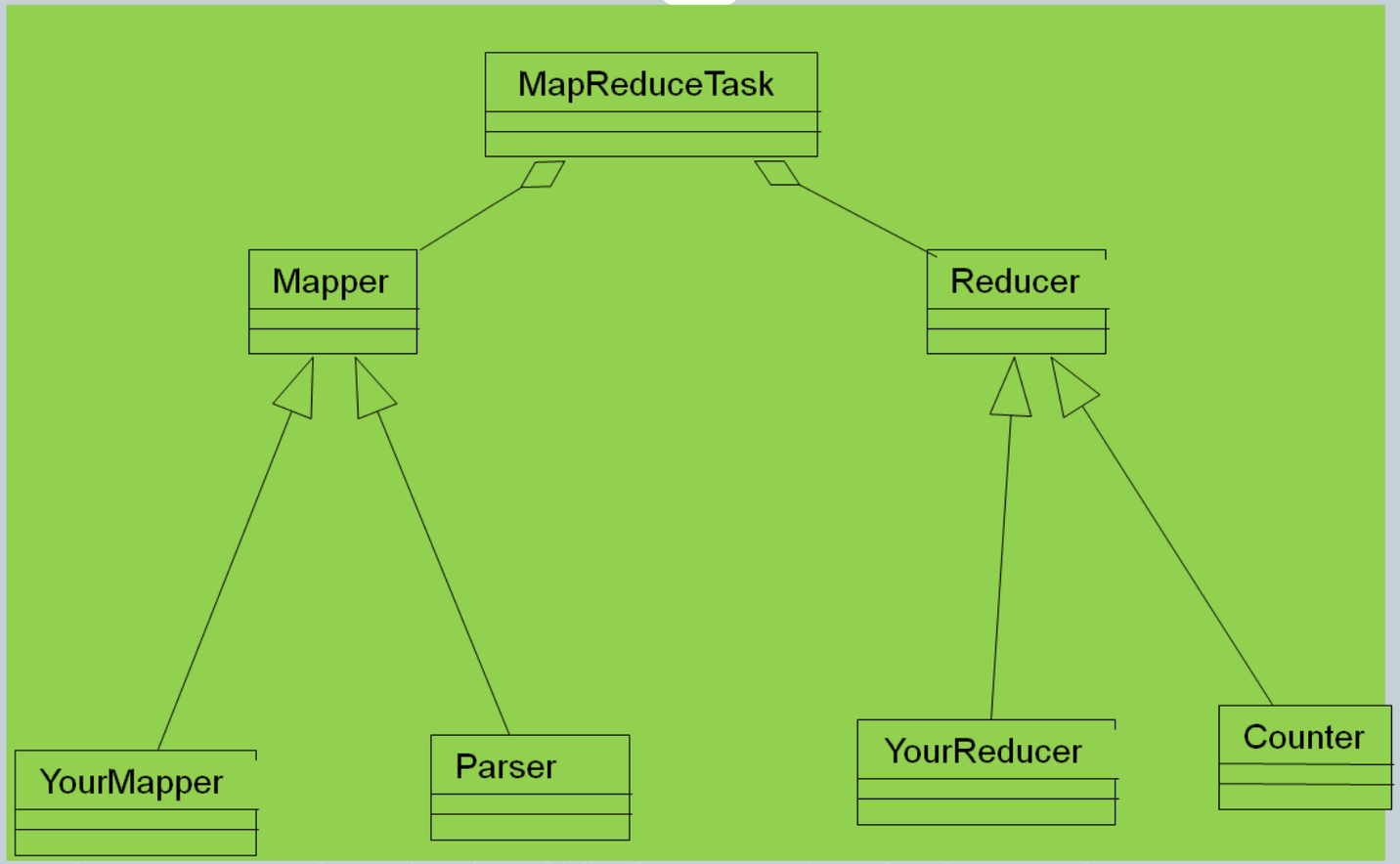
Our parse is a mapping operation:
MAP: input → <key, value> pairs

Our count is a reduce operation:
REDUCE: <key, value> pairs reduced

Map/Reduce originated from Lisp
But have different meaning here

Runtime adds distribution + fault tolerance + replication + monitoring + load balancing to your base application!

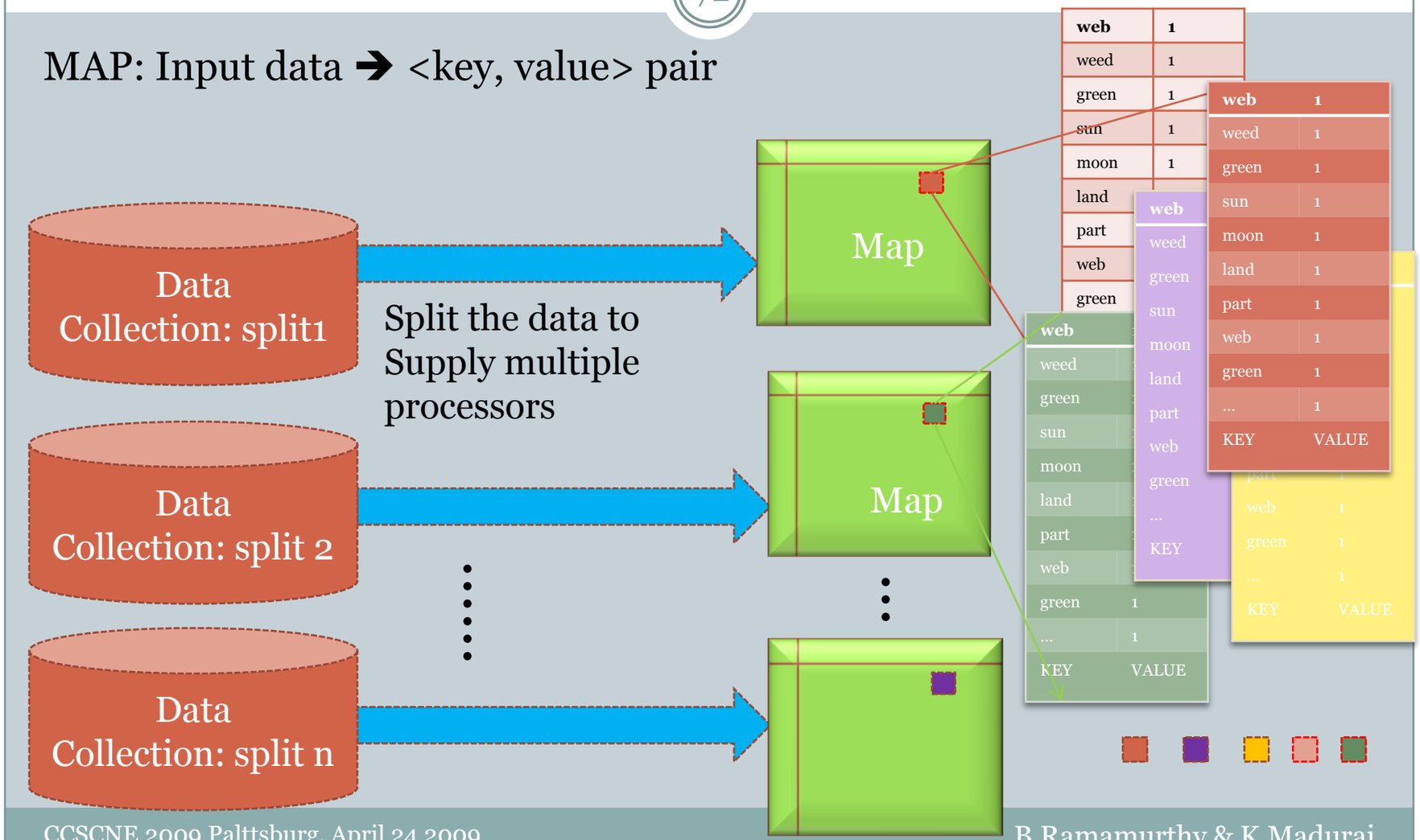
Mapper and Reducer



Remember: MapReduce is simplified processing for larger data sets:
MapReduce Version of WordCount Source code

Map Operation

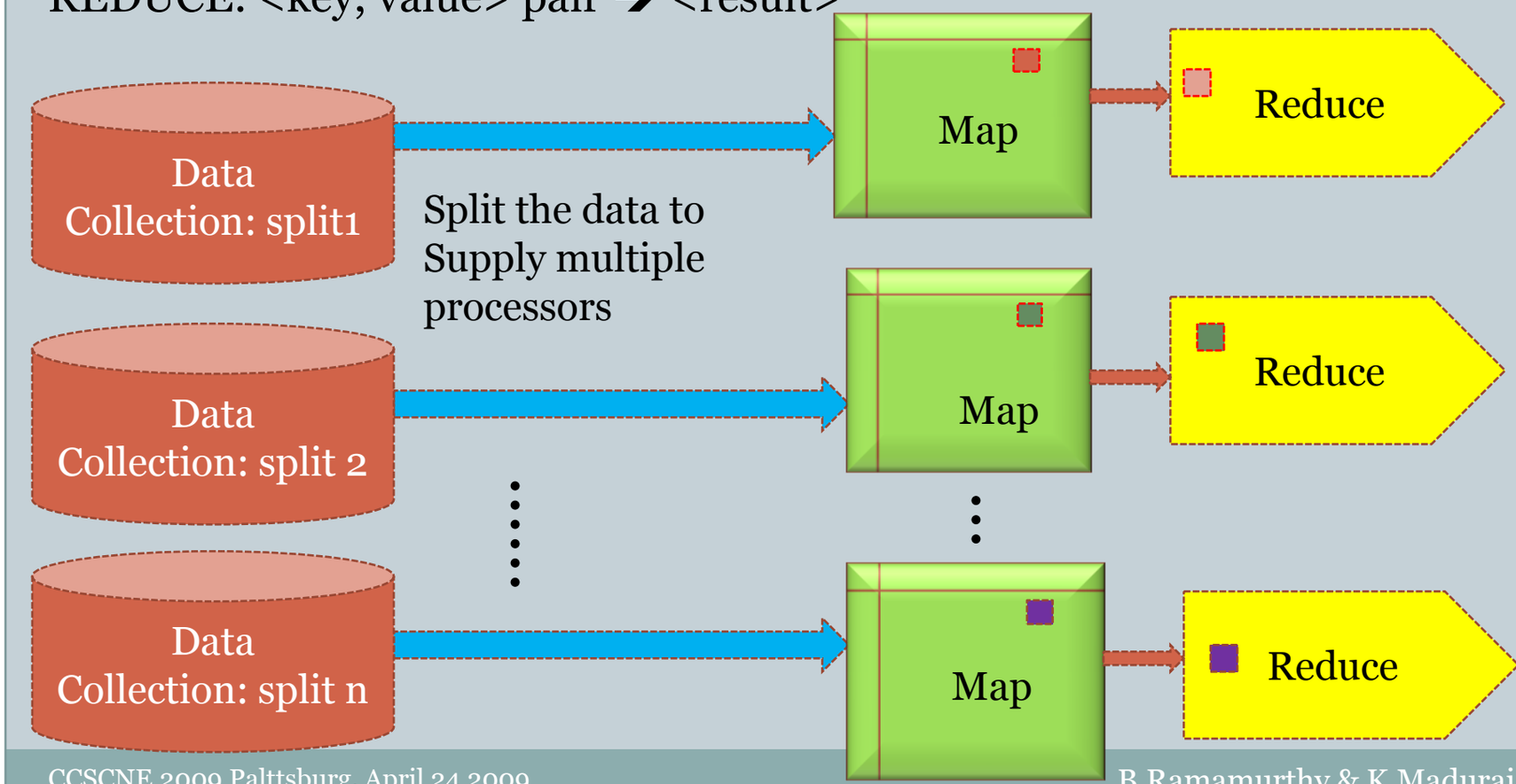
MAP: Input data ➔ <key, value> pair

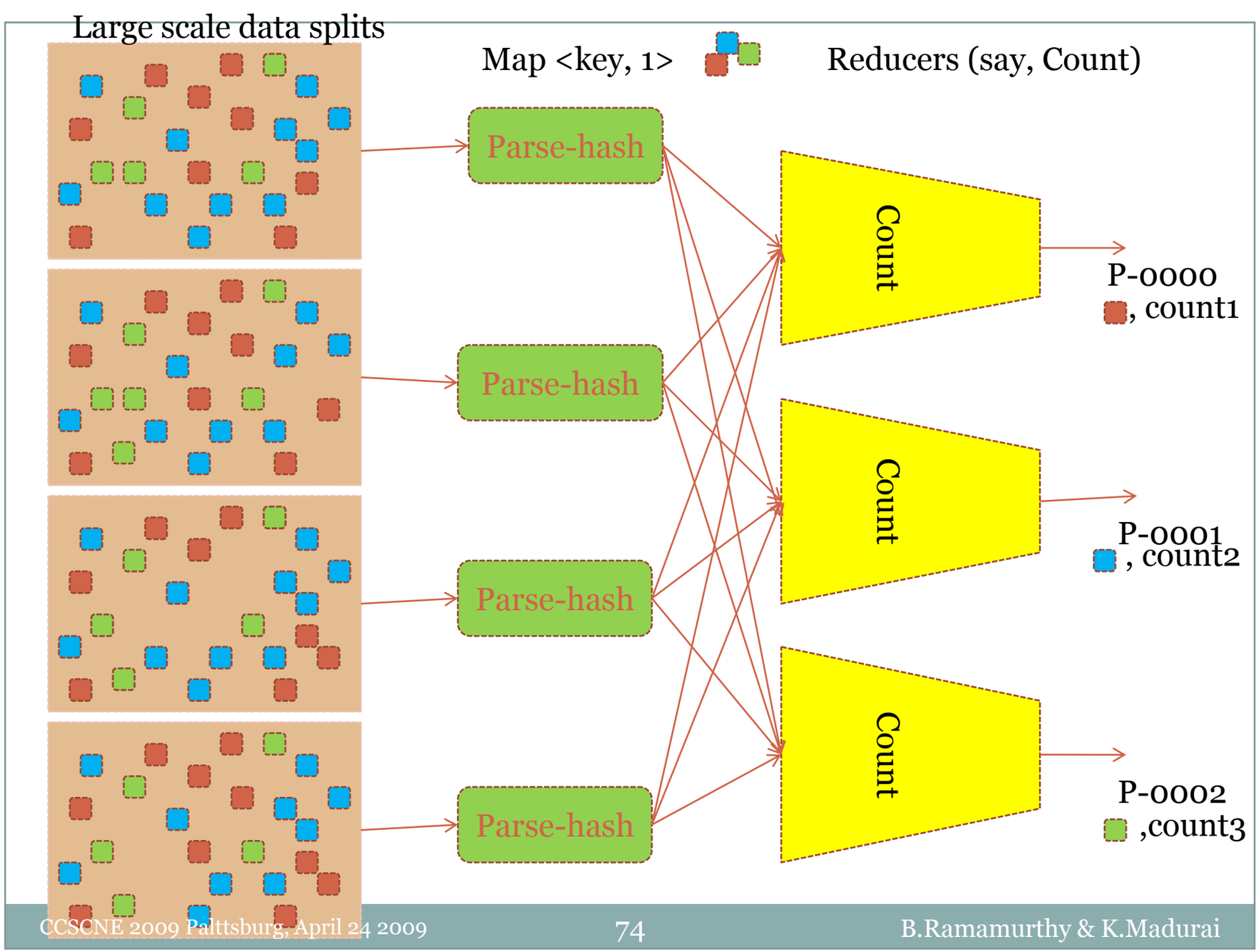


Reduce Operation

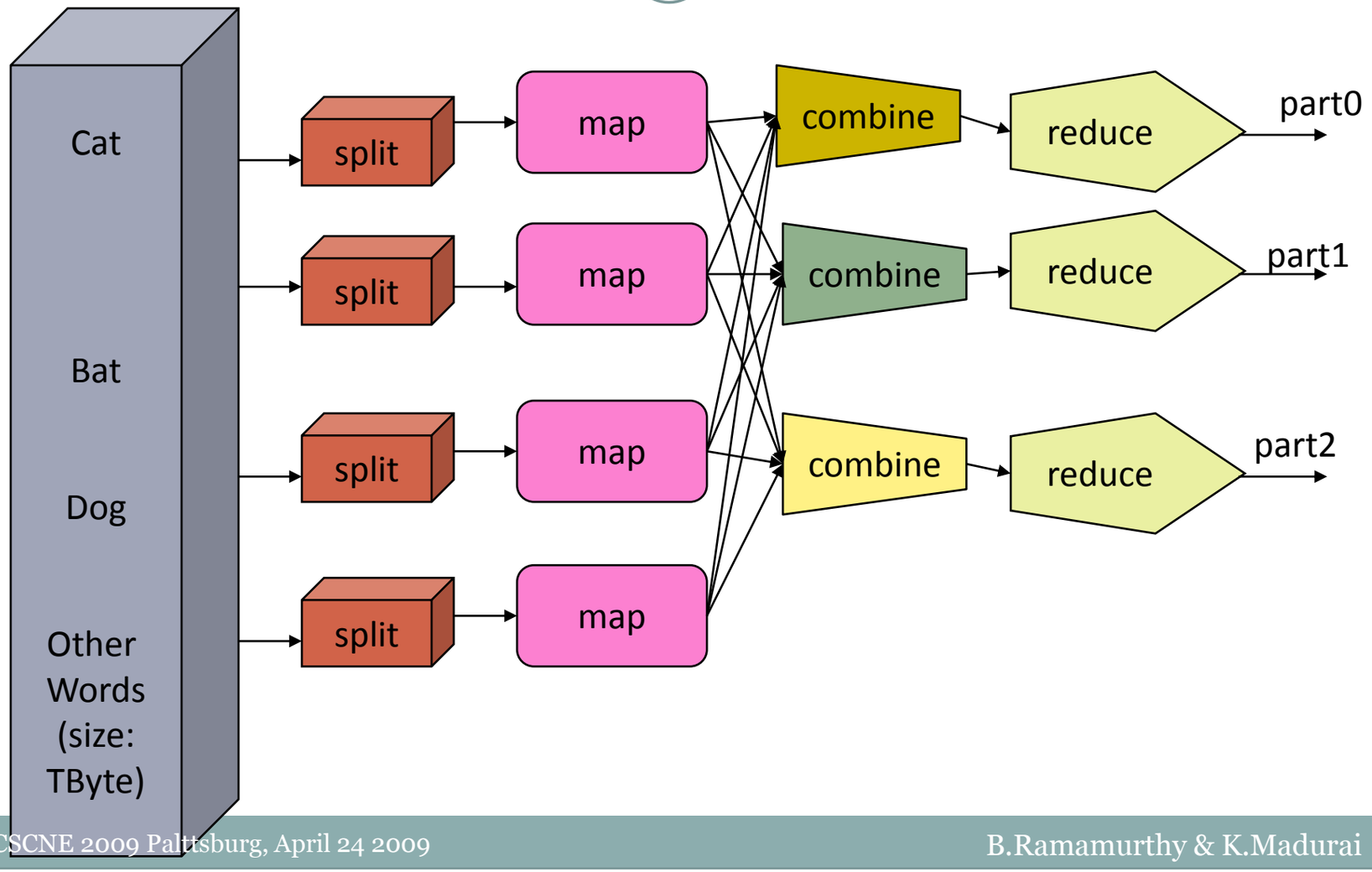
MAP: Input data ➔ <key, value> pair

REDUCE: <key, value> pair ➔ <result>




Large scale data splits
Map <key, 1>
Reducers (say, Count)
Parse-hash
Parse-hash
Parse-hash
Count
Count
Count
P-0000
, count1
P-0001
, count2
P-0002
,count3

MapReduce Example in my operating systems class



MapReduce Programming Model

MapReduce programming model

- Determine if the problem is parallelizable and solvable using MapReduce (ex: Is the data WORM?, large data set).
- Design and implement solution as Mapper classes and Reducer class.
- Compile the source code with hadoop core.
- Package the code as jar executable.
- Configure the application (job) as to the number of mappers and reducers (tasks), input and output streams
- Load the data (or use it on previously available data)
- Launch the job and monitor.
- Study the result.
- Detailed steps.

MapReduce Characteristics

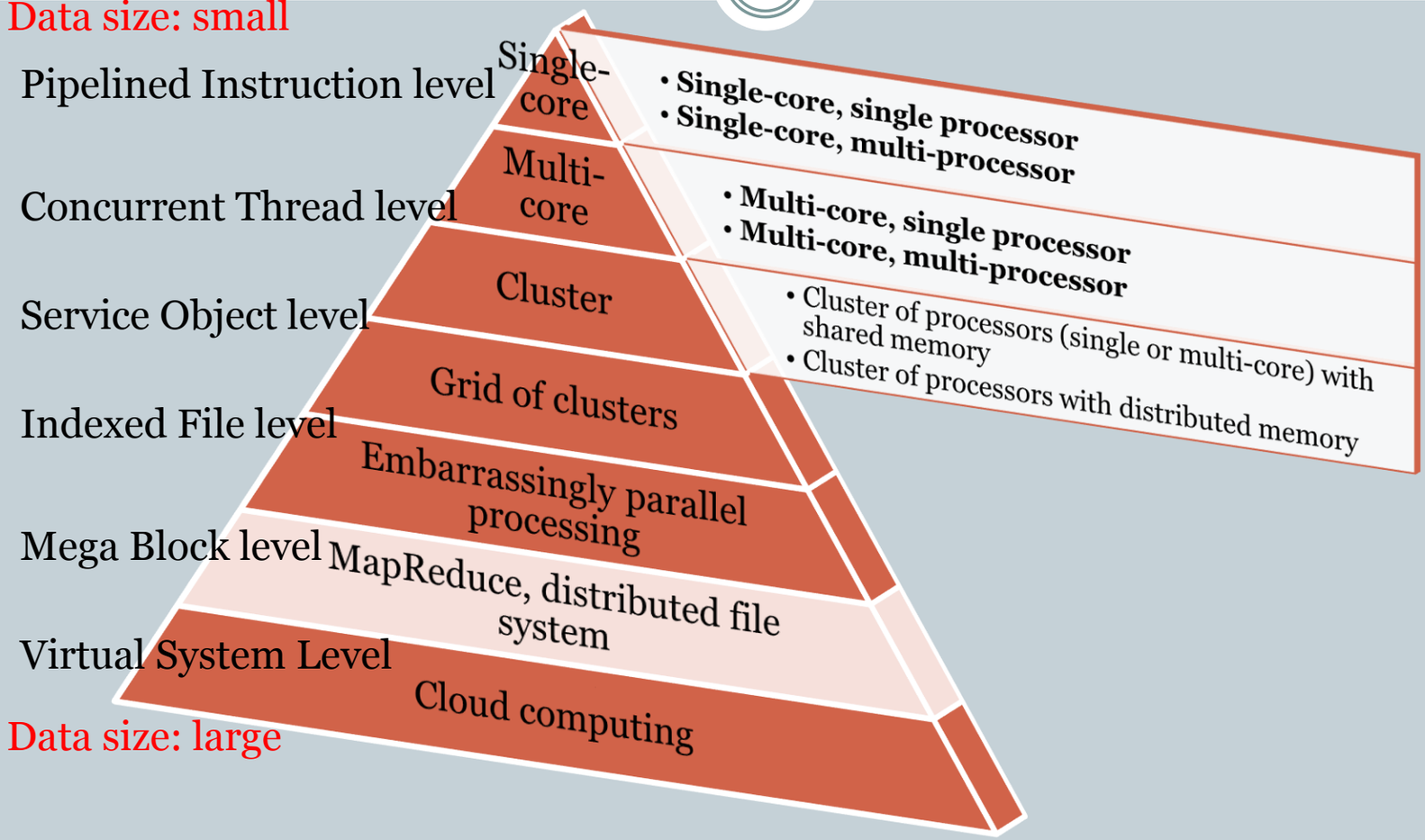
- Very large scale data: peta, exa bytes
- Write once and read many data: allows for parallelism without mutexes
- Map and Reduce are the main operations: simple code
- There are other supporting operations such as combine and partition (out of the scope of this talk).
- All the map should be completed before reduce operation starts.
- Map and reduce operations are typically performed by the same physical processor.
- Number of map tasks and reduce tasks are configurable.
- Operations are provisioned near the data.
- Commodity hardware and storage.
- Runtime takes care of splitting and moving data for operations.
- Special distributed file system. Example: Hadoop Distributed File System and Hadoop Runtime.

Classes of problems “mapreducable”

- Benchmark for comparing: Jim Gray’s challenge on data-intensive computing. Ex: “Sort”
- Google uses it (we think) for wordcount, adwords, pagerank, indexing data.
- Simple algorithms such as grep, text-indexing, reverse indexing
- Bayesian classification: data mining domain
- Facebook uses it for various operations: demographics
- Financial services use it for analytics
- Astronomy: Gaussian analysis for locating extra-terrestrial objects.
- Expected to play a critical role in semantic web and web3.0

Scope of MapReduce

Data size: small



Data size: large

Hadoop

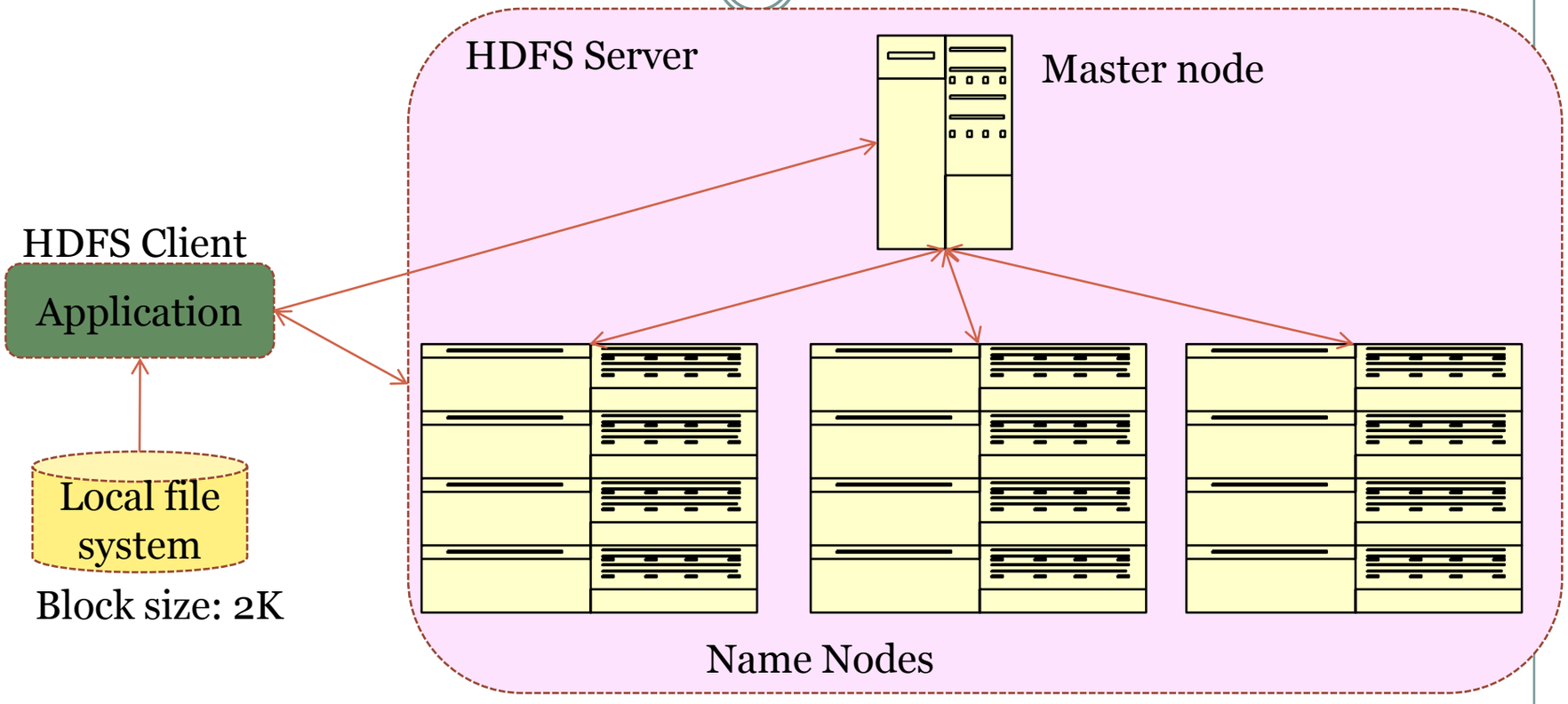
What is Hadoop?

- At Google MapReduce operation are run on a special file system called Google File System (GFS) that is highly optimized for this purpose.
- GFS is not open source.
- Doug Cutting and Yahoo! reverse engineered the GFS and called it Hadoop Distributed File System (HDFS).
- The software framework that supports HDFS, MapReduce and other related entities is called the project Hadoop or simply Hadoop.
- This is open source and distributed by Apache.

Basic Features: HDFS

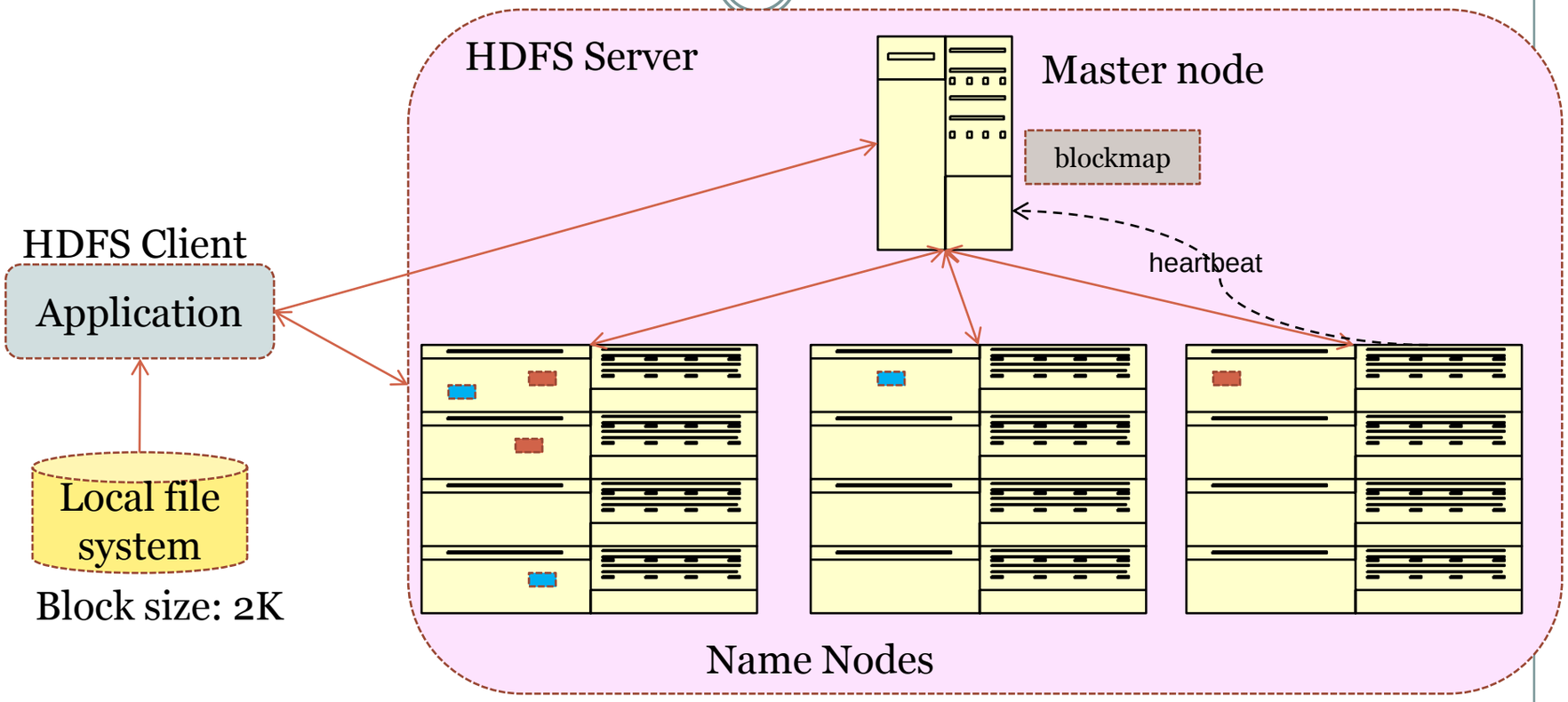
- Highly fault-tolerant
- High throughput
- Suitable for applications with large data sets
- Streaming access to file system data
- Can be built out of commodity hardware

Hadoop Distributed File System



More details: We discuss this in great detail in my Operating Systems course

Hadoop Distributed File System



More details: We discuss this in great detail in my Operating Systems course

Relevance and Impact on Undergraduate courses

- Data structures and algorithms: a new look at traditional algorithms such as sort: Quicksort may not be your choice! It is not easily parallelizable. Merge sort is better.
- You can identify mappers and reducers among your algorithms. Mappers and reducers are simply place holders for algorithms relevant for your applications.
- Large scale data and analytics are indeed concepts to reckon with similar to how we addressed "programming in the large" by OO concepts.
- While a full course on MR/HDFS may not be warranted, the concepts perhaps can be woven into most courses in our CS curriculum.

Demo

- VMware simulated Hadoop and MapReduce demo
- Remote access to NEXOS system at my Buffalo office
- 5-node HDFS running HDFS on Ubuntu 8.04
- 1 –name node and 4 data-nodes
- Each is an old commodity PC with 512 MB RAM, 120GB – 160GB external memory
- Zeus (namenode), datanodes: hermes, dionysus, aphrodite, athena

Summary

- We introduced MapReduce programming model for processing large scale data
- We discussed the supporting Hadoop Distributed File System
- The concepts were illustrated using a simple example
- We reviewed some important parts of the source code for the example.
- Relationship to Cloud Computing

References

1. Apache Hadoop Tutorial: http://hadoop.apache.org http://hadoop.apache.org/core/docs/current/mapred_tutorial.html
2. Dean, J. and Ghemawat, S. 2008. **MapReduce: simplified data processing on large clusters.** *Communication of ACM* 51, 1 (Jan. 2008), 107-113.
3. Cloudera Videos by Aaron Kimball: http://www.cloudera.com/hadoop-training-basic
4. http://www.cse.buffalo.edu/faculty/bina/mapreduce.html

Hive - SQL on top of Hadoop

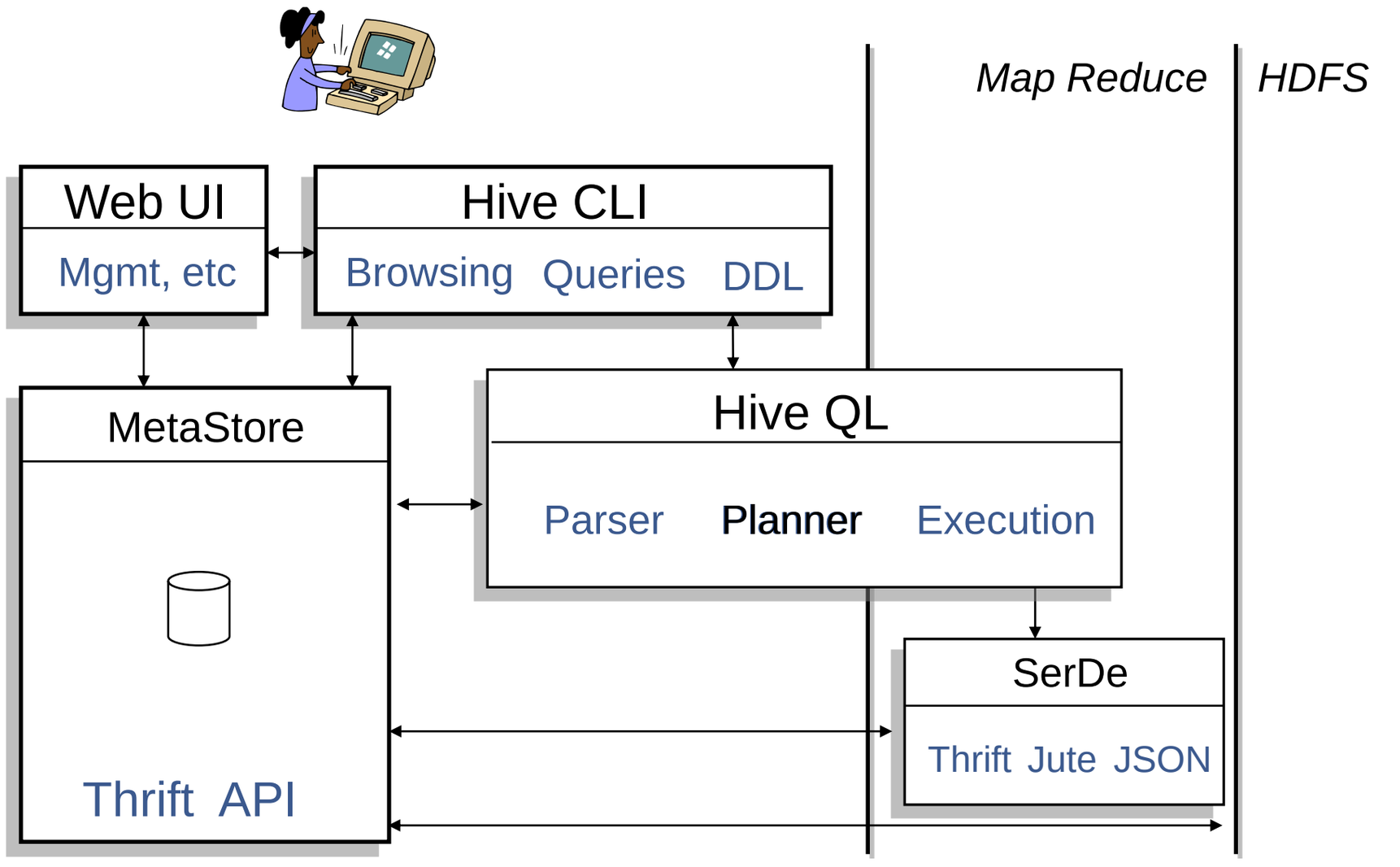
Map-Reduce and SQL

- **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/**

Hive Architecture



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

Map

key	value
111	<**1,**1>
111	<**1,**2>
222	<**1,**1>
111	<**2,**25>
222	<**2,**32>

Shuffle Sort

key	value
111	<**1,**1>
111	<**1,**2>
111	<**2,**25>
222	<**1,**1>
222	<**2,**32>

Reduce

pv_users

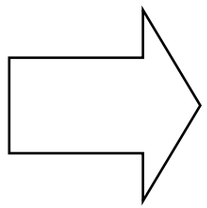
pageid	age
1	25
2	25

pageid	age
1	32

Hive QL – Group By

pv_users

pageid	age
1	25
2	25
1	32
2	25

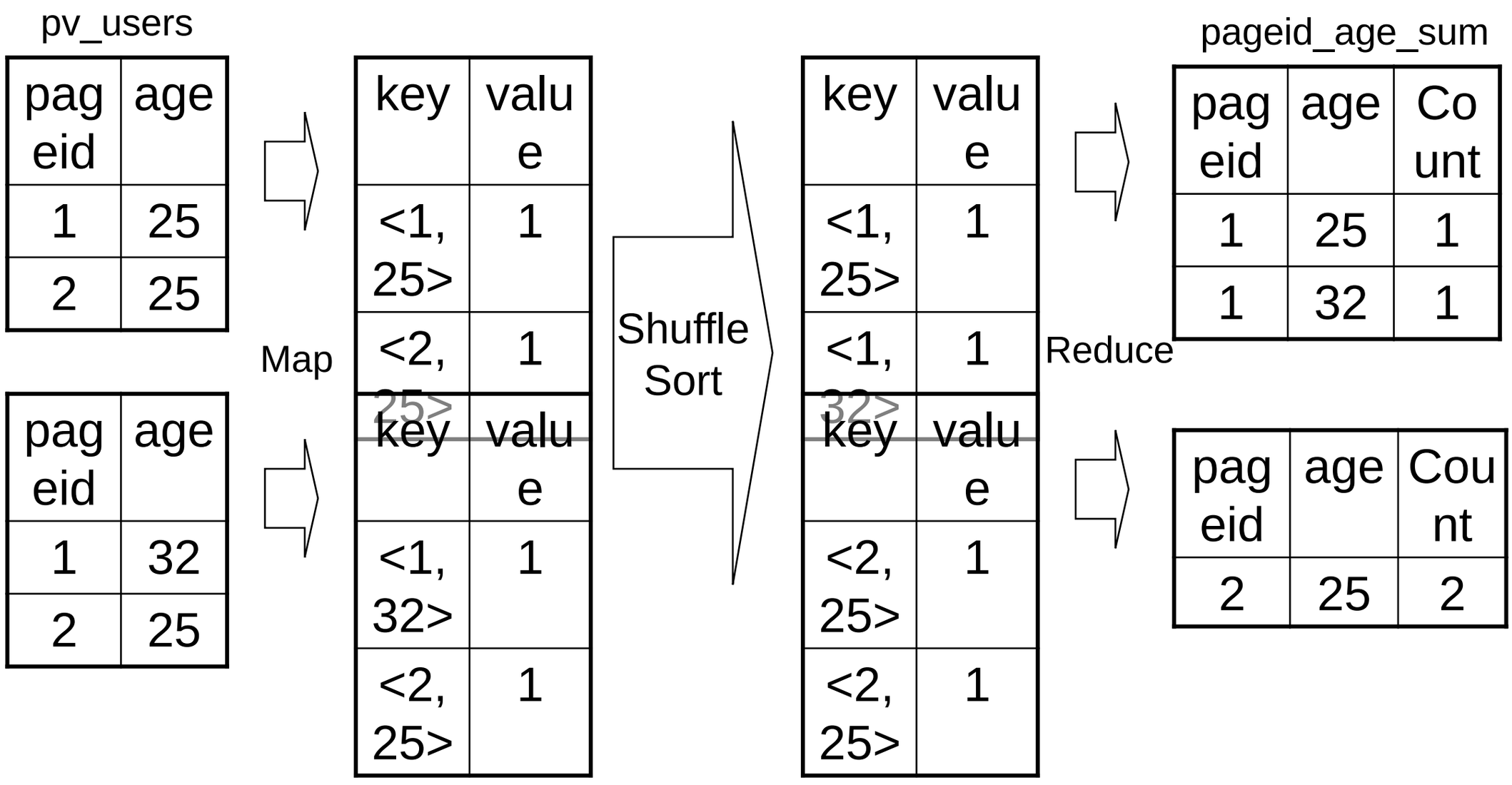


pageid_age_sum

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;

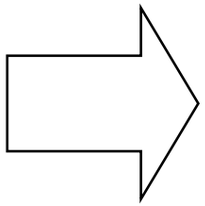
Hive QL – Group By in Map Reduce



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

Shuffle and Sort

Shuffle randomly.

key	v
<1,111>	9:08:01
<2,111>	9:08:13
<1,222>	9:08:14
<2,111>	9:08:20

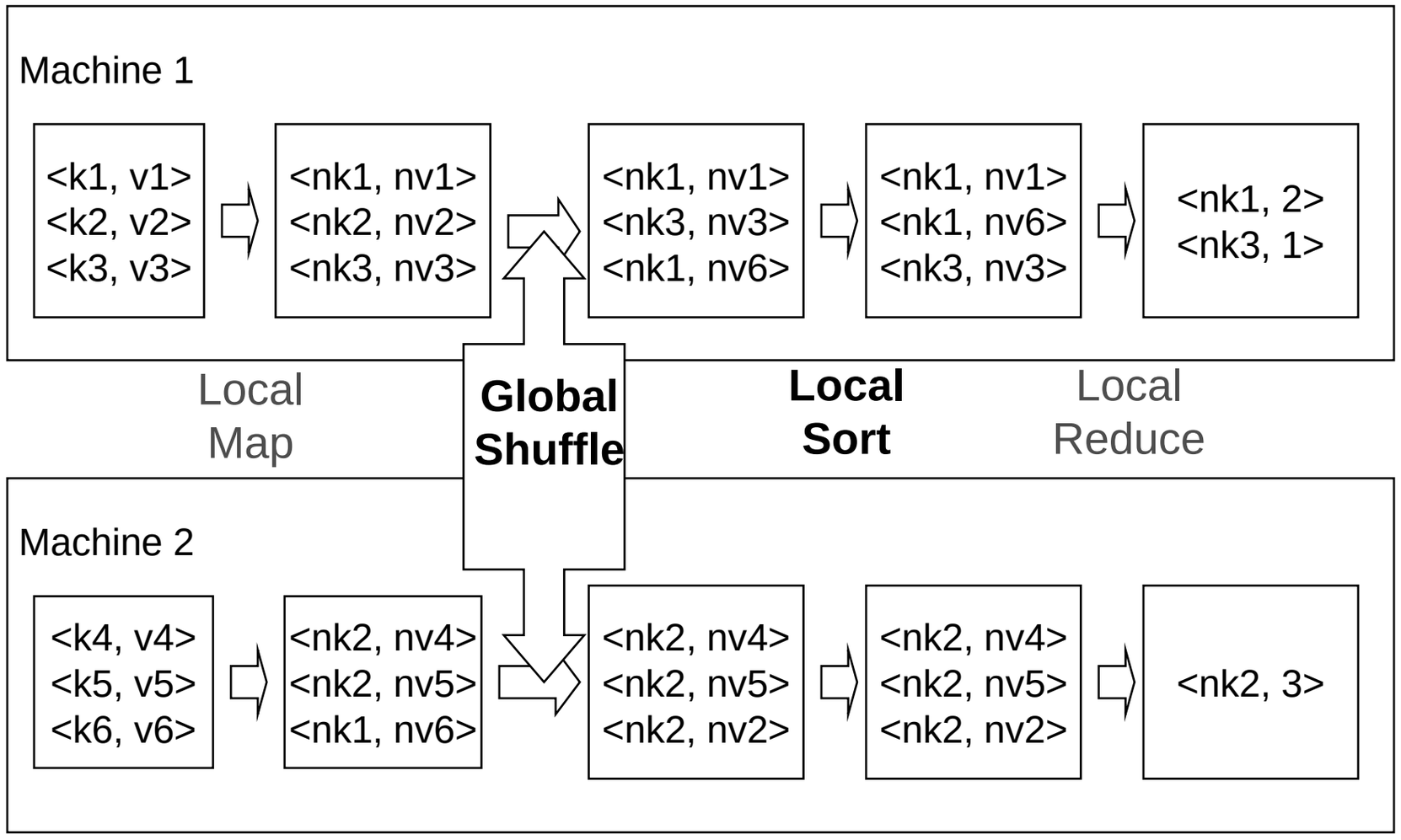
Reduce

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

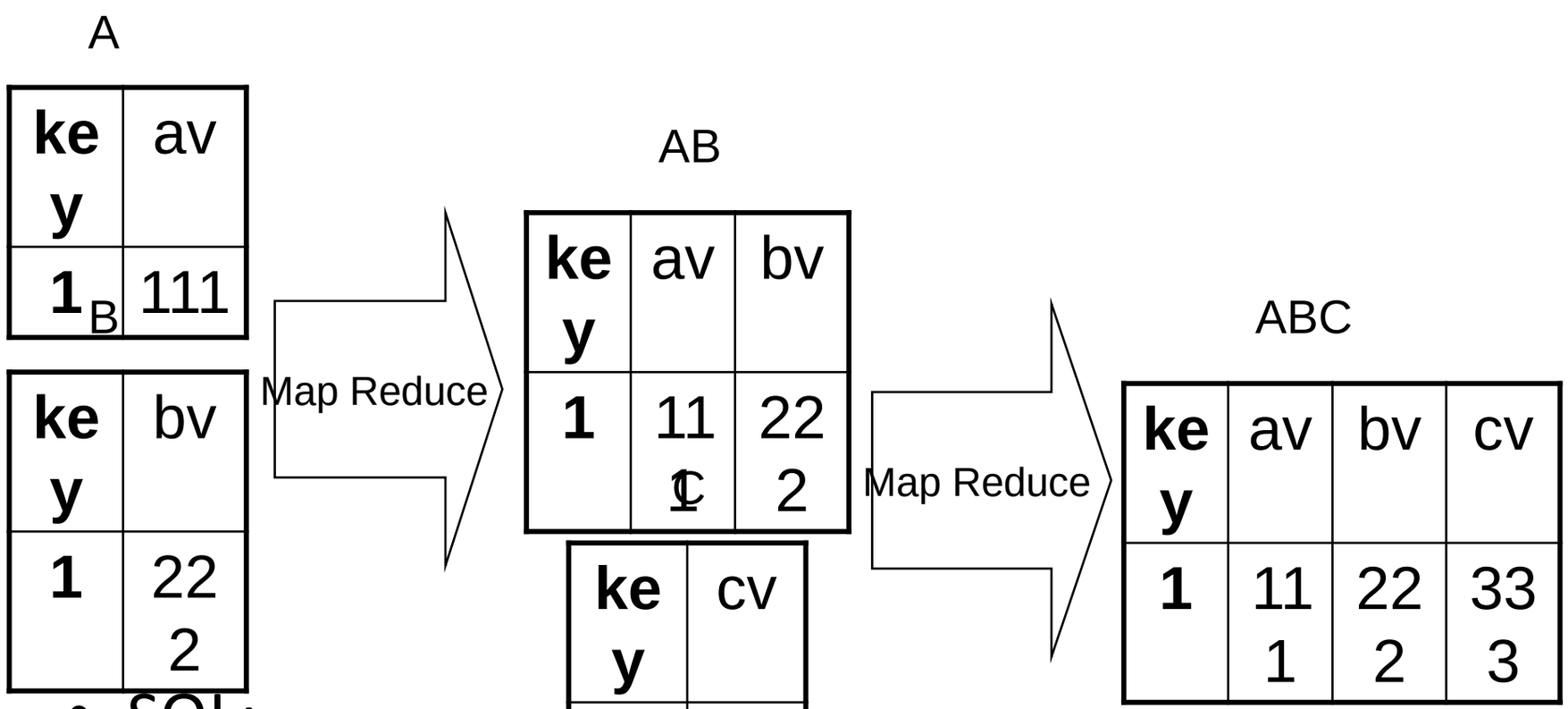
Hive Optimizations

Efficient Execution of SQL on top of Map-Reduce

(Simplified) Map Reduce Revisit



Merge Sequential Map Reduce Jobs



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

Share Common Read Operations

pageid	age
1	25
2	32

Map Reduce

pageid	count
1	1
2	1

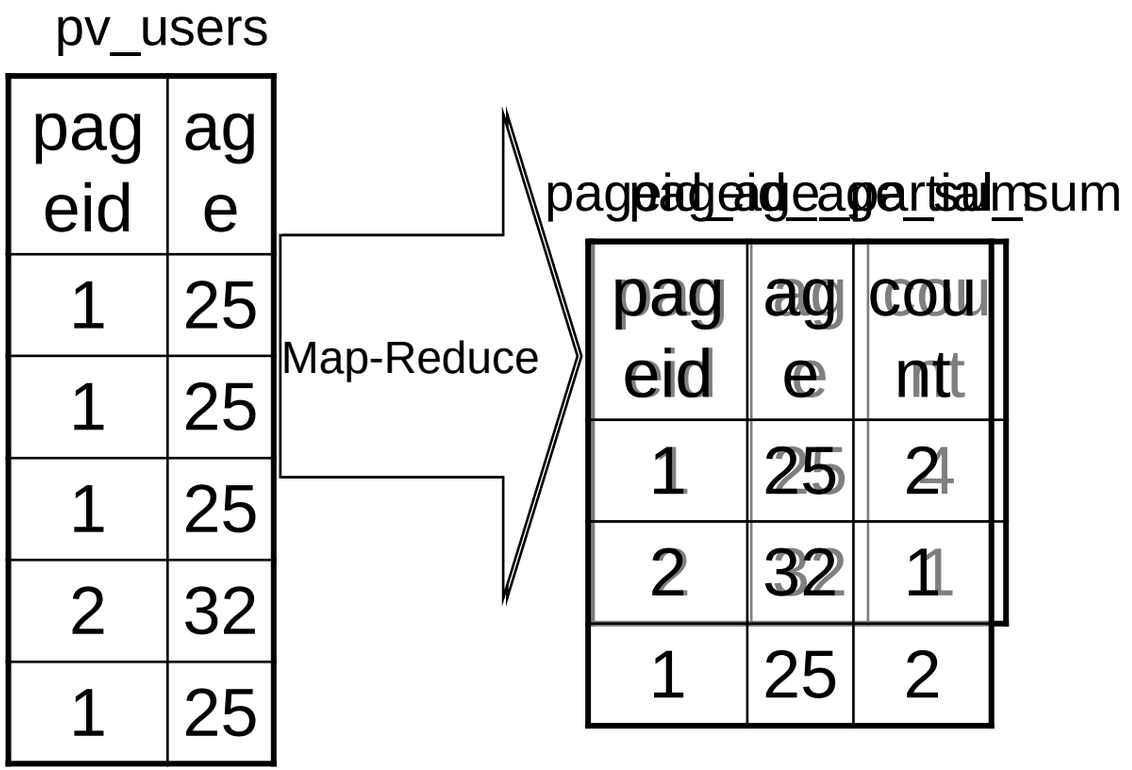
pageid	age
1	25
2	32

Map Reduce

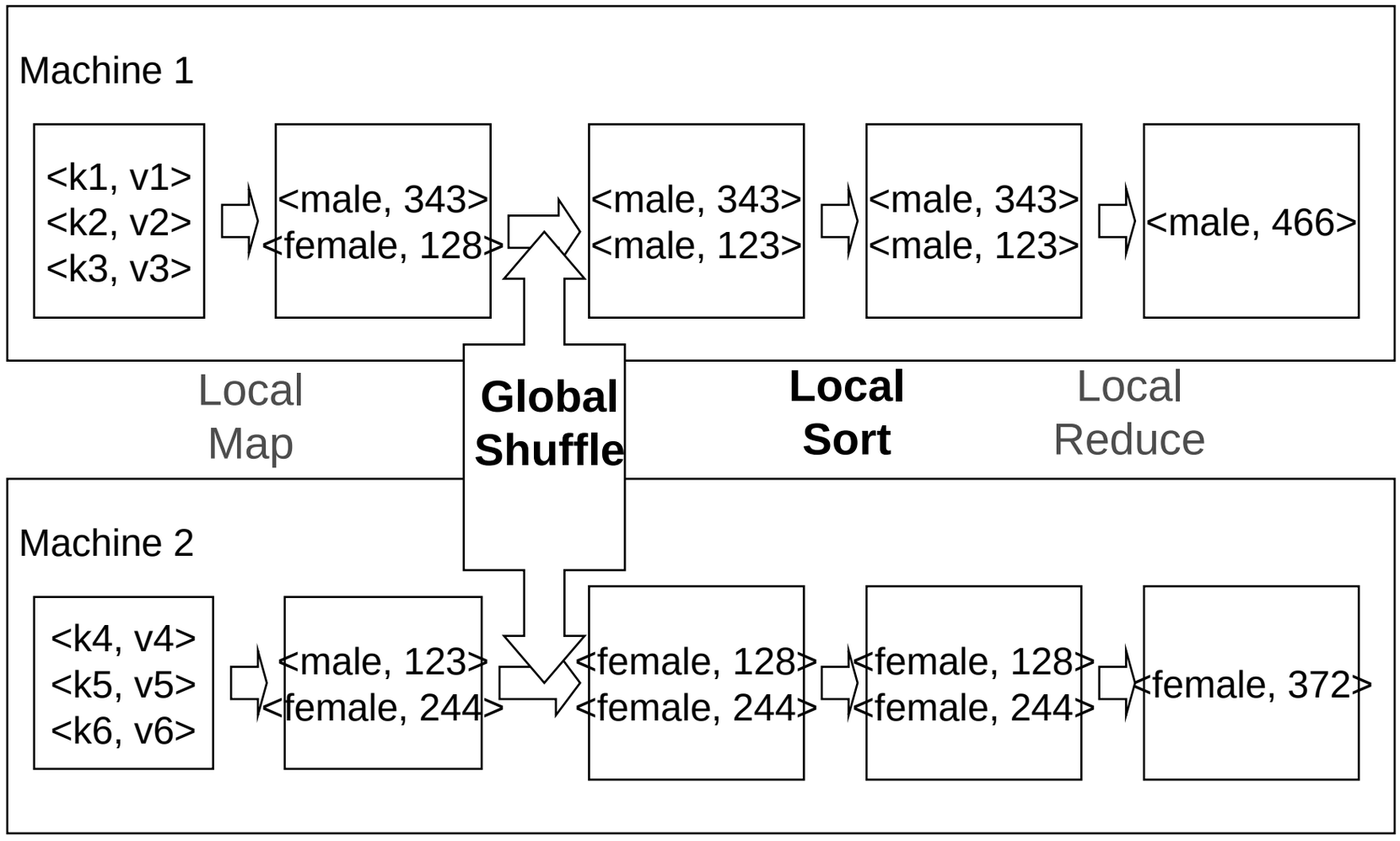
age	count
25	1
32	1

- 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);

TODO: Column-based Storage and Map-side Join

url	page quality	IP
http://a.com/	90	65.1.2.3
http://b.com/	20	68.9.0.81
http://c.com/	68	11.3.85.1

url	clicked	viewed
http://a.com/	12	145
http://b.com/	45	383
http://c.com/	23	67

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