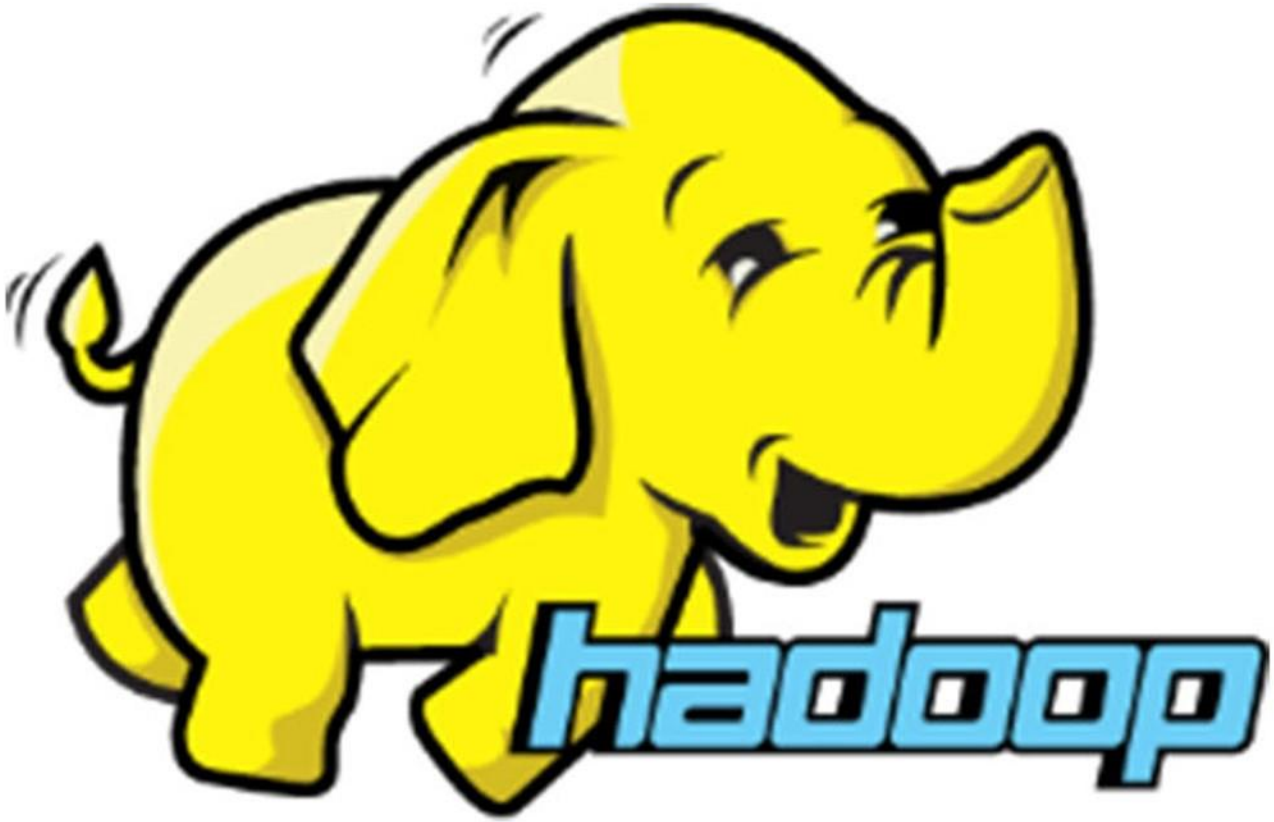
Week 02

DISTRIBUTED COMPUTING

INTRODUCTION TO HADOOP

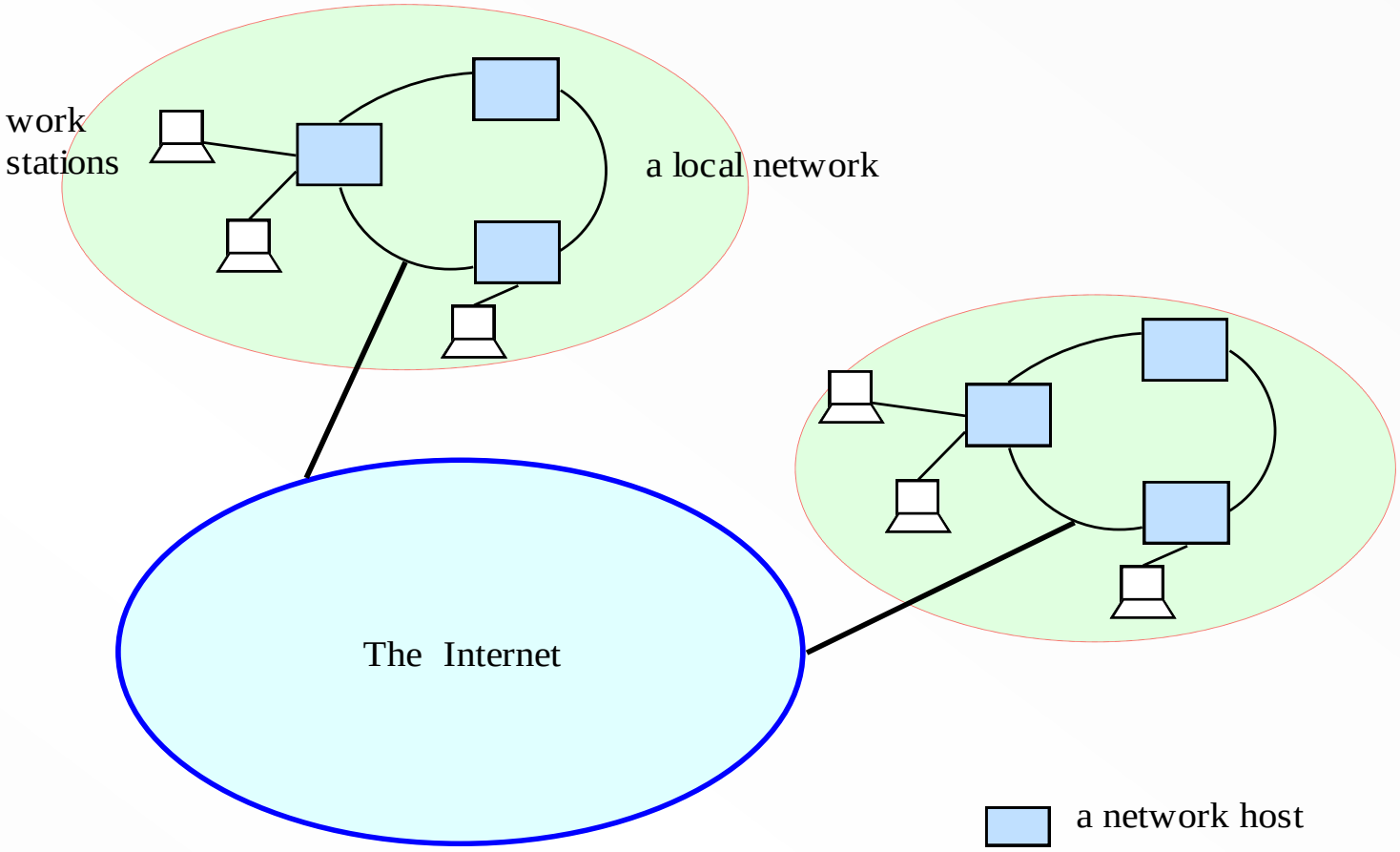


Course Instructor: Hassan Jahangir

Distributed system, distributed computing

- Early computing was performed on a single processor. Uni-processor computing can be called *centralized computing*.
- A *distributed system* is a collection of independent computers, interconnected via a network, capable of collaborating on a task.
- *Distributed computing* is computing performed in a distributed system.

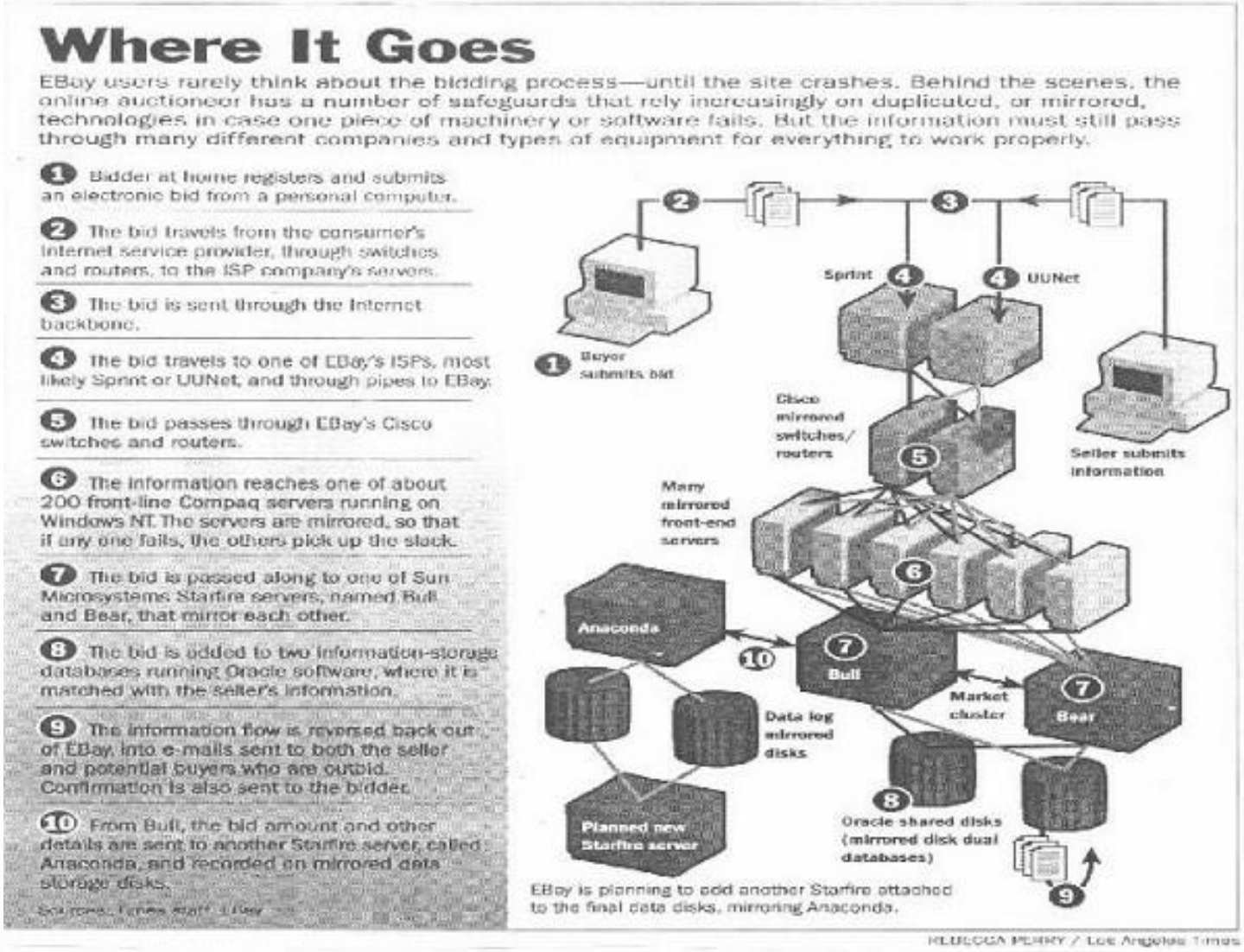
Distributed Systems



Examples of Distributed systems

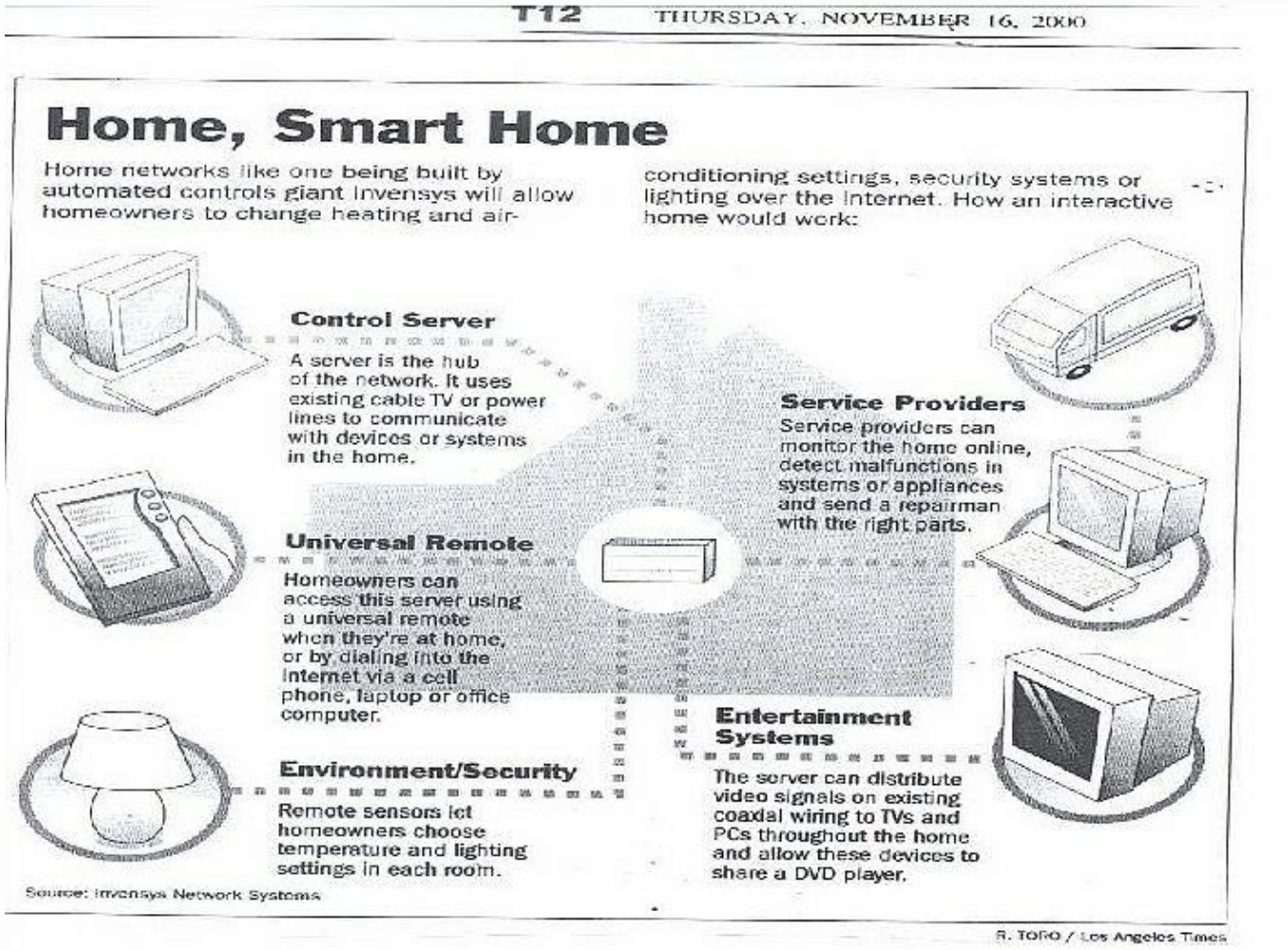
- Network of workstations (NOW): a group of networked personal workstations connected to one or more server machines.
- The Internet
- An intranet: a network of computers and workstations within an organization, segregated from the Internet via a protective device (a firewall).

Example of a large-scale distributed system – eBay **(Source: Los Angeles Times.)**



An example small-scale distributed system

(Source: Los Angeles Times.)



T12 THURSDAY, NOVEMBER 16, 2000

Home, Smart Home

Home networks like one being built by automated controls giant Invensys will allow homeowners to change heating and air-conditioning settings, security systems or lighting over the Internet. How an interactive home would work:

Control Server
A server is the hub of the network. It uses existing cable TV or power lines to communicate with devices or systems in the home.

Universal Remote
Homeowners can access this server using a universal remote when they're at home, or by dialing into the Internet via a cell phone, laptop or office computer.

Environment/Security
Remote sensors let homeowners choose temperature and lighting settings in each room.

Service Providers
Service providers can monitor the home online, detect malfunctions in systems or appliances and send a repairman with the right parts.

Entertainment Systems
The server can distribute video signals on existing coaxial wiring to TVs and PCs throughout the home and allow these devices to share a DVD player.

Source: Invensys Network Systems

R. TORO / Los Angeles Times

Computers in a Distributed System

- Workstations: computers used by end-users to perform computing
- Server machines: computers which provide resources and services
- Personal Assistance Devices: handheld computers connected to the system via a wireless communication link.

The power of the Internet

(Source: the Usability Professional Association's site.)

- 60 million American households use computers.*(The New York Times, 5/28/98)*
- The number of computer users in the workplace has increased from 600,000 in 1976 to 80 million today. *(San Francisco Examiner, 3/29/98)*
- 84% of Internet users say that the Web is indispensable. Nearly the same percentage find e-mail indispensable. 85% use the Internet every day. *(GVU, 1997)*

The Power of the Internet – 2

(Source: www.cisco.com)

- BACKBONE CAPACITY: The capacity of the Internet backbone to carry information is doubling every 100 days. (U.S. Internet Council, Apr. 1999).
- DATA TRAFFIC SURPASSING VOICE: Voice traffic is growing at 10% per year or less, while data traffic is conservatively estimated to be growing at 125% per year, meaning voice will be less than 1% of the total traffic by 2007. (Technology Futures, Inc March 2000).
- DOMAIN NAMES: There are 12,844,877 unique domain names (e.g. Cisco.com) registered worldwide, with 428,023 new domain names registered each week. (NetNames Statistics 12/28/1999).

The Power of the Internet – 3

(Source: www.cisco.com)

- DOMAIN NAMES: There are 12,844,877 unique domain names (e.g. Cisco.com) registered worldwide, with 428,023 new domain names registered each week. (NetNames Statistics 12/28/1999).
- HOST COMPUTERS: In July 1999 there were 56.2 million "host" computers supporting web pages. In July 1997 there were 19.5 million host computers, with 3.2 million hosts in July 1994, and a mere 80,000 in July 1989. (Internet Software Consortium – Internet Domain Survey).
- TOTAL AMOUNT OF DATA: 1,570,000,000 pages, 29,400,000,000,000 bytes of text, 353,000,000 images, and 5,880,000,000,000 bytes of image data. (The Censorware Project, Jan. 26, 1999).

The Power of the Internet – 4

(Source: www.cisco.com)

- EMAIL VOLUME: Average U.S. consumer will receive 1,600 commercial email messages in 2005, up from 40 in 1999, while non-marketing and personal correspondence will more than double from approximately 1,750 emails per year in 1999 to almost 4,000 in 2005 (Jupiter Communications, May 2000).
- 159 million computers in the U.S., 135 million in EU, and 116 million in Asia Pacific (as of April 2000).
- WEB HITS/DAY: U.S. web pages averaged one billion hits per day (aggregate) in October 1999. (eMarketer/Media Metrix, Nov. 1999).

NUMBER OF AMERICANS ONLINE – HISTORICAL

(Source: www.cisco.com)

- 1993 – 90,000 (U.S. Internet Council, Apr. 1999).
- 1997 – 19 million (Stratis Group, Apr. 1999).
- 1998 – 68 million in 1998. (Strategis Group, Nov. 1999).
- 1998 – 84 million from home or work (Stratis Group, Apr. 1999).
- 1998 – 37 million DAILY (Stratis Group, Apr. 1999).
- 1999, Nov. – 118.4 million (Cyberatlas/Nielsen Net Ratings, Dec. 1999).
- 1999, Nov. – 74 million actually went online (Cyberatlas/Nielsen Net Ratings, Dec. 1999).

PERCENTAGE OF AMERICANS ONLINE

(Source: www.cisco.com)

- 1998 – 28% (IDC, Oct. 1999).
- 1998 – 42% of the U.S. adult population. (Stratis Group, Apr. 1999)
- 2003 – 62% (IDC, Oct. 1999).
- 2003 – 67% (Yankee Group, 1999).
- 2005 – 91% (Strategy Analytics, Dec. 1999).

The Power of the Internet – 5

(Source: www.cisco.com)

- NEW USERS Q1 2000: More than 5 million Americans joined the online world in the first quarter of 2000, which averages to roughly 55,000 new users each day, 2,289 new users each hour, or 38 new users each minute. (CyberAtlas / Telecommunications Reports International, May 2000).
- US INTERNET USAGE: Average US Internet user went online 18 sessions, spent a total of 9 hours, 5 minutes and 24 seconds online and visited 10 unique sites per month. (Nielsen NetRatings, June 2000).

The Power of the Internet – 6

(Source: www.cisco.com)

- E-MAIL 1998: The U.S. Postal Service delivered 101 billion pieces of paper mail in 1998. Estimates for e-mail messages sent in 1998 range from 618 billion to 4 trillion. (U.S. Internet Council, Apr. 1999).
- E-MAIL 1999: There are 270 million e-mailboxes in the U.S. -- roughly 2.5 per user. (eMarketer/ Messaging Online, Nov. 1999).
- HOURS ONLINE (Veronis, Suhler & Associates, Nov. 1999):

  1997 – 28 hours per capita

  1998 – 74 hours per capita

  2003 – 192 hours per capita

ONLINE WORLDWIDE

(Source: www.cisco.com)

- 1998 – 95.43 million people. (eMarketer eStats 1999).
- 1998, Dec. – 144 million (IDC, Dec. 1999).
- 1999, Dec. – 240 million (IDC, Dec. 1999).
- 2002 – over 490 million (Computer Industry Almanac, Nov. 1999).
- 2005 – over 765 million (Computer Industry Almanac, Nov. 1999
- U.S. -- 136 million (36% of world's total) (eMarketer, May 2000) – followed by Japan (27 M), UK (18M), and China (16 M).

Wireless access to the Internet

(Source: www.cisco.com)

- U.S. WIRELESS USERS: 61.5 million Americans will be using wireless devices to access the Internet in 2003, up from 7.4 million in the US today (728% increase). (IDC Research, Feb. 2000).
- MOBILE DATA: Almost 80% of the US Internet population will access data from mobile phones in a year's time, up from the current figure of 3%. (Corechange, Inc & Cap Gemini USA, Apr. 2000).

“The network really is the computer.”

Tim O’Reilly, in an address at 6/2000 Java One:

“By now, it's a truism that the Internet runs on open source. Bind, the Berkeley Internet Name Daemon, is the single most mission critical program on the Internet, followed closely by Sendmail and Apache, open source servers for two of the Internet's most widely used application protocols, **SMTP** (Simple Mail Transfer Protocol) and **HTTP** (HyperText Transfer Protocol).”

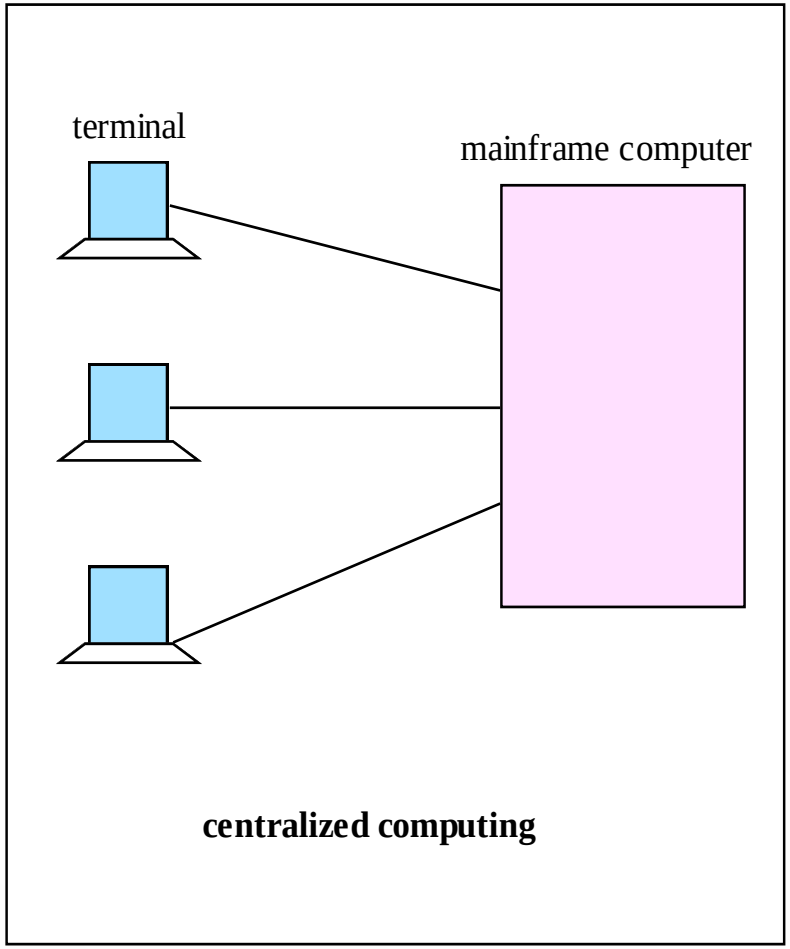
Early “killer apps”:

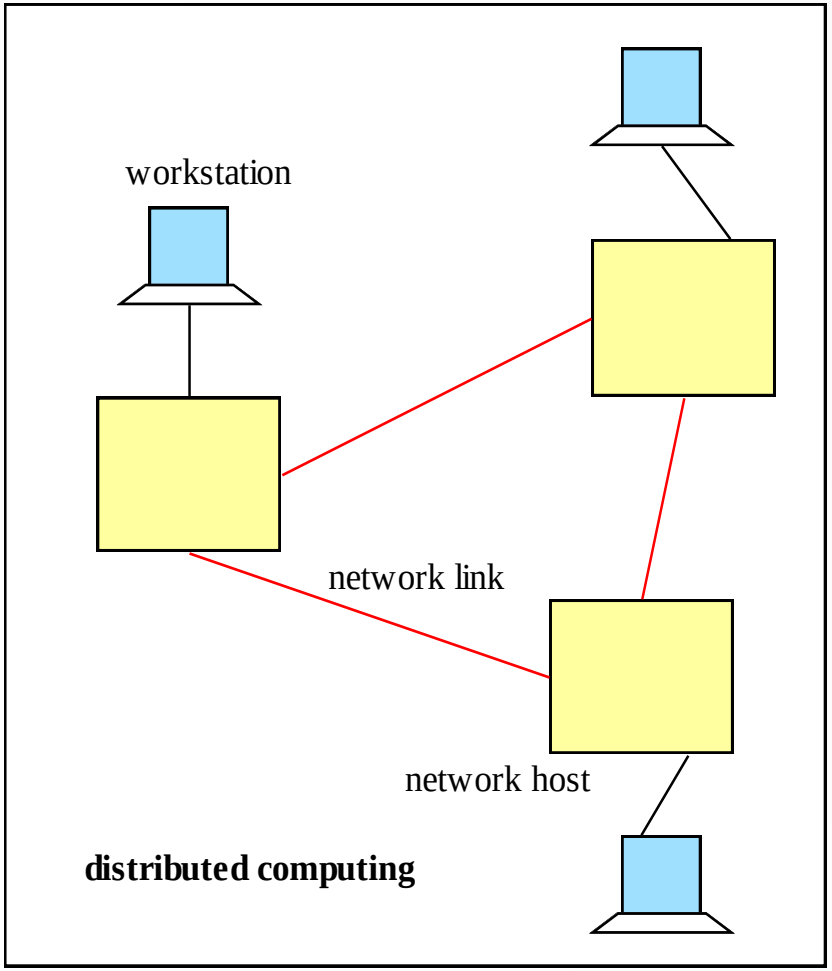
- usenet: distributed bulletin board
- email
- talk

Recent “killer apps”:

- the web
- Peer-to-Peer (e.g., Napster, Gnutella, KaZaA)
- collaborative computing (e.g., Java Shared Data Toolkit)

Centralized vs. Distributed Computing





Monolithic mainframe applications vs. distributed applications

based on http://www.inprise.com/visibroker/papers/distributed/wp.html

- **The monolithic mainframe application architecture**:
  - Separate, single-function applications, such as order-entry or billing
  - Applications cannot share data or other resources
  - Developers must create multiple instances of the same functionality (service).
  - Proprietary (user) interfaces
- **The distributed application architecture**:
  - Integrated applications
  - Applications can share resources
  - A single instance of functionality (service) can be reused.
  - Common user interfaces

Evolution of paradigms

- Client-server: Socket API, RMI (remote method invocation)
- Distributed objects
- Object broker: CORBA
- Network service: Jini
- Object space: JavaSpaces
- Mobile agents
- Message oriented middleware (MOM): Java Message Service
- Collaborative applications

Why distributed computing?

- Economics: distributed systems allow the pooling of resources, including CPU cycles, data storage, input/output devices, and services.
- Reliability: a distributed system allow replication of resources and/or services, thus reducing service outage due to failures.
- The Internet has become a universal platform for distributed computing.

The Weaknesses and Strengths of Distributed Computing

In any form of computing, there is always a tradeoff in advantages and disadvantages

Some of the reasons for the popularity of distributed computing :

- **The affordability of computers and availability of network access**
- **Resource sharing**
- **Scalability**
- **Fault Tolerance**

The Weaknesses and Strengths of Distributed Computing

The disadvantages of distributed computing:

- **Multiple Points of Failures:** the failure of one or more participating computers, or one or more network links, can spell trouble.
- **Security Concerns**: In a distributed system, there are more opportunities for unauthorized attack.

Summary - 1

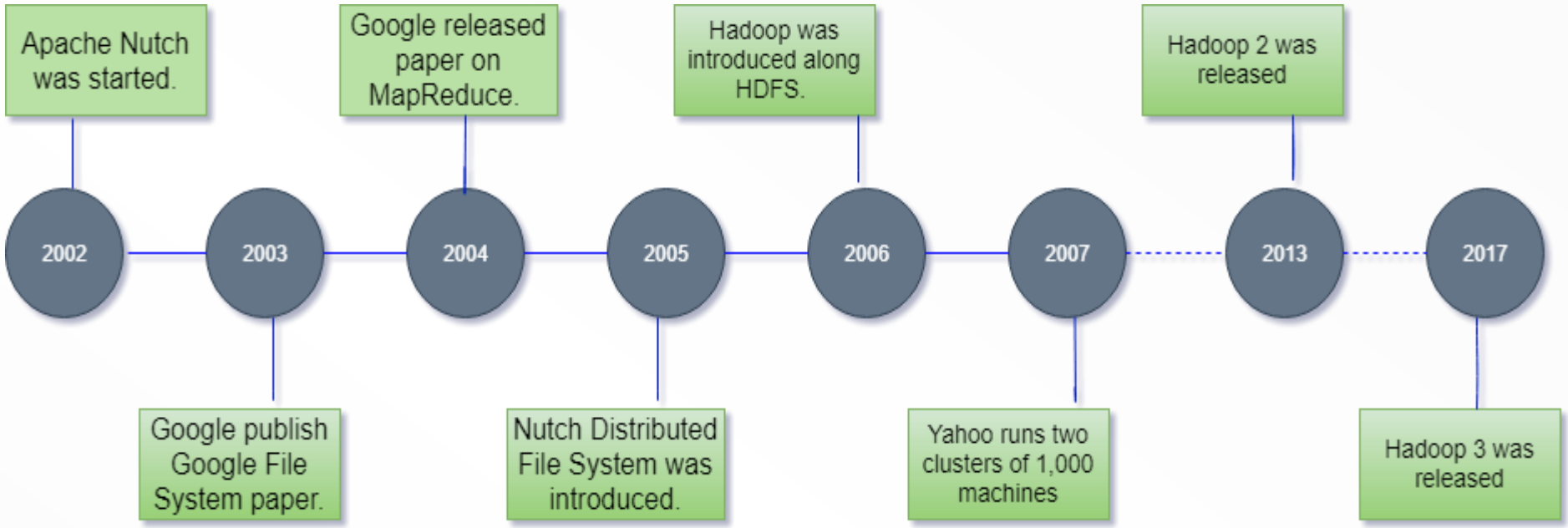
We discussed the following topics:

- What is meant by distributed computing
- Distributed system
- Distributed computing vs. parallel computing
- Basic concepts in operating system: processes and threads

Introduction to Hadoop

History of Hadoop

- The Hadoop was started by Doug Cutting and Mike Cafarella in 2002. Its origin was the Google File System paper, published by Google.



Year	Event
2003	Google released the paper, Google File System (GFS).
2004	Google released a white paper on Map Reduce.
2006	•Hadoop introduced. •Hadoop 0.1.0 released. •Yahoo deploys 300 machines and within this year reaches 600 machines.
2007	•Yahoo runs 2 clusters of 1000 machines. •Hadoop includes HBase.
2008	•YARN JIRA opened •Hadoop becomes the fastest system to sort 1 terabyte of data on a 900 node cluster within 209 seconds. •Yahoo clusters loaded with 10 terabytes per day. •Cloudera was founded as a Hadoop distributor.
2009	•Yahoo runs 17 clusters of 24,000 machines. •Hadoop becomes capable enough to sort a petabyte. •MapReduce and HDFS become separate subproject.
2010	•Hadoop added the support for Kerberos. •Hadoop operates 4,000 nodes with 40 petabytes. •Apache Hive and Pig released.
2011	•Apache Zookeeper released. •Yahoo has 42,000 Hadoop nodes and hundreds of petabytes of storage.
2012	Apache Hadoop 1.0 version released.
2013	Apache Hadoop 2.2 version released.
2014	Apache Hadoop 2.6 version released.
2015	Apache Hadoop 2.7 version released.
2017	Apache Hadoop 3.0 version released.
2018	Apache Hadoop 3.1 version released.

What is Hadoop

- Hadoop is an open source framework from Apache and is used to store process and analyze data which are very huge in volume.
- It is being used by Facebook, Yahoo, Google, Twitter, LinkedIn and many more. Moreover it can be scaled up just by adding nodes in the cluster.

HDFS (Hadoop Distributed File System)

- A distributed file system that provides high-throughput access to application data

- HDFS uses a master/slave architecture in which one device (master) termed as NameNode controls one or more other devices (slaves) termed as DataNode.

- It breaks Data/Files into small blocks (128 MB each block) and stores on DataNode and each block replicates on other nodes to accomplish fault tolerance.

- NameNode keeps the track of blocks written to the DataNode.

Hadoop Ecosystem

HBASE (NoSQL Database)

Hive (Analytics)

Pig (Scripting)

Oozie (Scheduling)

Zookeeper (Coordination)

Processing

Resource Management

Storage

Ingestion

Hadoop Ecosystem

- **HDFS:** Hadoop Distributed File System
- **YARN:** Yet Another Resource Negotiator
- **MapReduce:** Programming based Data Processing
- **Spark:** In-Memory data processing
- **PIG, HIVE:** Query based processing of data services
- **HBase:** NoSQL Database
- **Mahout, Spark MLLib:** Machine Learning algorithm libraries
- **Solar, Lucene:** Searching and Indexing
- **Zookeeper:** Managing cluster
- **Oozie:** Job Scheduling

HDFS:

- HDFS is the primary or major component of Hadoop ecosystem and is responsible for storing large data sets of structured or unstructured data across various nodes and thereby maintaining the metadata in the form of log files.
- HDFS consists of three core components i.e.
  - Name node [FS Image + Edit Log]
  - Secondary Name Node [Update FS Image File and Edit Log]
  - Data Node [Read Write Data + Heart Beat massage]
- Name Node is the prime node which contains metadata (data about data) requiring comparatively fewer resources than the data nodes that stores the actual data. These data nodes are commodity hardware in the distributed environment. Undoubtedly, making Hadoop cost effective.
- HDFS maintains all the coordination between the clusters and hardware, thus working at the heart of the system.

Name Node

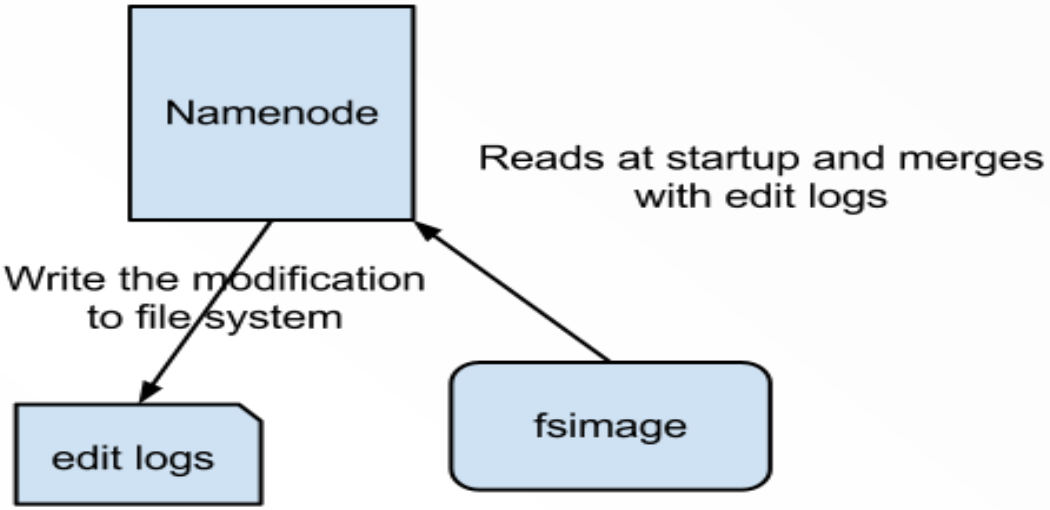
- Keeps the metadata of all files/blocks in the file system, and tracks where across the cluster the file data is kept. It does not store the data of these files itself. Kind of block lookup dictionary(index or address book of blocks).

- Client applications talk to the NameNode whenever they wish to locate a file, or when they want to add/copy/move/delete a file. The NameNode responds the successful requests by returning a list of relevant DataNode servers where the data lives

Name Node (Contd.)

fsimage - Its the snapshot of the filesystem when NameNode started

Edit logs - Its the sequence of changes made to the filesystem after NameNode started



Data Node

- DataNode stores data in the Hadoop Filesystem
- A functional filesystem has more than one DataNode, with data replicated across them
- On startup, a DataNode connects to the NameNode; spinning until that service comes up. It then responds to requests from the NameNode for filesystem operations.
- Client applications can talk directly to a DataNode, once the NameNode has provided the location of the data

Data Replication

- **Why need data replication ?**

- HDFS is designed to handle large scale data in distributed environment
- Hardware or software failure, or network partition exist
- Therefore need replications for those fault tolerance

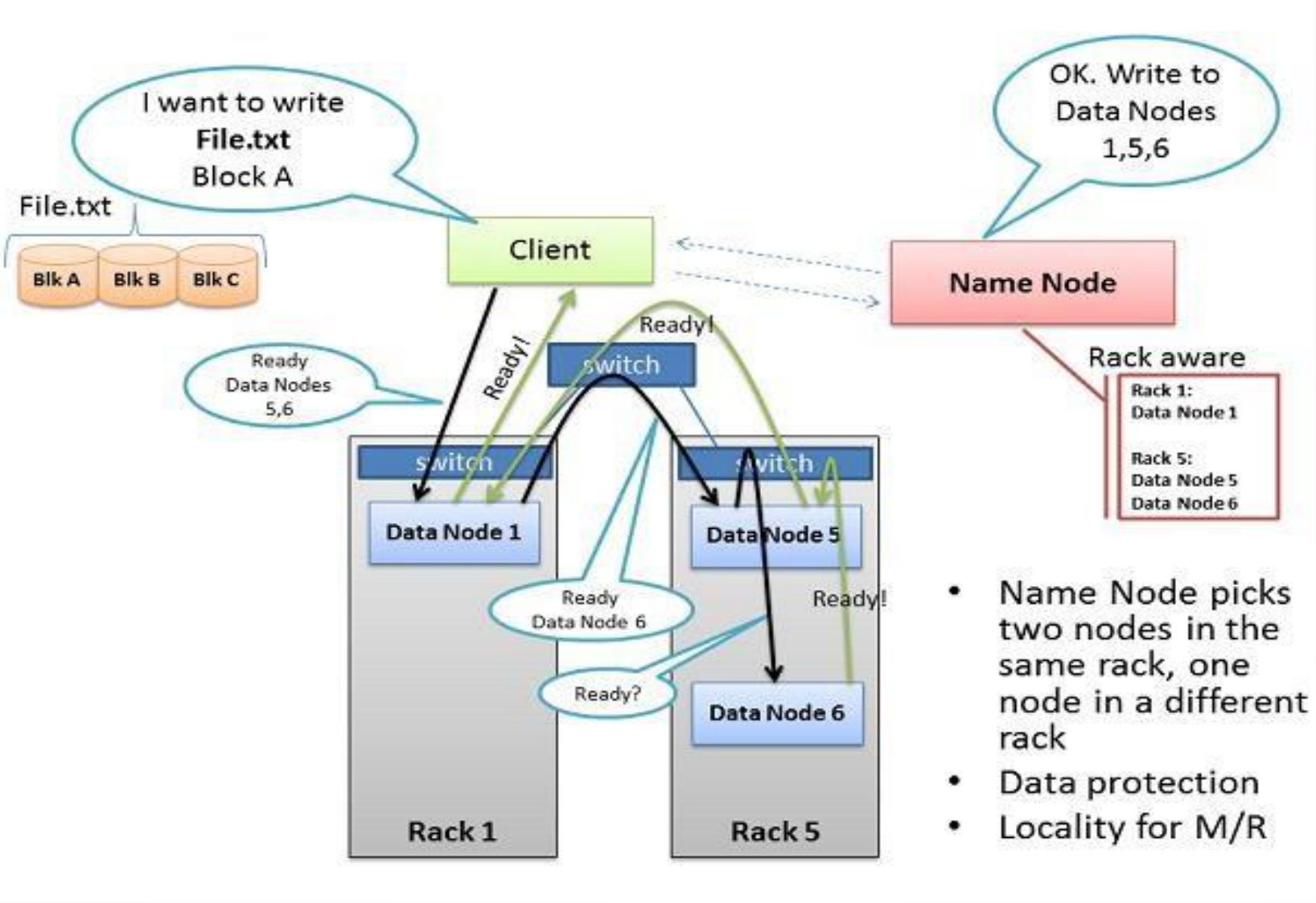
Replication (Contd.)

- **Replication factor**
  - Decided by users, and can be dynamically tuned.

- **How to Create replications efficiently ?**
  - Replication pipeline: Instead of single machine create replications, a pipe line is applied
  - Machine 1 make replication to machine 2, at the same time machine 2 make the replication to machine 3, etc.

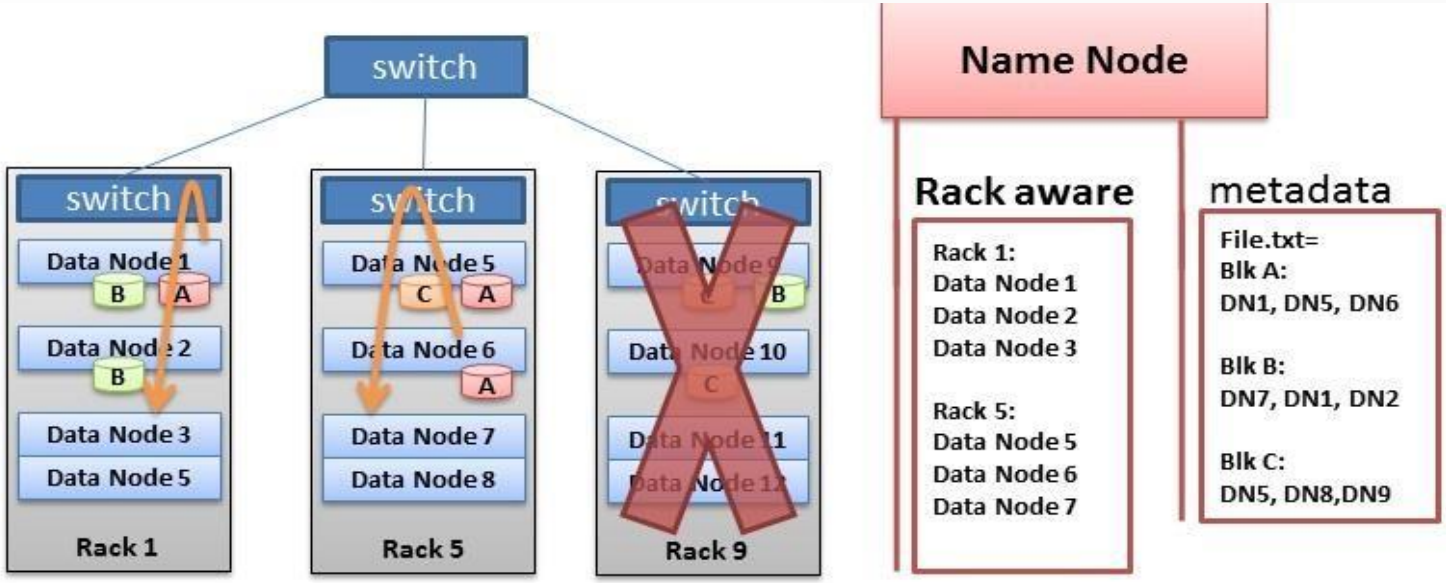
- **Replication placement**
  - High initialization time to create replication to all machines
  - An approximate solution: Only 3 replications One replication resides in current node One replication resides in current rack One replication resides in another rack

Replication Pipeline



- Name Node picks two nodes in the same rack, one node in a different rack
- Data protection
- Locality for M/R

Rack Awareness



- Never loose all data if entire rack fails
- Keep bulky flows in-rack when possible
- Assumption that in-rack is higher bandwidth, lower latency

Data Node Failure

Data Node Failure Condition

If a data node failed, Name Node could know the blocks it contains, create same replications to other alive nodes, and unregister this dead node

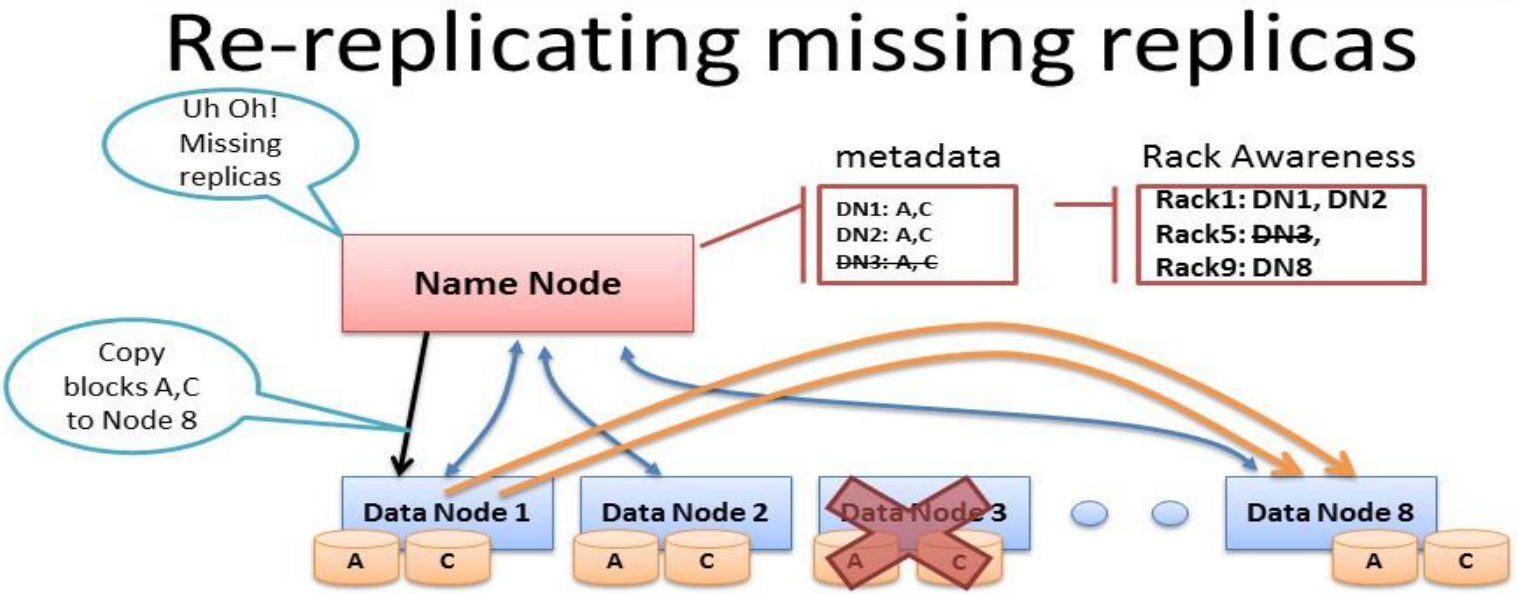
Data Integrity

Corruption may occur in network transfer, Hardware failure etc.

Apply checksum checking on the contents of files on HDFS, and store the checksum in HDFS namespace

If checksum is not correct after fetching, drop it and fetch another replication from other machines.

Heartbeats and Re-Replication



- Missing Heartbeats signify lost Nodes
- Name Node consults metadata, finds affected data
- Name Node consults Rack Awareness script
- Name Node tells a Data Node to re-replicate

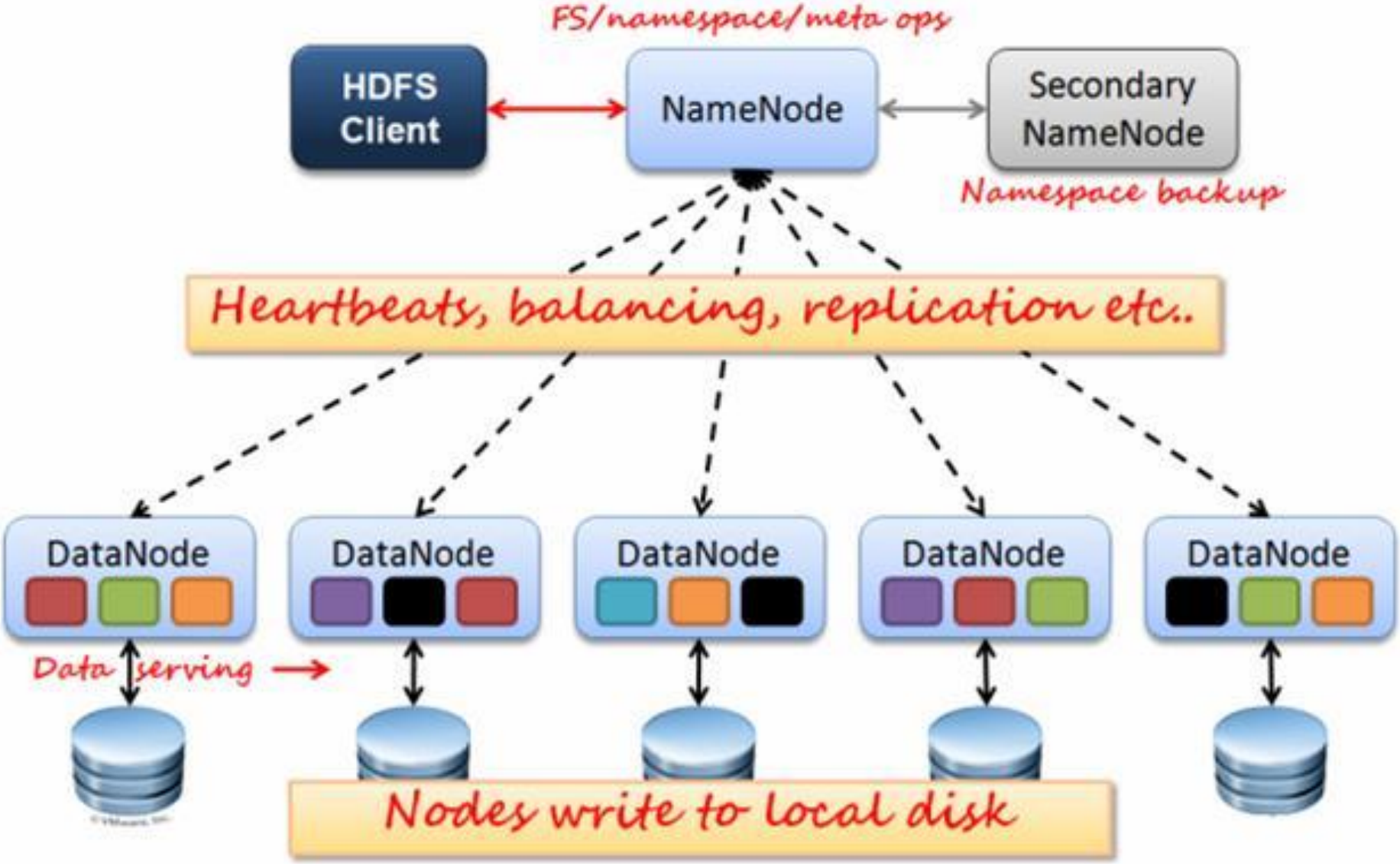
Secondary Name Node

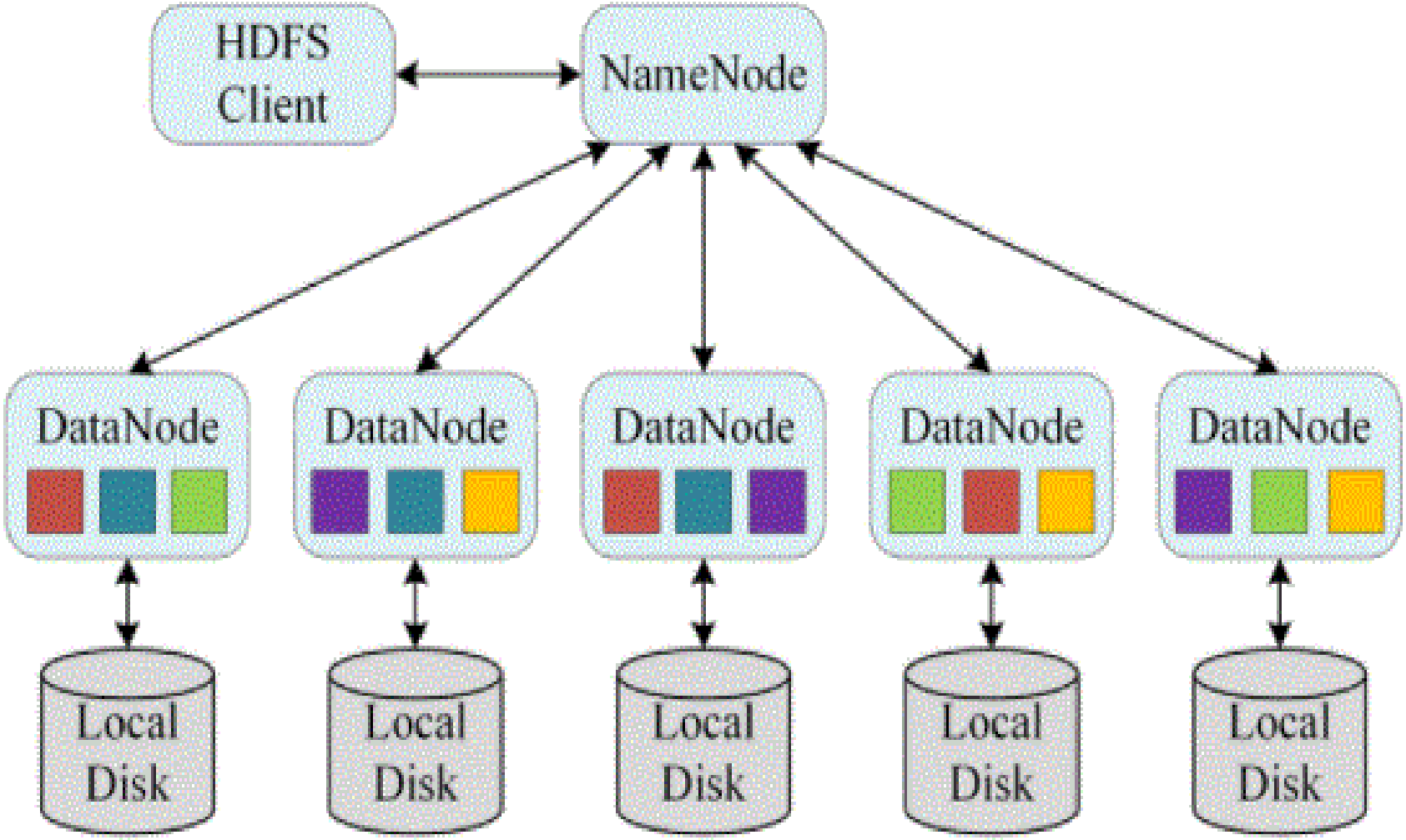
- Not a failover NameNode
- The only purpose of the secondary name-node is to perform periodic checkpoints. The secondary name-node periodically downloads current name-node image and edits log files, joins them into new image and uploads the new image back to the (primary and the only) name-node
- Default checkpoint time is one hour. It can be set to one minute on highly busy clusters where lots of write operations are being performed.

Name Node Failure

- NameNode is the single point of failure in the cluster
- If NameNode is down due to software glitch, restart the machine
- If original NameNode can be restored, secondary can re-establish the most current metadata snapshot
- If machine don't come up, metadata for the cluster is irretrievable. In this situation create a new NameNode, use secondary to copy metadata to new primary, restart whole cluster

HDFS Architecture




HDFS
Client
NameNode
DataNode
DataNode
DataNode
DataNode
DataNode
Local
Disk
Local
Disk
Local
Disk
Local
Disk
Local
Disk

Use Cases for Hadoop

- To aggregate "data exhaust" — messages, posts, blog entries, photos, video clips, maps, web graph

- To give data context — friends networks, social graphs, recommendations, collaborative filtering

- To keep apps running — web logs, system logs, system metrics, database query logs

- To deliver novel mashup services – mobile location data, clickstream data, SKUs, pricing

Assumptions and Goals of HDFS

- Hardware Failure
- Streaming Data Access (Best for batch processing)
- Large Data Sets
- Simple Coherency Model (write-once-read-many access model)
- Portability Across Heterogeneous Hardware and Software Platforms

MapReduce

Motivation: Google Example

- 20+ billion web pages x 20KB = 400+ TB
- 1 computer reads 30-35 MB/sec from disk
  - ~4 months to read the web
- ~1,000 hard drives to store the web
- Takes even more to **do** something useful with the data!
- **Today, a standard architecture for such problems is emerging:**
  - Cluster of commodity Linux nodes
  - Commodity network (ethernet) to connect them

Large-scale Computing

- **Large-scale computing** for **data mining** problems on **commodity hardware**
- **Challenges:**
  - **How do you distribute computation?**
  - **How can we make it easy to write distributed programs?**
  - **Machines fail:**
    - One server may stay up 3 years (1,000 days)
    - If you have 1,000 servers, expect to loose 1/day
    - People estimated Google had ~1M machines in 2011
      - 1,000 machines fail every day!

Idea and Solution

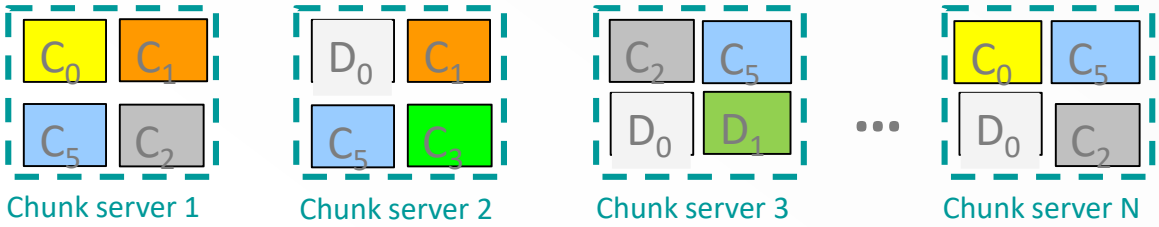
- **Issue:** Copying data over a network takes time
- **Idea:**
  - Bring computation close to the data
  - Store files multiple times for reliability
- **Map-reduce** addresses these problems
  - Google's computational/data manipulation model
  - Elegant way to work with big data
  - **Storage Infrastructure – File system**
    - Google: GFS. Hadoop: HDFS
  - **Programming model**
    - Map-Reduce

Storage Infrastructure

- **Problem:**
  - If nodes fail, how to store data persistently?
- **Answer:**
  - **Distributed File System:**
    - Provides global file namespace
    - Google GFS; Hadoop HDFS;
- **Typical usage pattern**
  - Huge files (100s of GB to TB)
  - Data is rarely updated in place
  - Reads and appends are common

Distributed File System

- **Reliable distributed file system**
- Data kept in "chunks" spread across machines
- Each chunk replicated on different machines
  - Seamless recovery from disk or machine failure



Bring computation directly to the data!

Chunk servers also serve as compute servers

Programming Model: MapReduce

Warm-up task:

- We have a huge text document
- Count the number of times each distinct word appears in the file
- **Sample application:**
  - Analyze web server logs to find popular URLs

Task: Word Count

Case 1:

- File too large for memory, but all <word, count> pairs fit in memory

Case 2:

- Count occurrences of words:
  - **`words(doc.txt) | sort | uniq -c`**
    - where **`words`** takes a file and outputs the words in it, one per a line
- Case 2 captures the essence of **MapReduce**
  - Great thing is that it is naturally parallelizable

Apache MapReduce

- A software framework for distributed processing of large data sets
- The framework takes care of scheduling tasks, monitoring them and re-executing any failed tasks.
- It splits the input data set into independent chunks that are processed in a completely parallel manner.
- MapReduce framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically, both the input and the output of the job are stored in a file system.

MapReduce Architecture

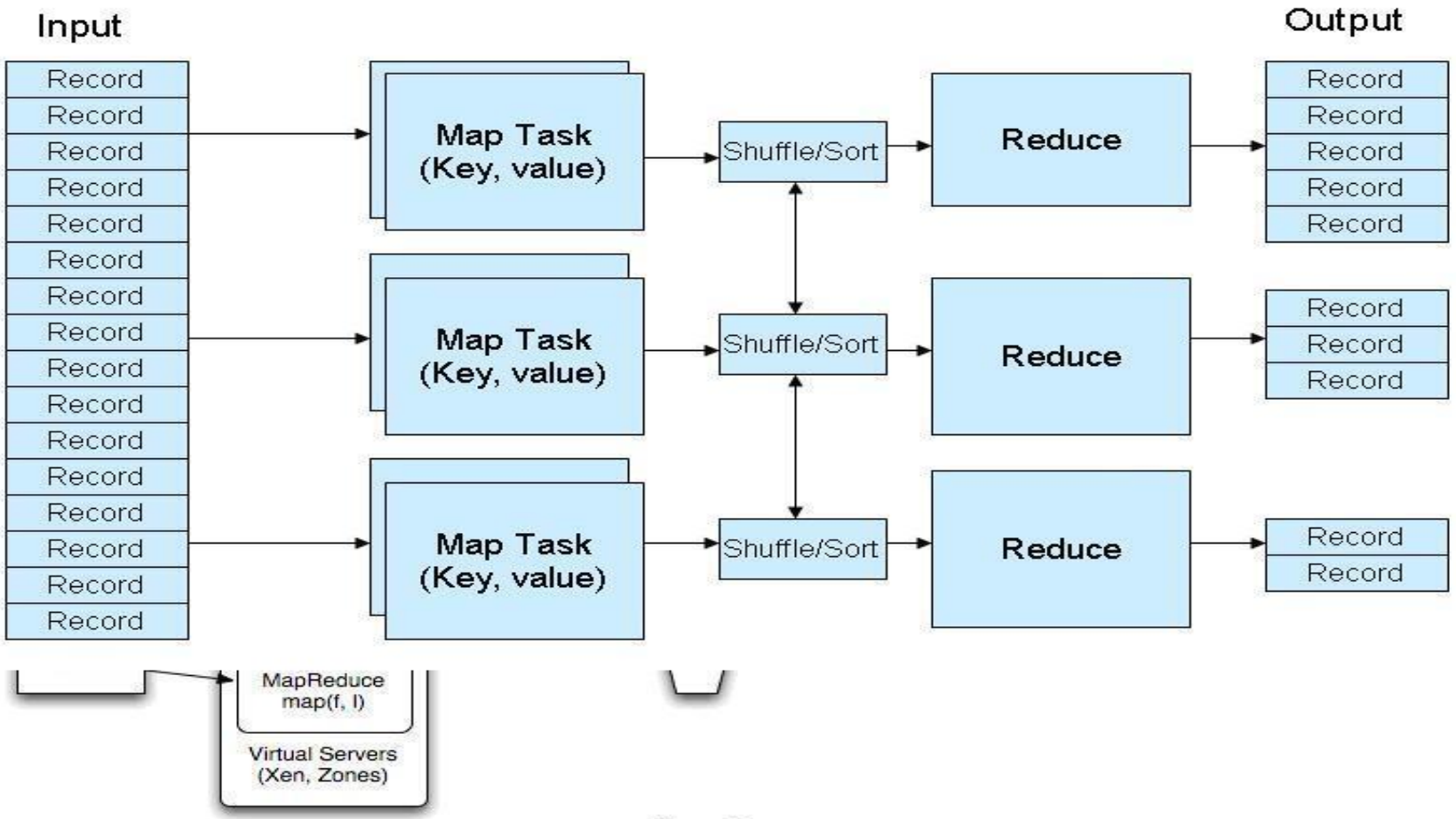
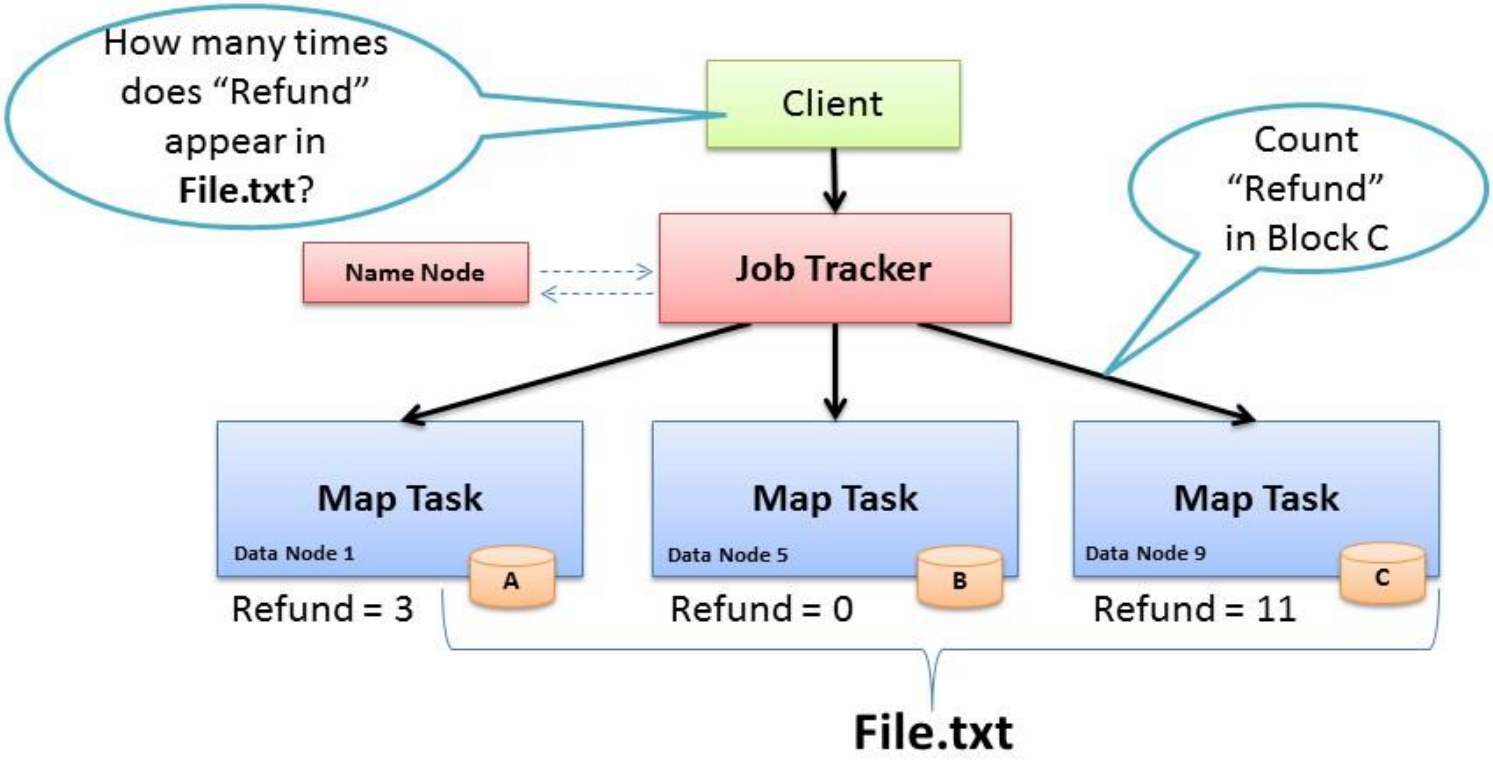


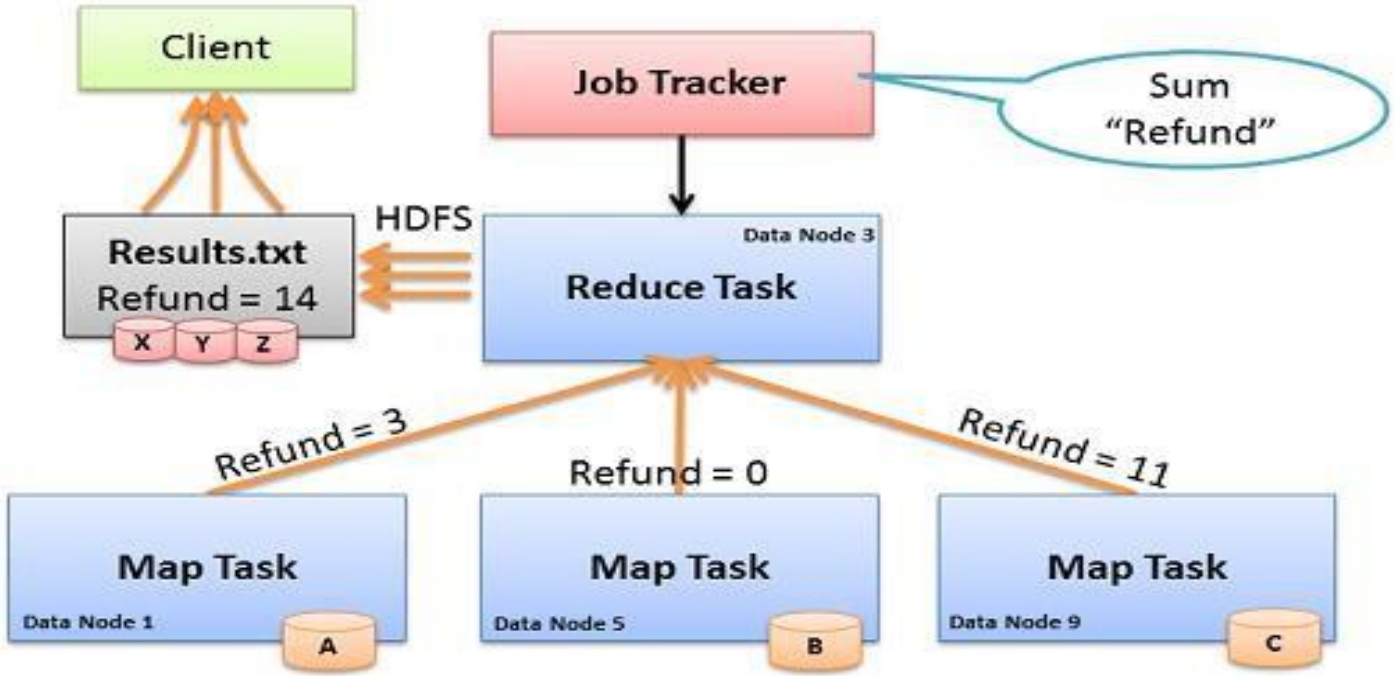
Figure 2

Map Task



- **Map:** "Run this computation on your local data"
- Job Tracker delivers Java code to Nodes with local data

Reduce Task



- **Reduce:** "Run this computation across Map results"
- Map Tasks send output data to Reducer over the network
- Reduce Task data output written to and read from HDFS

MapReduce Dataflow

- An input reader
- A Map function
- A partition function
- A compare function
- A Reduce function
- An output writer

MapReduce Daemons

- JOB TRACKER
- TASK TRACKER

JobTracker

- JobTracker is the daemon service for submitting and tracking MapReduce jobs in Hadoop

- JobTracker performs following actions in Hadoop :

  - It accepts the MapReduce Jobs from client applications
  - Talks to NameNode to determine data location
  - Locates available TaskTracker Node
  - Submits the work to the chosen TaskTracker Node

TaskTracker

- A TaskTracker node accepts map, reduce or shuffle operations from a JobTracker
- Its configured with a set of *slots*, these indicate the number of tasks that it can accept
- JobTracker seeks for the free slot to assign a job
- TaskTracker notifies the JobTracker about job success status.
- TaskTracker also sends the heartbeat signals to the job tracker to ensure its availability, it also reports the no. of available free slots with it.

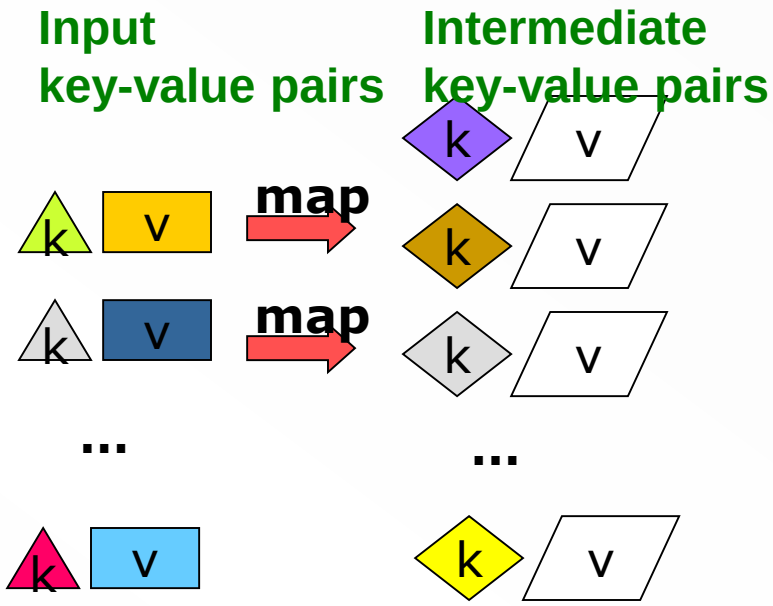
MapReduce: Overview

- Sequentially read a lot of data
- **Map:**
  - Extract something you care about
- **Group by key:** Sort and Shuffle
- **Reduce:**
  - Aggregate, summarize, filter or transform
- Write the result

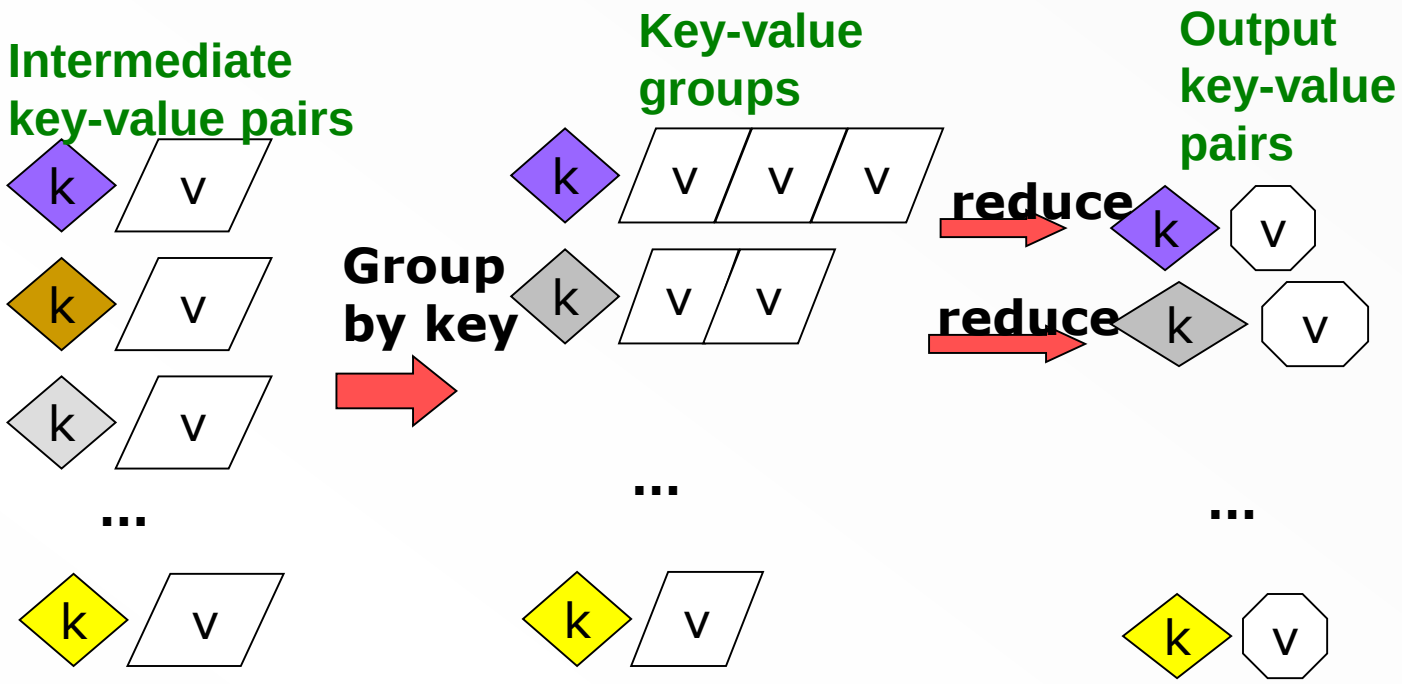
Outline stays the same, **Map** and **Reduce** change to fit the problem

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

MapReduce: The Map Step



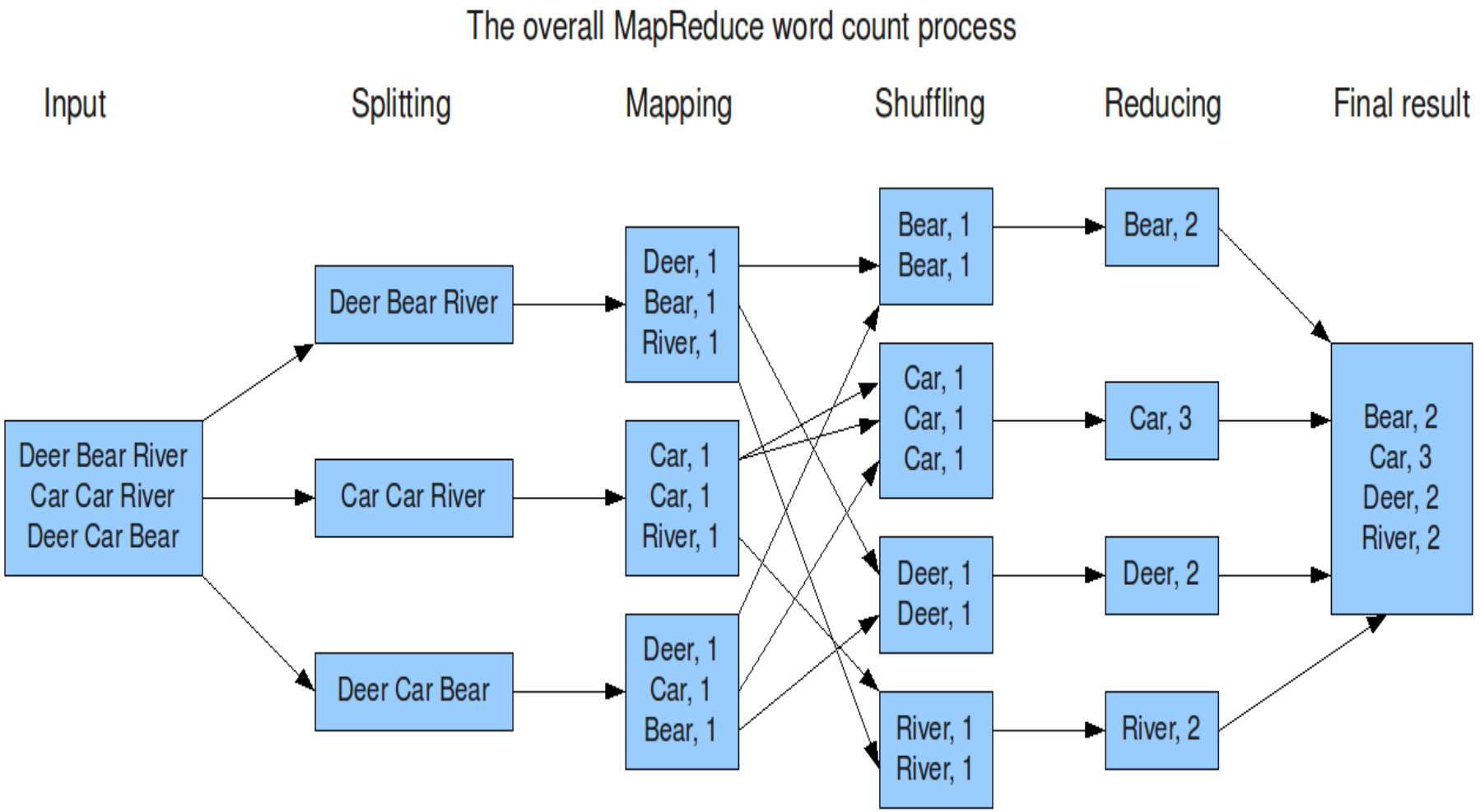
MapReduce: The Reduce Step



More Specifically

- **Input:** a set of key-value pairs
- Programmer specifies two methods:
  - **Map(k, v)** → <k', v'>*
    - Takes a key-value pair and outputs a set of key-value pairs
      - E.g., key is the filename, value is a single line in the file
    - There is one Map call for every *(k,v)* pair
  - **Reduce(k', <v'>*)** → <k', v''>*
    - **All values *v'* with same key *k'* are reduced together and processed in *v'* order**
    - There is one Reduce function call per unique key *k'*

MapReduce: Word Counting

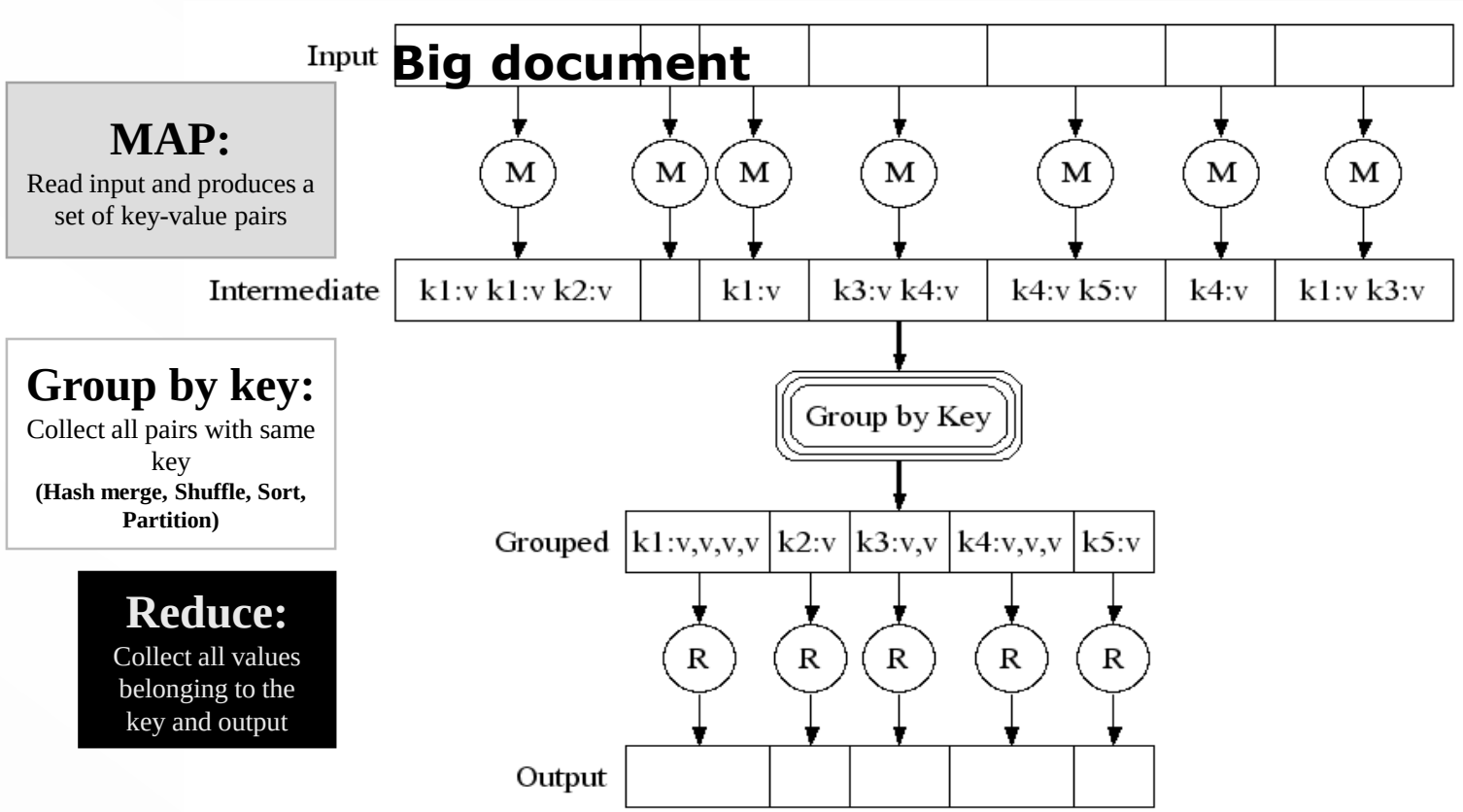


Map-Reduce: Environment

Map-Reduce environment takes care of:

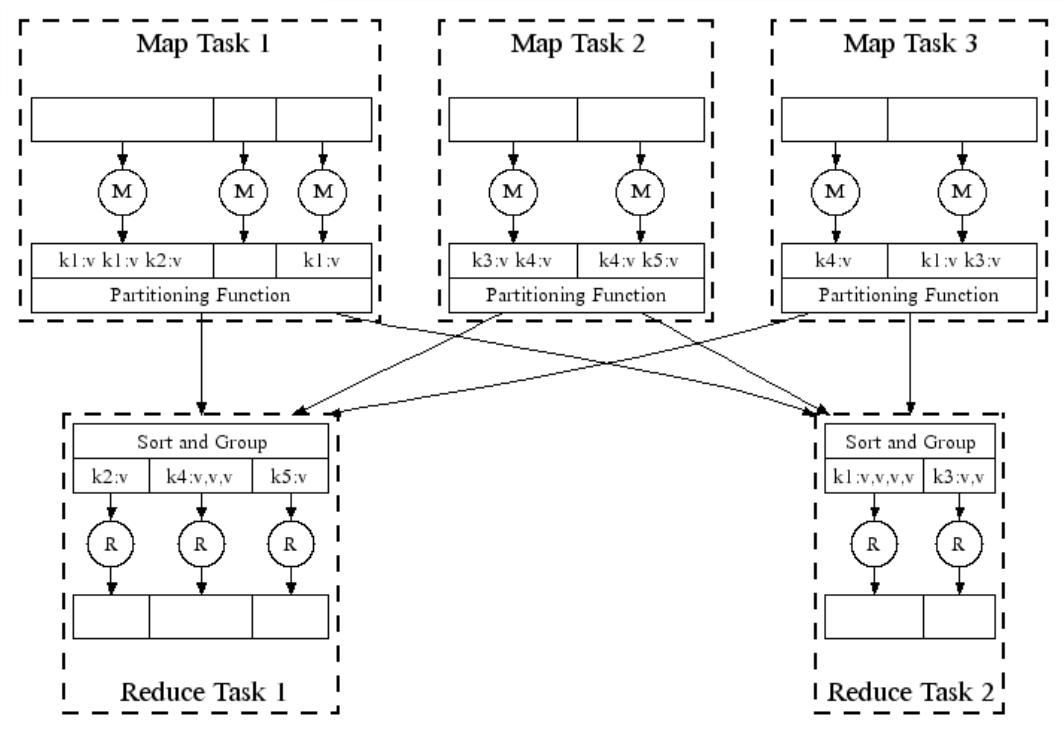
- Partitioning the input data
- Scheduling the program's execution across a set of machines
- Performing the **group by key** step
- Handling machine failures
- Managing required inter-machine communication

Map-Reduce: A diagram



skovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

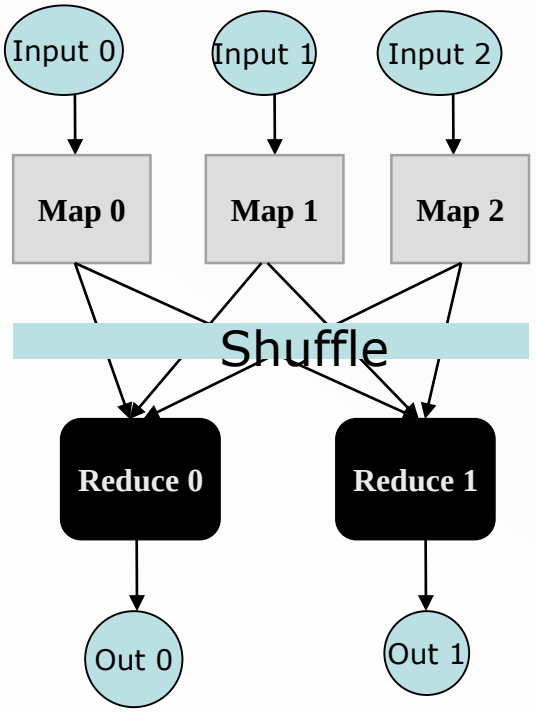
Map-Reduce: In Parallel



All phases are distributed with many tasks doing the work

Map-Reduce

- Programmer specifies:
  - Map and Reduce and input files
- Workflow:
  - Read inputs as a set of key-value-pairs
  - Map transforms input kv-pairs into a new set of k'v'-pairs
  - Sorts & Shuffles the k'v'-pairs to output nodes
  - All k'v'-pairs with a given k' are sent to the same **reduce**
  - **Reduce** processes all k'v'-pairs grouped by key into new k"v"-pairs
  - Write the resulting pairs to files
- All phases are distributed with many tasks doing the work



J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org