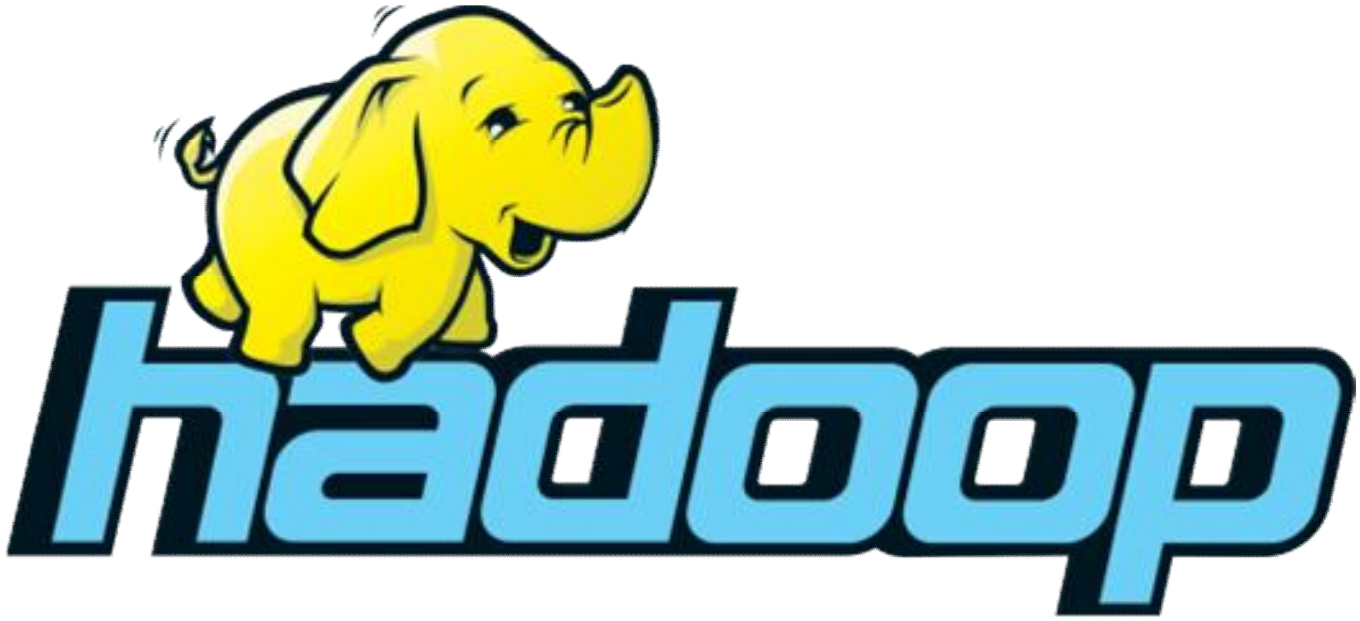
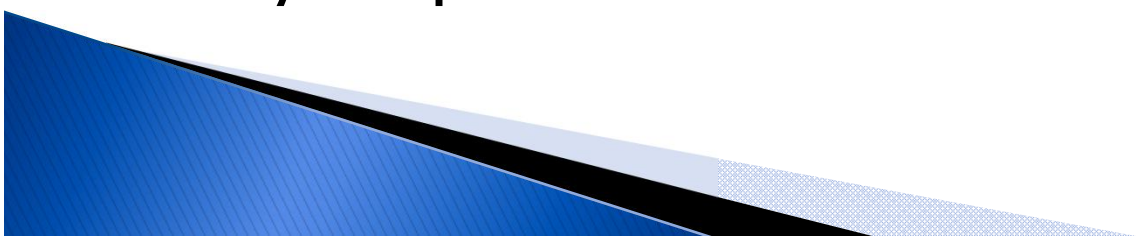


INDRA AULIA

Department of Information Technology
Faculty of Computer Science and Information Technology
Universitas Sumatera Utara

What is Apache Hadoop?

- Open source software framework designed for storage and processing in the industry for large-scale, massively parallel, and distributed data processing on clusters of commodity hardware
- Created by Doug Cutting and Mike Carafella in 2005.
- Cutting named the program after his son's toy elephant.



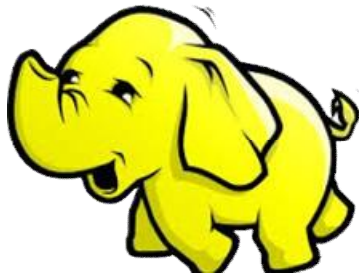
Hadoop's Developers

Doug Cutting

2005: Doug Cutting and Michael J. Cafarella developed Hadoop to support distribution for the Nutch search engine project.

The project was funded by Yahoo.

2006: Yahoo gave the project to Apa
Software Foundation.



Google Origins

2003

The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

Google*

2004

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

2006

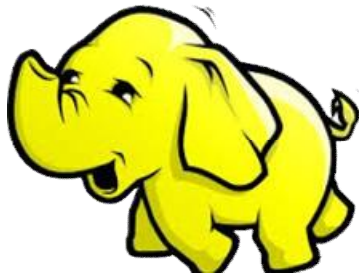
Bigtable: A Distributed Storage System for Structured Data

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber

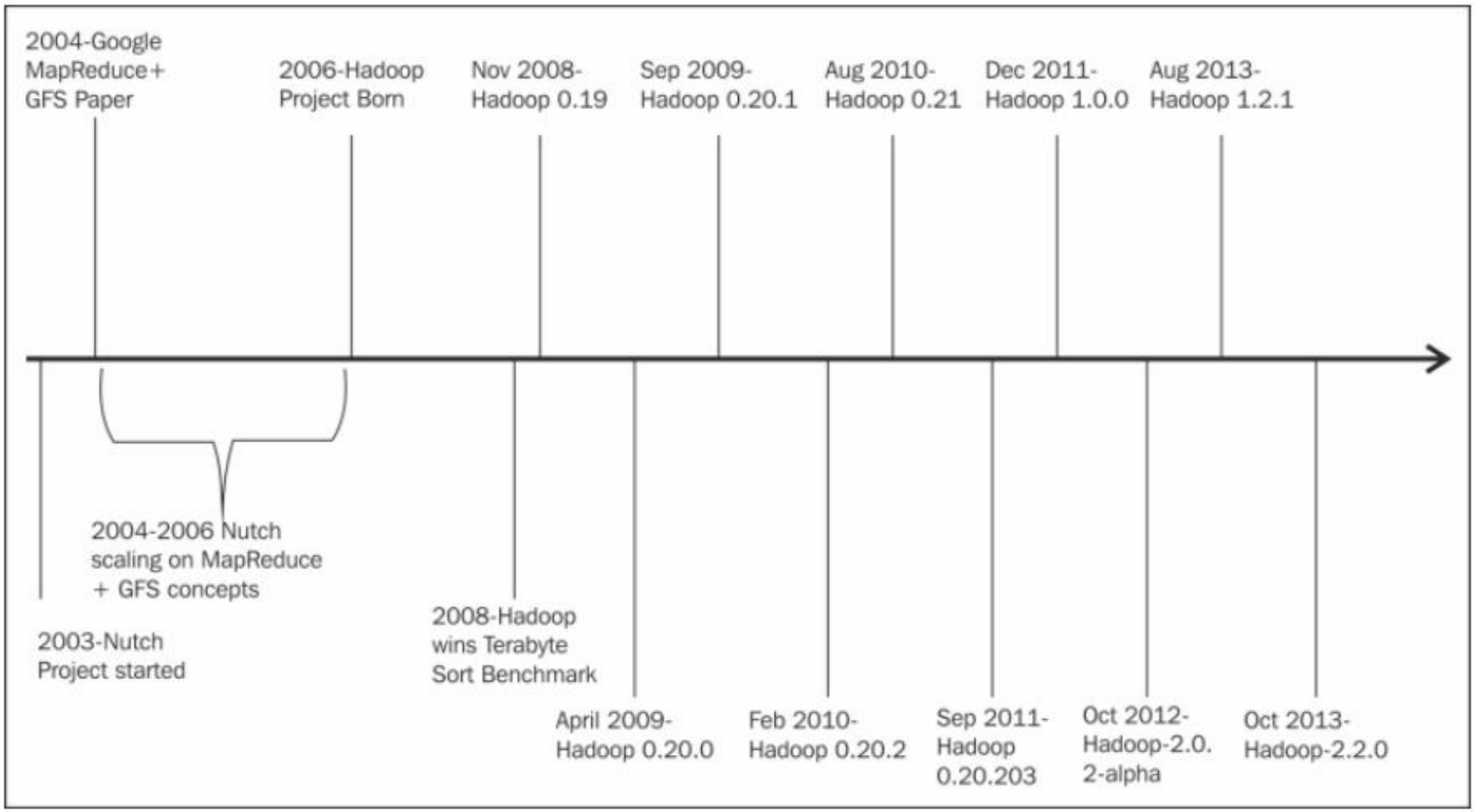
{fay,jeff,sanjay,wilsonh,kerr,m3b,tushar,fikes,gruber}@google.com

Google, Inc.

Abstract

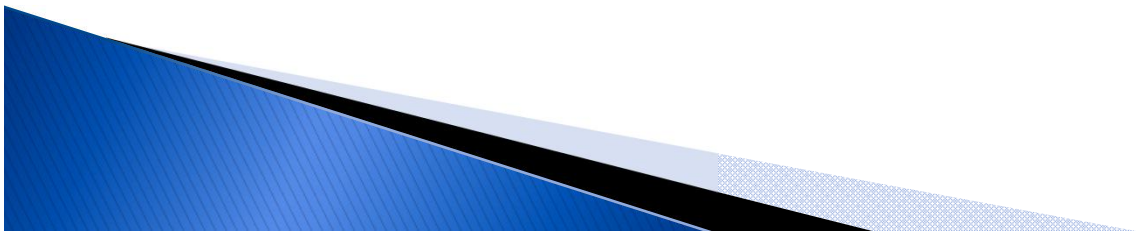


Hadoop Milestones



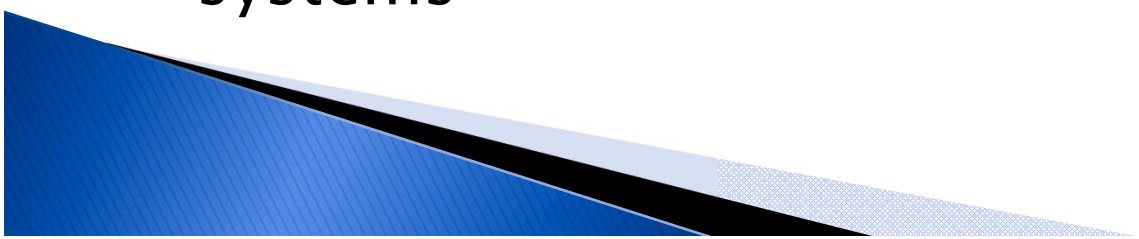
Advantages of Hadoop

- Low cost—Runs on commodity hardware
- Storage flexibility
- Open source community
- Fault tolerant
- Complex data analytics



Uses for Hadoop

- Data-intensive text processing
- Assembly of large genomes
- Graph mining
- Machine learning and data mining
- Large scale social network analysis
- Searching/text mining
- Log processing
- Recommendation systems
- Business intelligence/data warehousing
- Video and image analysis
- Archiving
- Graph creation and analysis
- Pattern recognition
- Risk assessment
- Sentiment analysis

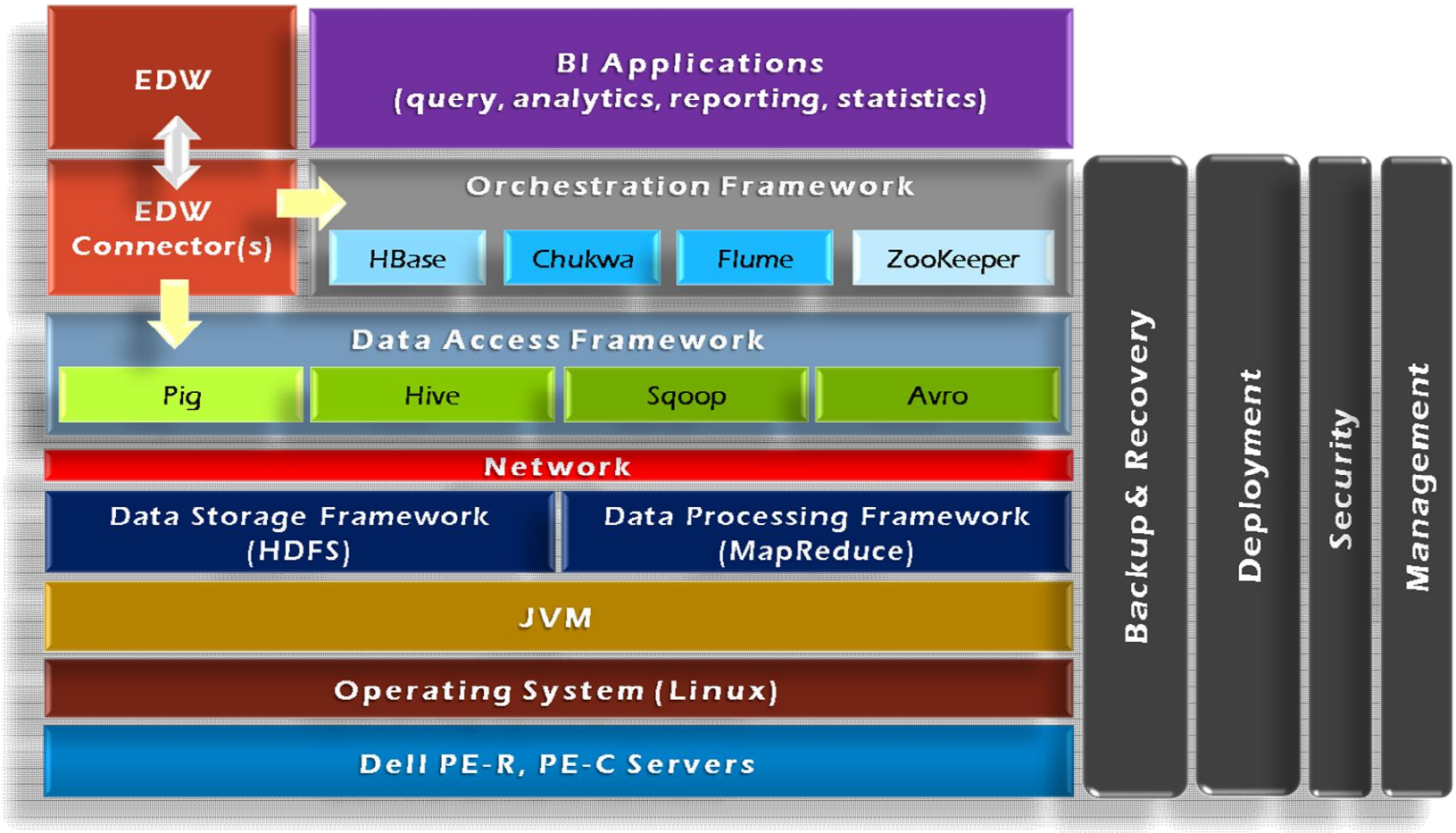


Who Uses Hadoop?

The Hadoop Ecosystem

Layer	Utility/Tool name
Distributed filesystem	Apache HDFS
Distributed programming	Apache MapReduce
	Apache Hive
	Apache Pig
	Apache Spark
NoSQL databases	Apache HBase
Data ingestion	Apache Flume
	Apache Sqoop
	Apache Storm
Service programming	Apache Zookeeper
Scheduling	Apache Oozie
Machine learning	Apache Mahout
System deployment	Apache Ambari

The Hadoop Framework Tools

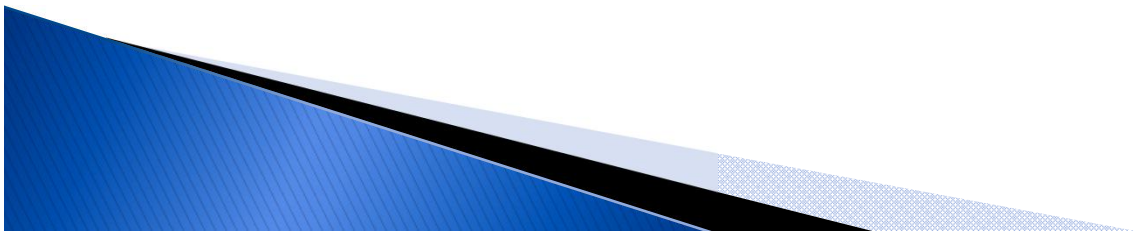


Apache Hadoop (V 2. 6)

Hadoop Common	• Contains Libraries and other modules
HDFS	• Hadoop Distributed File System
Hadoop YARN	• Yet Another Resource Negotiator
Hadoop MapReduce	• A programming model for large scale data processing

Apache Hadoop (V 2. 6)

- Apache Hadoop can be deployed in the following three modes:
  - Standalone
  - Pseudo distributed
  - Distributed

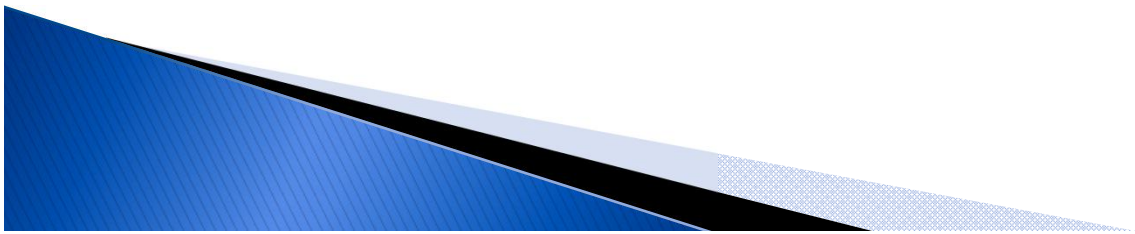


Motivations for Hadoop

- What considerations led to its design

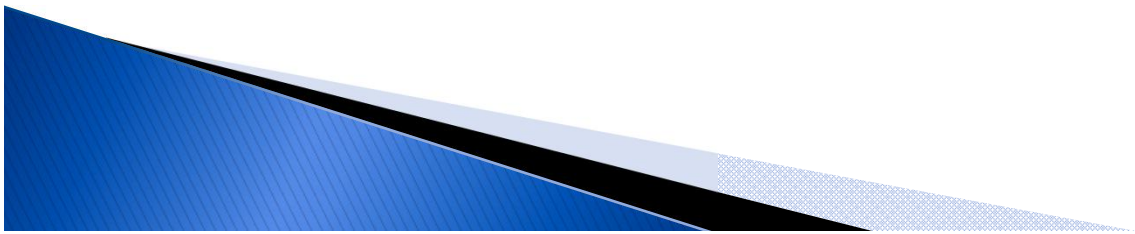
Motivations for Hadoop

- What were the limitations of earlier large-scale computing?
- What requirements should an alternative approach have?
- How does Hadoop address those requirements?

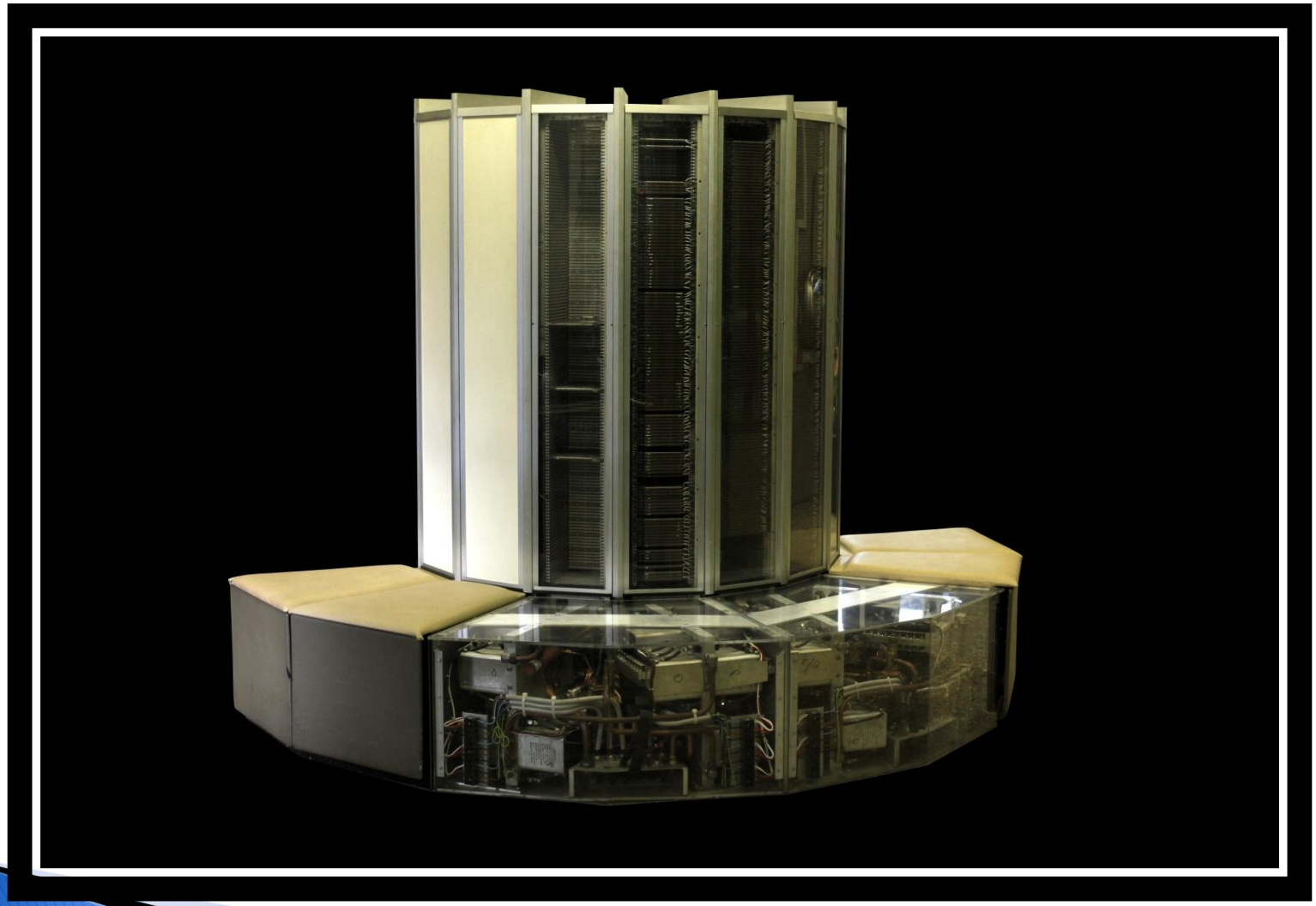


Early Large Scale Computing

- Historically computation was processor-bound
  - Data volume has been relatively small
  - Complicated computations are performed on that data
- Advances in computer technology has historically centered around improving the power of a single machine

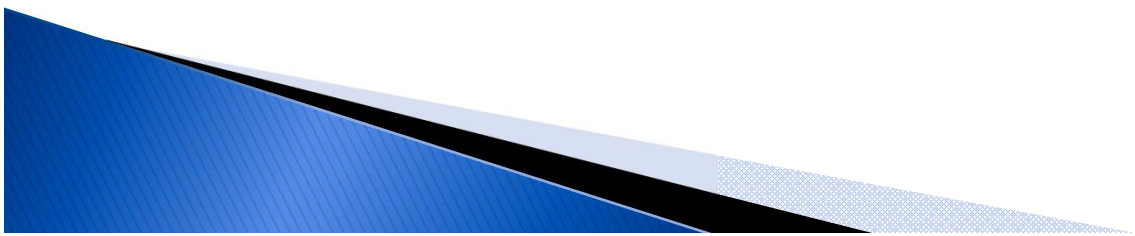


Cray-1



Advances in CPUs

- Moore's Law
  - The number of transistors on a dense integrated circuit doubles every two years
- Single-core computing can't scale with current computing needs



Single-Core Limitation

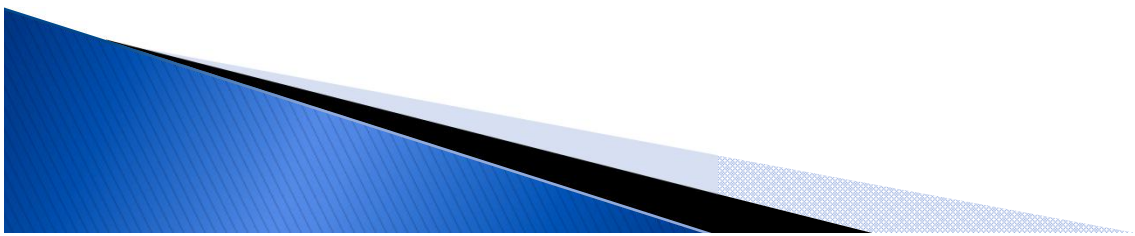
- Power consumption limits the speed increase we get from transistor density

Distributed Systems

- Allows developers to use multiple machines for a single task

Distributed System: Problems

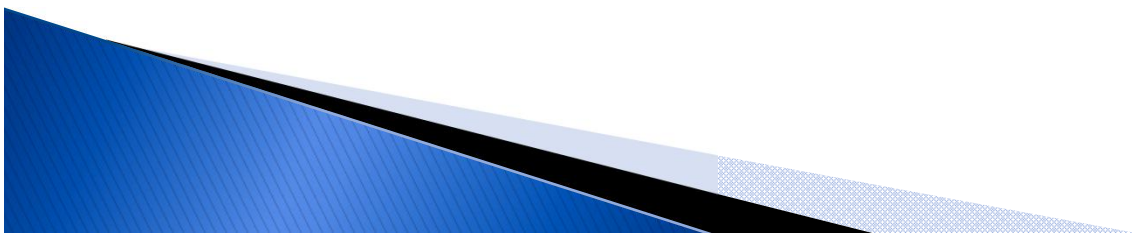
- Programming on a distributed system is much more complex
  - Synchronizing data exchanges
  - Managing a finite bandwidth
  - Controlling computation timing is complicated



Distributed System: Problems

"You know you have a distributed system when the crash of a computer you've never heard of stops you from getting any work done." –Leslie Lamport

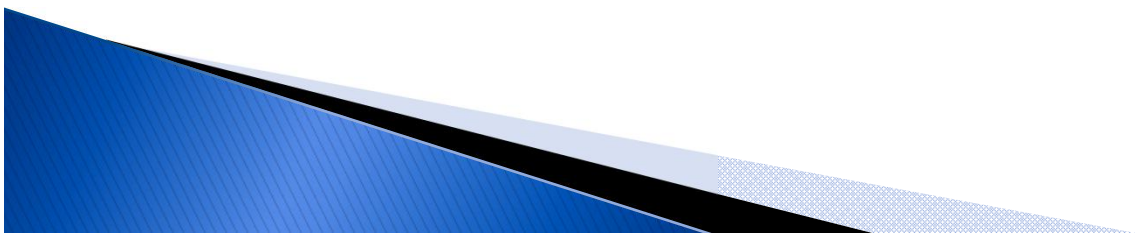
- Distributed systems must be designed with the expectation of failure



Distributed System: Data Storage

- Typically divided into Data Nodes and Compute Nodes
- At compute time, data is copied to the Compute Nodes
- Fine for relatively small amounts of data

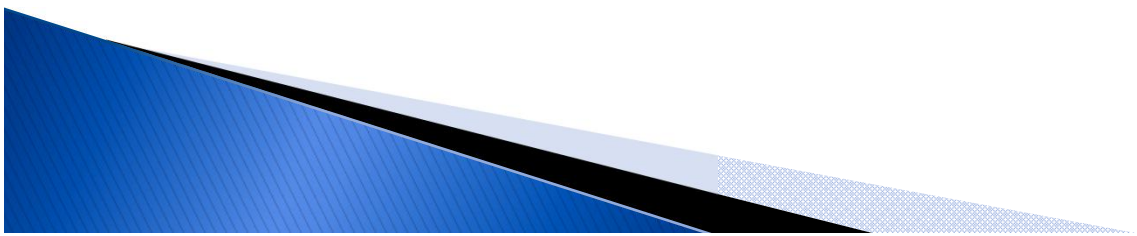
- Modern systems deal with far more data than was gathering in the past



How much data?

- Facebook
  - 500 TB per day
- Yahoo
  - Over 170 PB
- eBay
  - Over 6 PB

- Getting the data to the processors becomes the bottleneck

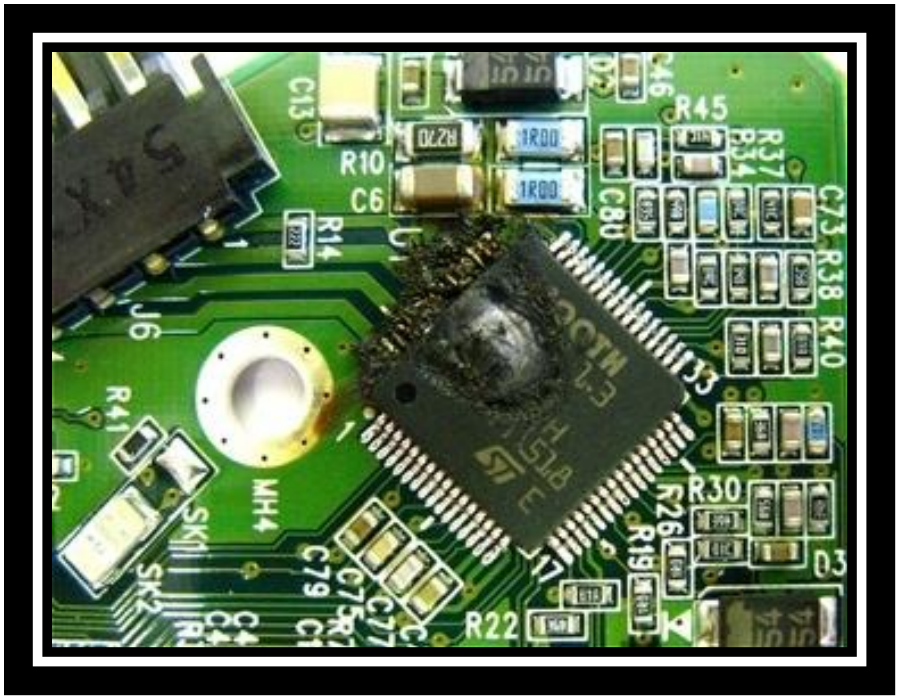


Requirements for Hadoop

- Must support partial failure
- Must be scalable

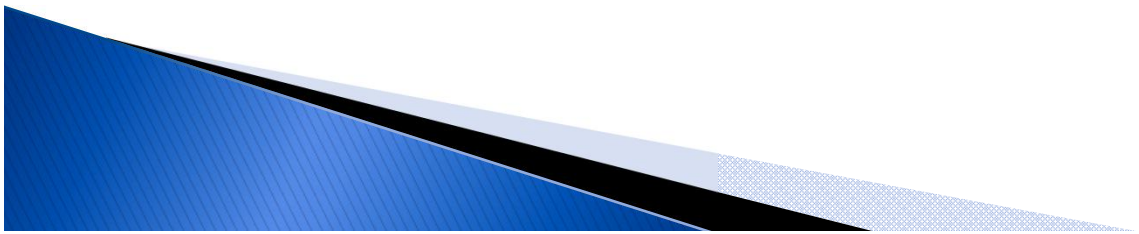
Partial Failures

- Failure of a single component must not cause the failure of the entire system only a degradation of the application performance
- Failure should not result in the loss of any data



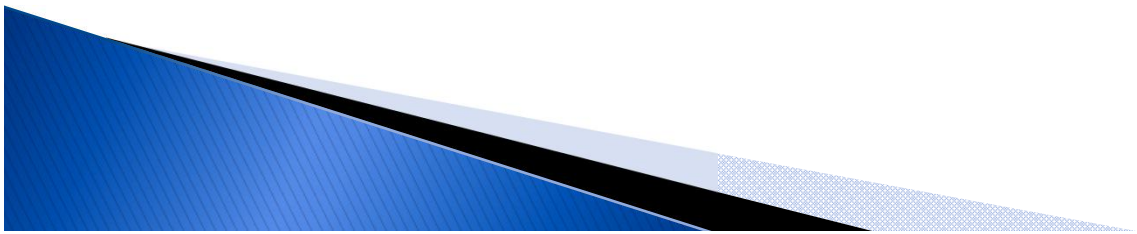
Component Recovery

- If a component fails, it should be able to recover without restarting the entire system
- Component failure or recovery during a job must not affect the final output



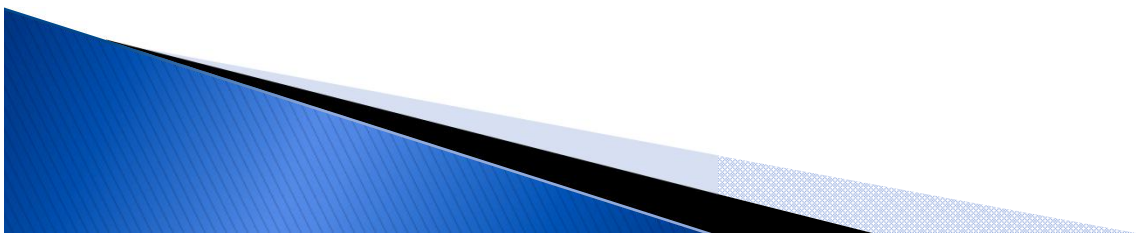
Scalability

- Increasing resources should increase load capacity
- Increasing the load on the system should result in a graceful decline in performance for all jobs
  - Not system failure



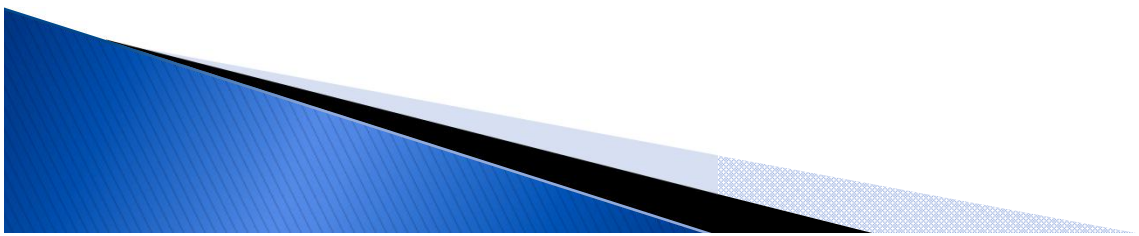
Hadoop

- Based on work done by Google in the early 2000s
  - “The Google File System” in 2003
  - “MapReduce: Simplified Data Processing on Large Clusters” in 2004
- The core idea was to distribute the data as it is initially stored
  - Each node can then perform computation on the data it stores without moving the data for the initial processing



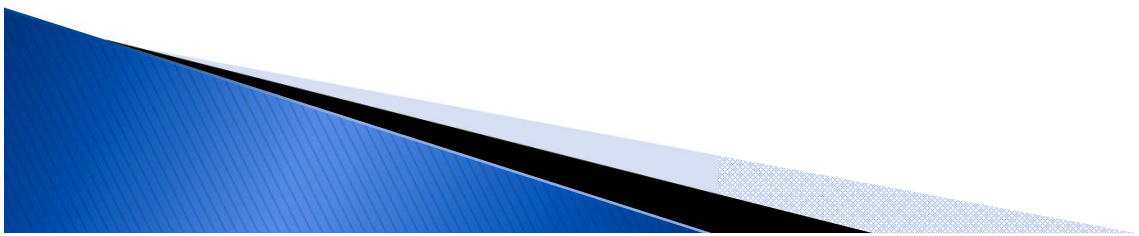
Core Hadoop Concepts

- Applications are written in a high-level programming language
  - No network programming or temporal dependency
- Nodes should communicate as little as possible
  - A “shared nothing” architecture
- Data is spread among the machines in advance
  - Perform computation where the data is already stored as often as possible



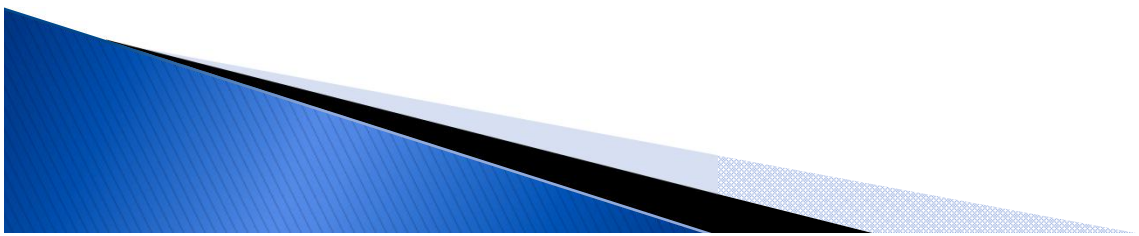
High-Level Overview

- When data is loaded onto the system it is divided into blocks
  - Typically 64MB or 128MB
- Tasks are divided into two phases
  - Map tasks which are done on small portions of data where the data is stored
  - Reduce tasks which combine data to produce the final output
- A master program allocates work to individual nodes



Fault Tolerance

- Failures are detected by the master program which reassigns the work to a different node
- Restarting a task does not affect the nodes working on other portions of the data
- If a failed node restarts, it is added back to the system and assigned new tasks
- The master can redundantly execute the same task to avoid slow running nodes

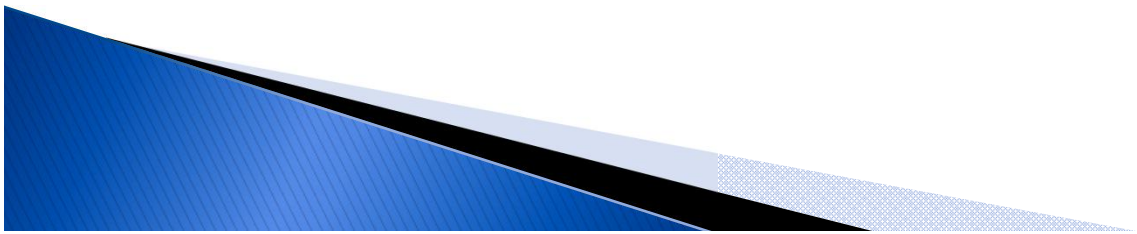


Hadoop Distributed File System

- HDFS

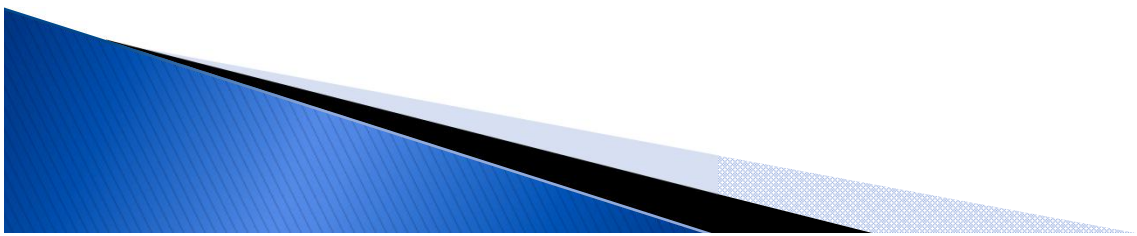
Overview

- Responsible for storing data on the cluster
- Data files are split into blocks and distributed across the nodes in the cluster
- Each block is replicated multiple times



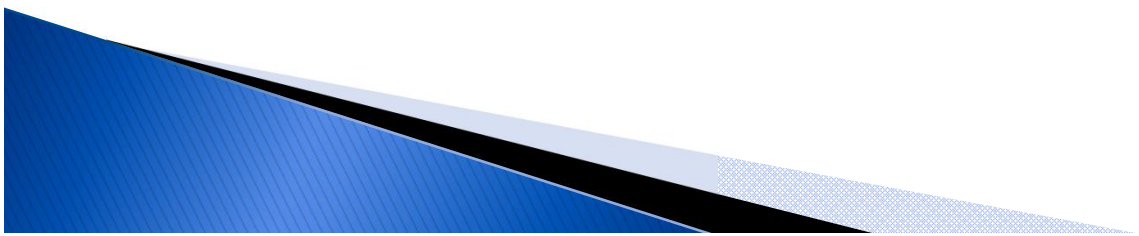
HDFS Basic Concepts

- HDFS is a file system written in Java based on the Google's GFS
- Provides redundant storage for massive amounts of data



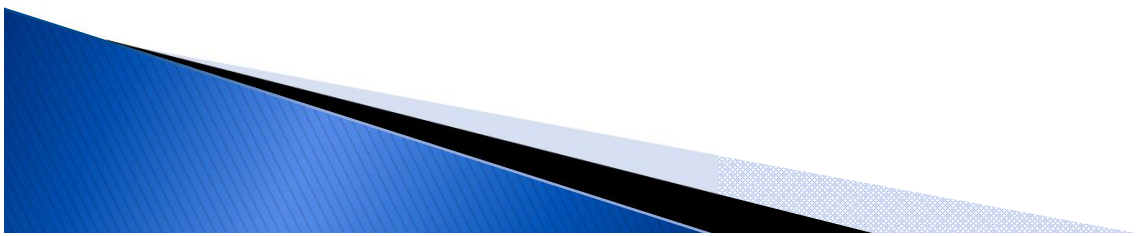
HDFS Basic Concepts

- HDFS works best with a smaller number of large files
  - Millions as opposed to billions of files
  - Typically 100MB or more per file
- Files in HDFS are write once
- Optimized for streaming reads of large files and not random reads



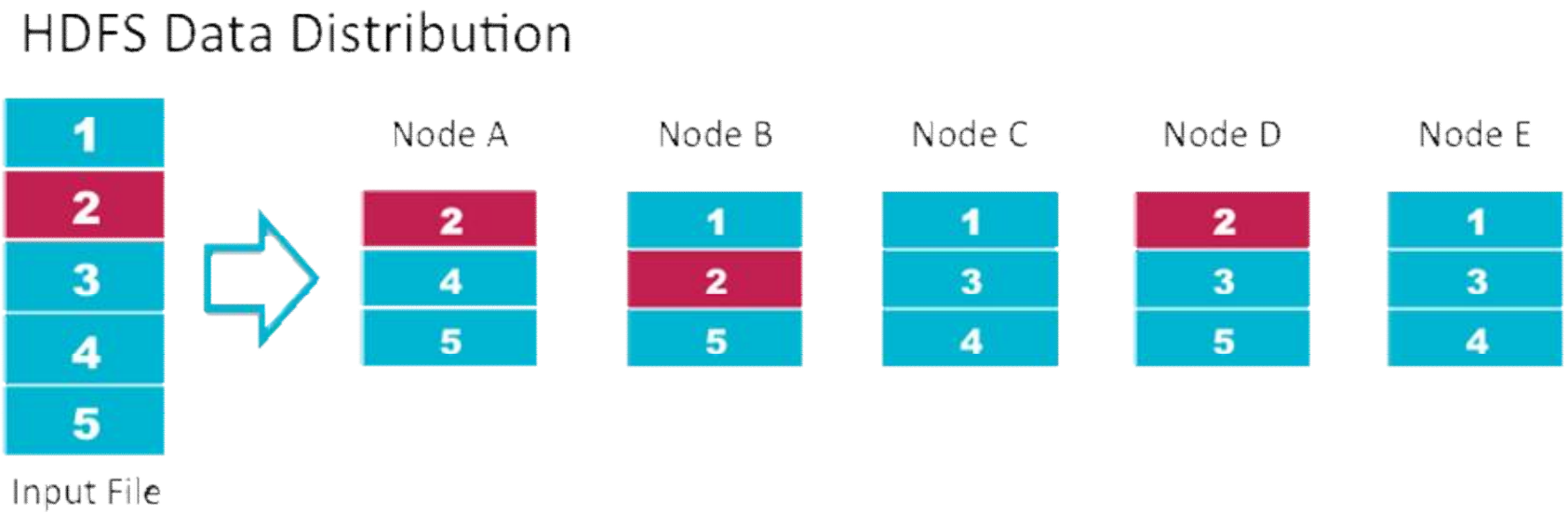
How are Files Stored

- Files are split into blocks
- Blocks are split across many machines at load time
  - Different blocks from the same file will be stored on different machines
- Blocks are replicated across multiple machines
- The NameNode keeps track of which blocks make up a file and where they are stored



Data Replication

- Default replication is 3-fold



Data Retrieval

- When a client wants to retrieve data
  - Communicates with the NameNode to determine which blocks make up a file and on which data nodes those blocks are stored
  - Then communicated directly with the data nodes to read the data