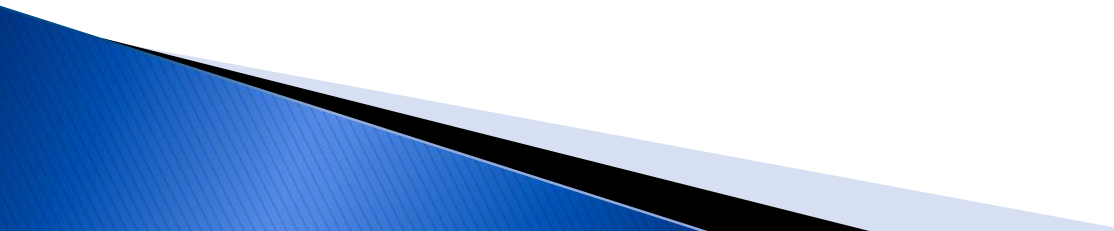
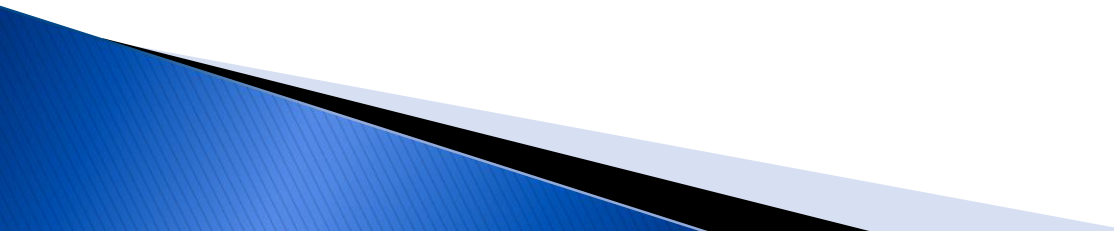




What is Apache Hadoop?

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

Uses for Hadoop

- ▶ Data-intensive text processing
 - ▶ Assembly of large genomes
 - ▶ Graph mining
 - ▶ Machine learning and data mining
 - ▶ Large scale social network analysis
- 

Who Uses Hadoop?



facebook



The New York Times



eHarmony

twitter



NETFLIX



amazon.com



NING



YAHOO!

The Hadoop Ecosystem

Hadoop Common

- Contains Libraries and other modules

HDFS

- Hadoop Distributed File System

Hadoop YARN

- Yet Another Resource Negotiator

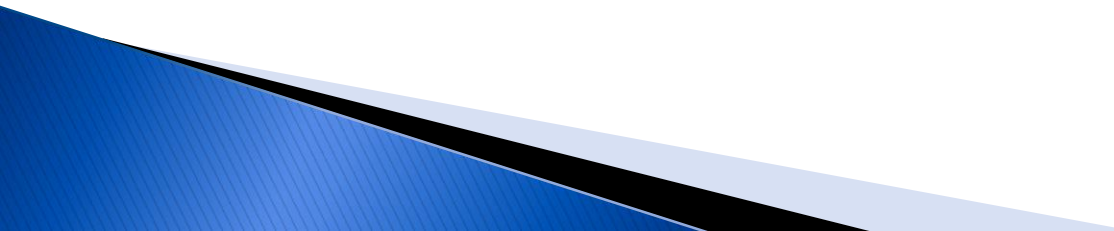
Hadoop
MapReduce

- A programming model for large scale data processing

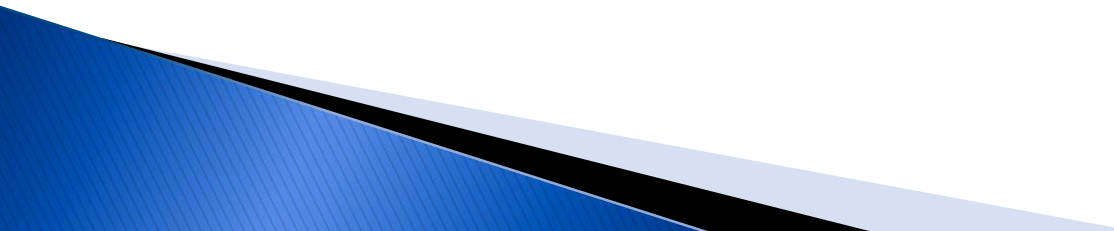
Motivations for Hadoop

- »» What considerations led to its design

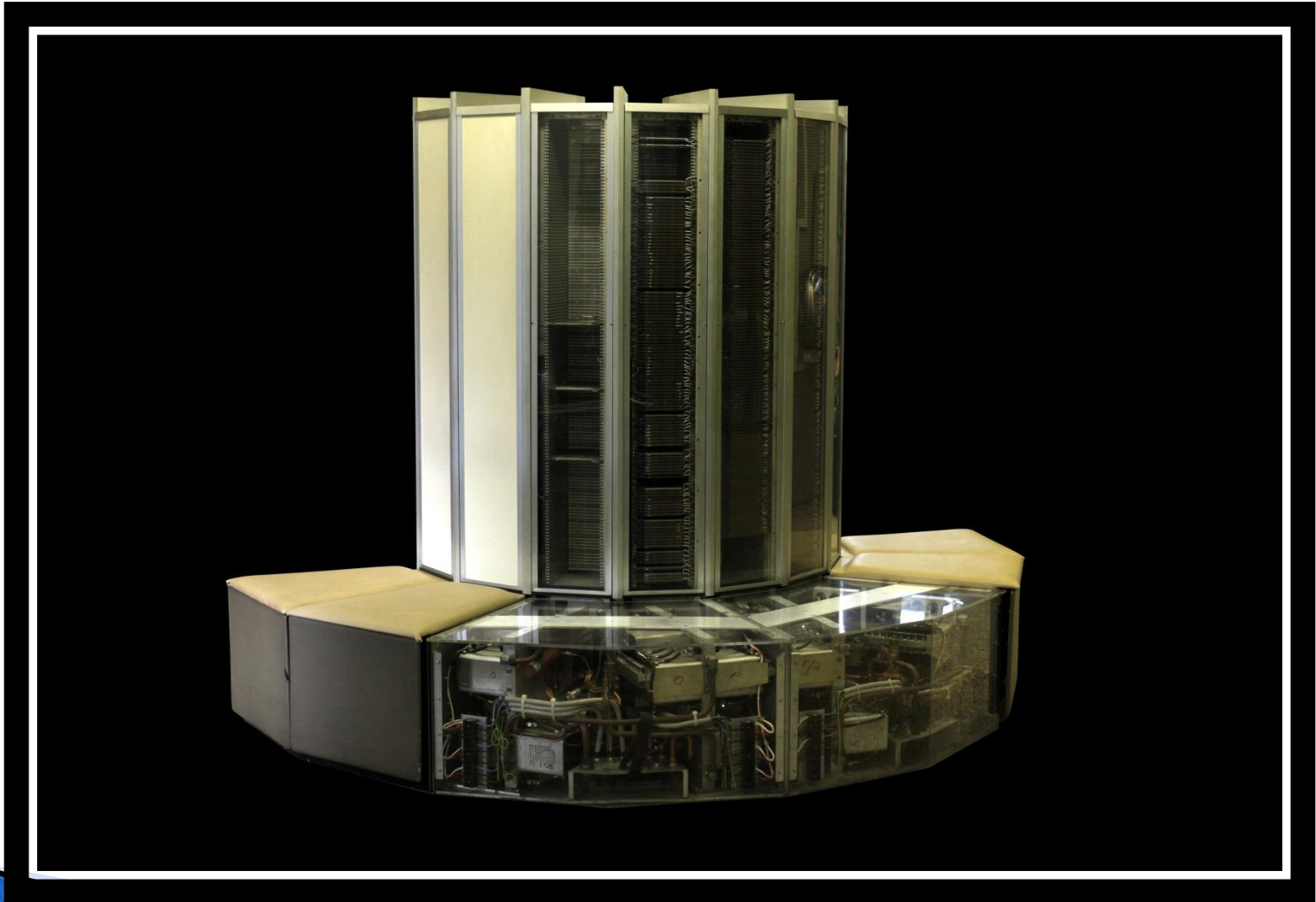
Motivations for Hadoop

- ▶ 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

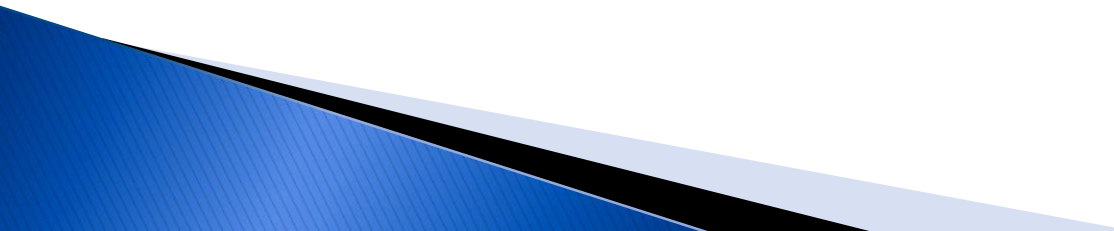
- ▶ Programming on a distributed system is much more complex
 - Synchronizing data exchanges
 - Managing a finite bandwidth
 - Controlling computation timing is complicated

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

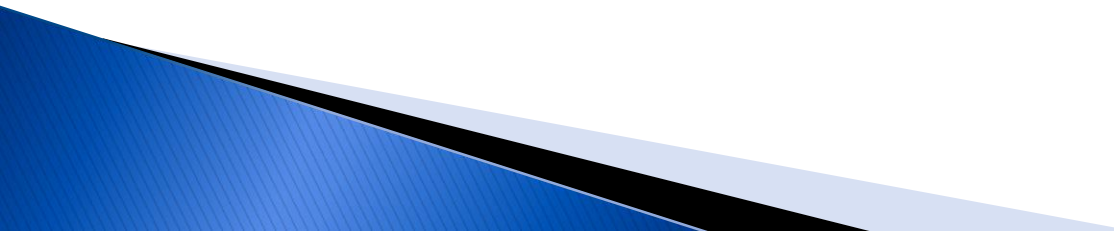
- ▶ Distributed systems must be designed with the expectation of failure

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

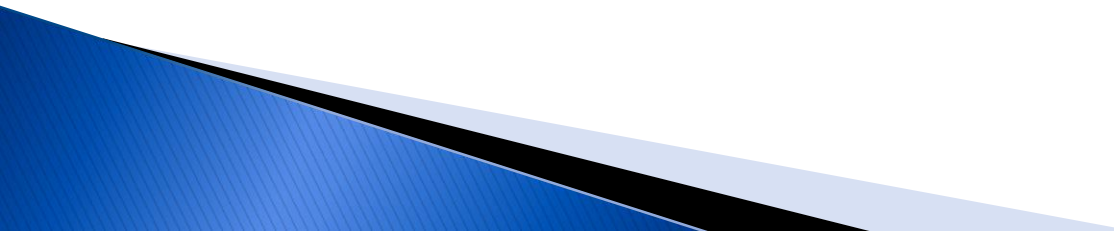


Partial Failures

- ▶ 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
- 

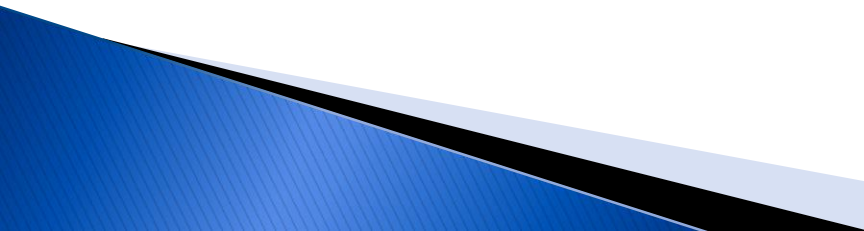
Scalability

- ▶ 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

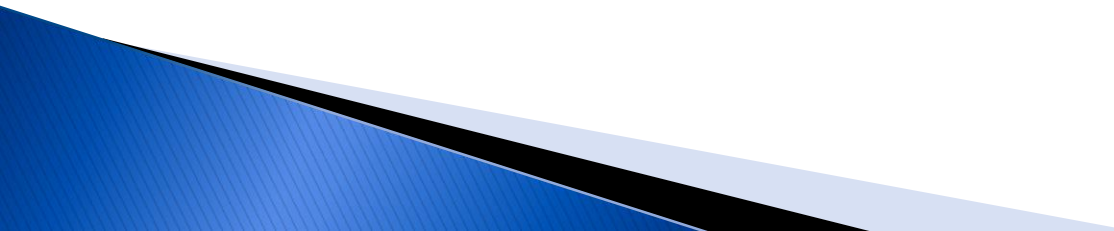
Core Hadoop Concepts

- ▶ 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

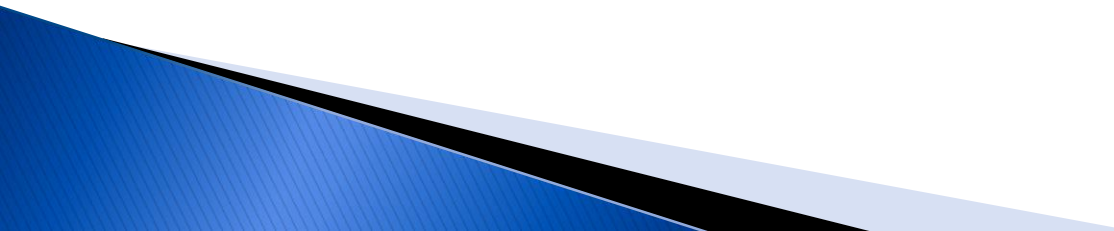
Fault Tolerance

- ▶ 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
- 

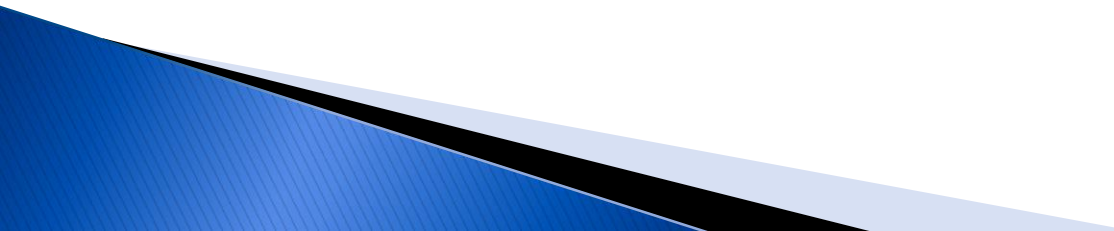
HDFS Basic Concepts

- ▶ 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

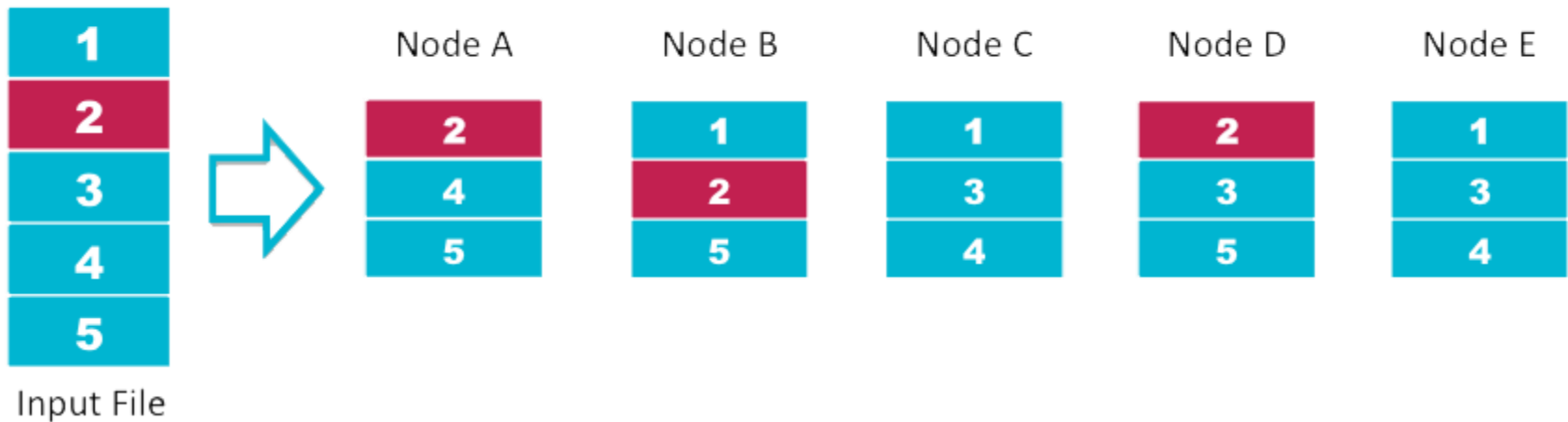
How are Files Stored

- ▶ 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

HDFS Data Distribution



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

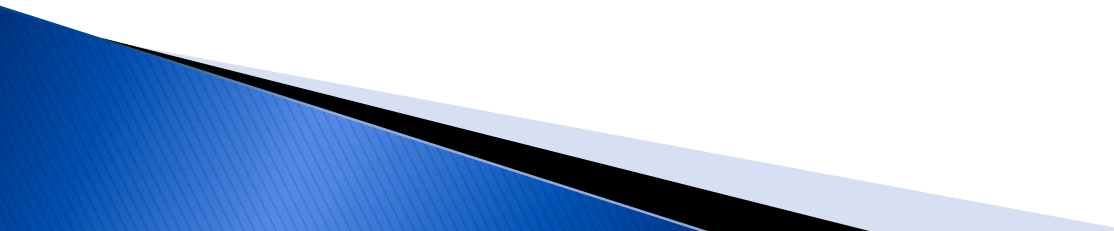
MapReduce

- »» Distributing computation across nodes

MapReduce Overview

- ▶ A method for distributing computation across multiple nodes
- ▶ Each node processes the data that is stored at that node
- ▶ Consists of two main phases
 - Map
 - Reduce

MapReduce Features

- ▶ Automatic parallelization and distribution
 - ▶ Fault-Tolerance
 - ▶ Provides a clean abstraction for programmers to use
- 



The Mapper

- ▶ Reads data as key/value pairs
 - The key is often discarded
- ▶ Outputs zero or more key/value pairs

Shuffle and Sort

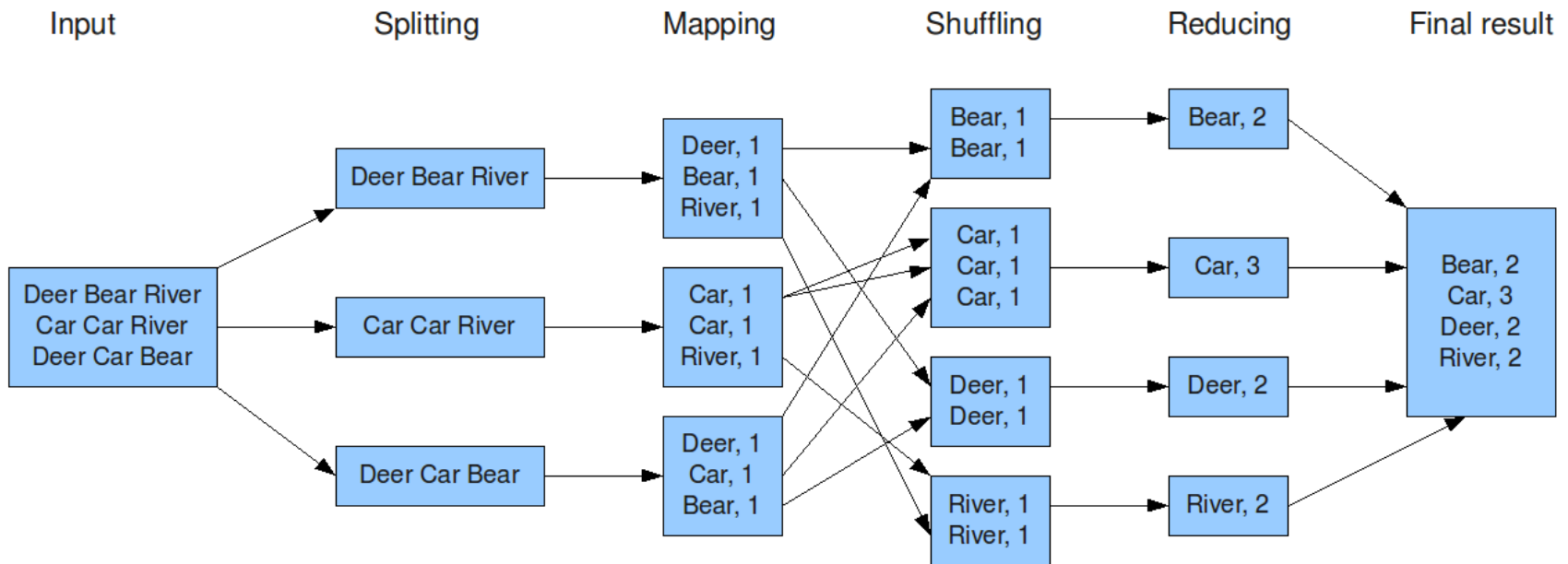
- ▶ Output from the mapper is sorted by key
- ▶ All values with the same key are guaranteed to go to the same machine

The Reducer

- ▶ Called once for each unique key
- ▶ Gets a list of all values associated with a key as input
- ▶ The reducer outputs zero or more final key/value pairs
 - Usually just one output per input key

MapReduce: Word Count

The overall MapReduce word count process



Anatomy of a Cluster

- »» What parts actually make up a Hadoop cluster

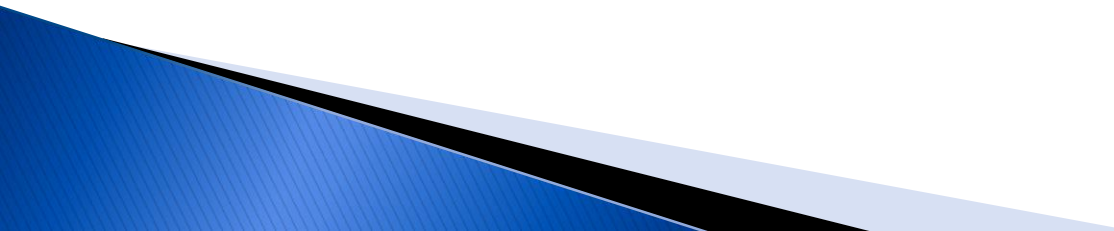
Overview

- ▶ **NameNode**
 - Holds the metadata for the HDFS
- ▶ **Secondary NameNode**
 - Performs housekeeping functions for the NameNode
- ▶ **DataNode**
 - Stores the actual HDFS data blocks
- ▶ **JobTracker**
 - Manages MapReduce jobs
- ▶ **TaskTracker**
 - Monitors individual Map and Reduce tasks

The NameNode

- ▶ Stores the HDFS file system information in a fsimage
- ▶ Updates to the file system (add/remove blocks) do not change the fsimage file
 - They are instead written to a log file
- ▶ When starting the NameNode loads the fsimage file and then applies the changes in the log file

The Secondary NameNode

- ▶ **NOT** a backup for the NameNode
 - ▶ Periodically reads the log file and applies the changes to the fsimage file bringing it up to date
 - ▶ Allows the NameNode to restart faster when required
- 

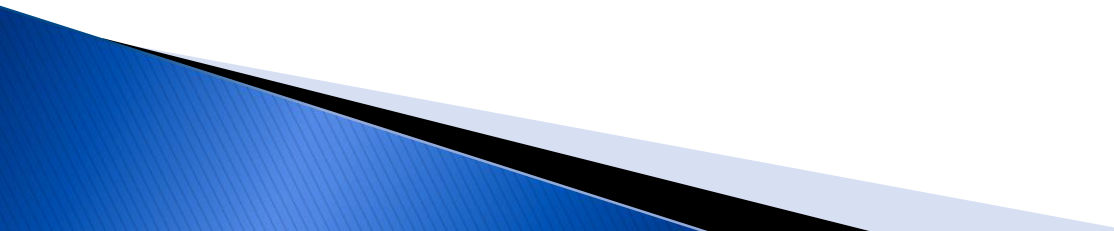
JobTracker and TaskTracker

- ▶ JobTracker
 - Determines the execution plan for the job
 - Assigns individual tasks
- ▶ TaskTracker
 - Keeps track of the performance of an individual mapper or reducer

Hadoop Ecosystem

»» Other available tools

Why do these tools exist?

- ▶ MapReduce is very powerful, but can be awkward to master
 - ▶ These tools allow programmers who are familiar with other programming styles to take advantage of the power of MapReduce
- 

Other Tools

- ▶ Hive
 - Hadoop processing with SQL
- ▶ Pig
 - Hadoop processing with scripting
- ▶ Cascading
 - Pipe and Filter processing model
- ▶ HBase
 - Database model built on top of Hadoop
- ▶ Flume
 - Designed for large scale data movement