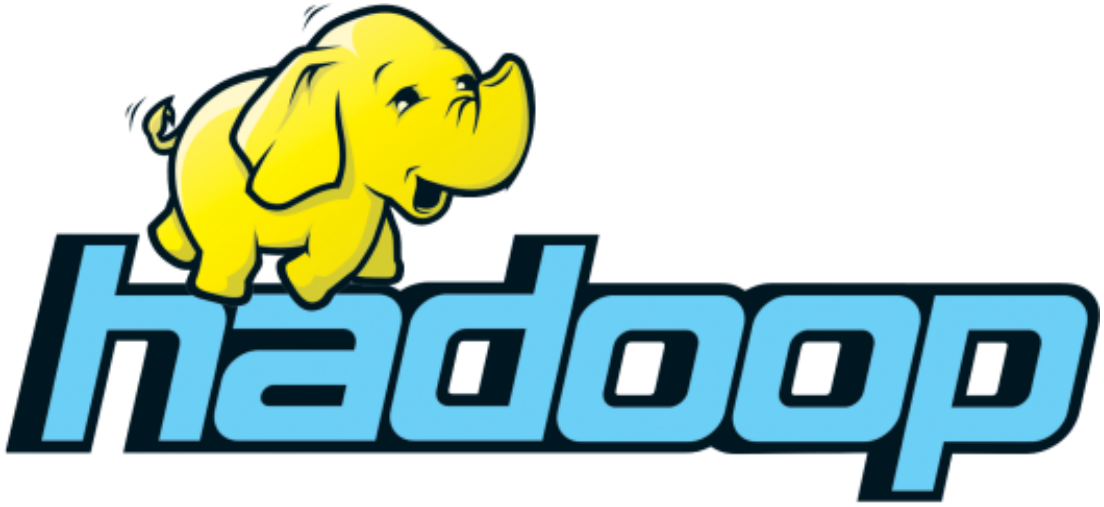
BigData Analytics



Agenda

- **Introduction to BIG DATA**
  - Characteristics of BIG DATA
- **Introduction to Hadoop**
  - Distributed System
  - Why Hadoop over legacy RDBMS
  - Uses of Hadoop
  - Hadoop Users
  - Hadoop EcoSystem
- **HDFS**
  - HDFS Hands On
- **Map Reduce(MR)**
  - MR Example

What is BigData?

- 'Big Data' is a term used to describe collection of data that is huge in size and yet growing exponentially with time

- **Categories of BigData**

- **Structured data** : Relational data.
- **Semi Structured data** : XML data.
- **Unstructured data** : Word, PDF, Text, Media Logs.

BigData: Characteristics

- Volume– 'Volume' is one characteristic which needs to be considered while dealing with 'Big Data'
- Variety– Data in the form of emails, photos, videos, monitoring devices, PDFs, audio, etc. is also being considered in the analysis applications. This variety of unstructured data poses certain issues for storage, mining and analysing data
- Velocity– Velocity deals with the speed at which data flows in from sources like business processes, application logs, networks and social media sites, sensors, Mobile devices, etc. The flow of data is massive and continuous
- Variability– This refers to the inconsistency which can be shown by the data at times, thus hampering the process of being able to handle and manage the data effectively

What is Apache Hadoop?

- Apache top level project, open-source implementation of frameworks for reliable, scalable, distributed computing and data storage
- It is a flexible and highly-available architecture for large scale computation and data processing on a network of commodity hardware

Facts

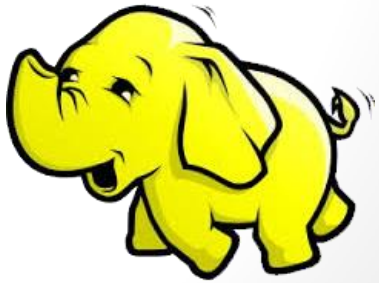
Doug Cutting

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

Cutting named the program after his son's toy elephant

The project was funded by Yahoo.

2006: Yahoo gave the project to Apache Software Foundation.



Application of Hadoop

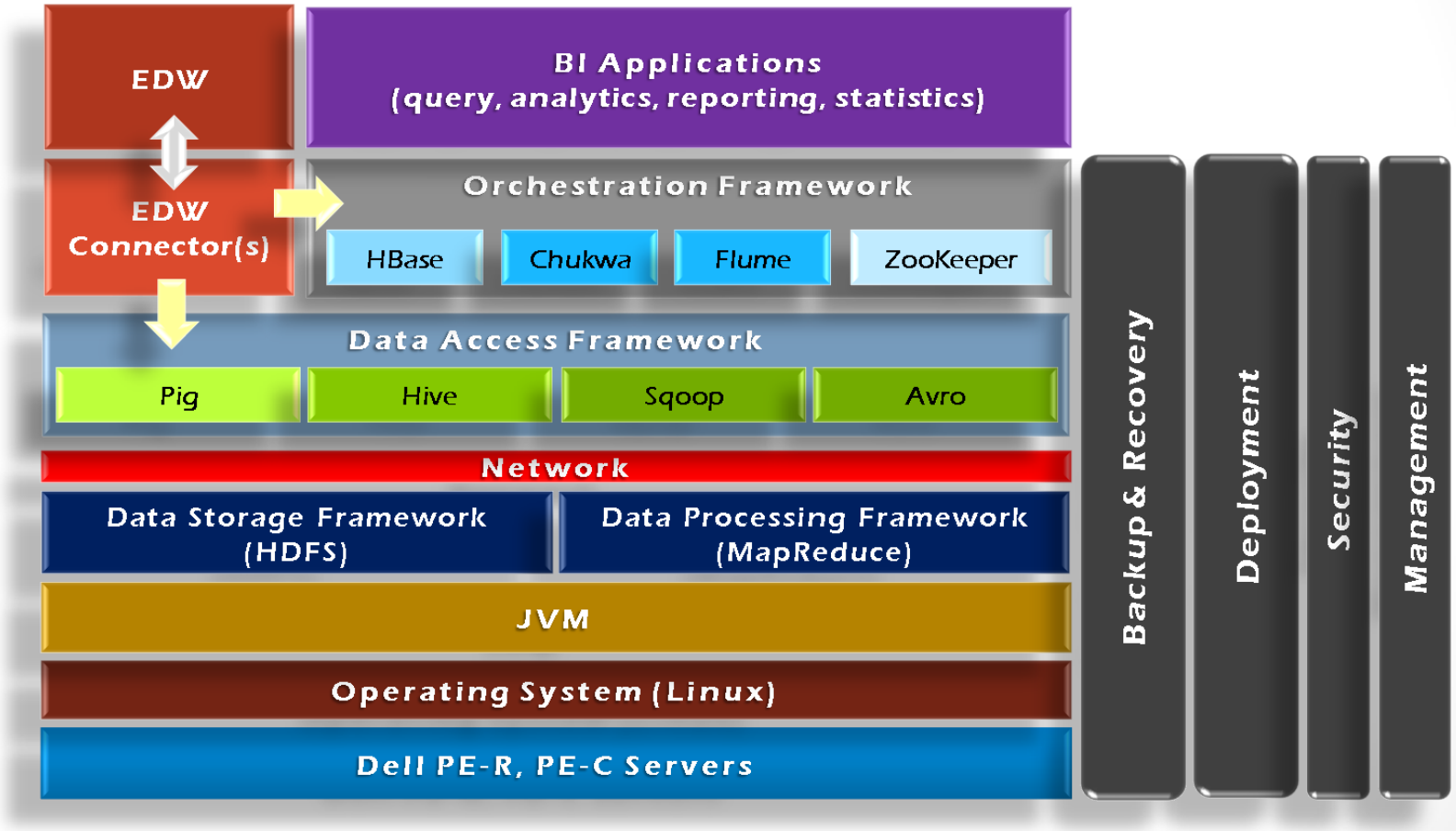
- Data-intensive text processing
- Real Time Data Processing
- Machine learning and data mining
- Large scale social network analysis

Hadoop Users

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

Hadoop Ecosystem

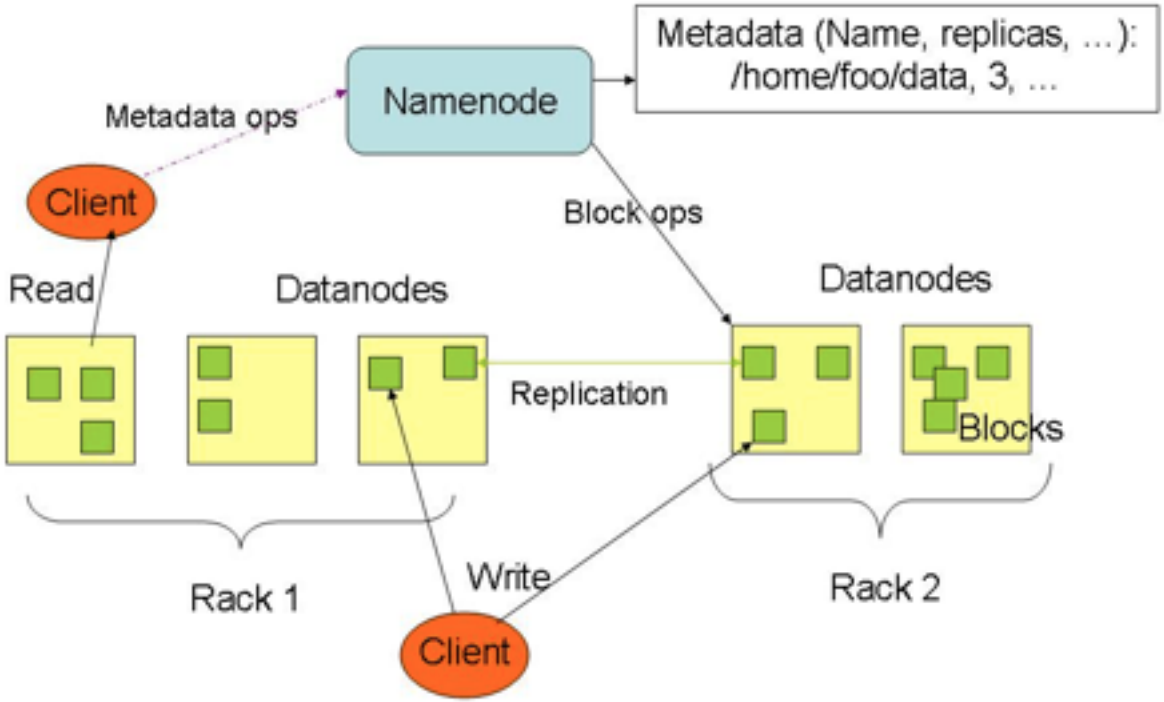


Hadoop Architecture

- Distributed, with some centralization
- Main nodes of cluster are where most of the computational power and storage of the system lies
- Main nodes run TaskTracker to accept and reply to MapReduce tasks, and also DataNode to store needed blocks closely as possible
- Central control node runs NameNode to keep track of HDFS directories & files, and JobTracker to dispatch compute tasks to TaskTracker

- Written in Java, also supports Python and Ruby

Hadoop – MR


Metadata (Name, replicas, …):
/home/foo/data, 3, …
Namenode
Metadata ops
Client
Block ops
Read
Datanodes
Datanodes
Replication
Blocks
Rack 1
Write
Rack 2
Client

Hadoop - MR

- **MapReduce Engine:**

- JobTracker & TaskTracker

- JobTracker splits up data into smaller tasks(“Map”) and sends it to the TaskTracker process in each node

- TaskTracker reports back to the JobTracker node and reports on job progress, sends data (“Reduce”) or requests new jobs

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

Hadoop Characteristics

- **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

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

HDFS

- 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

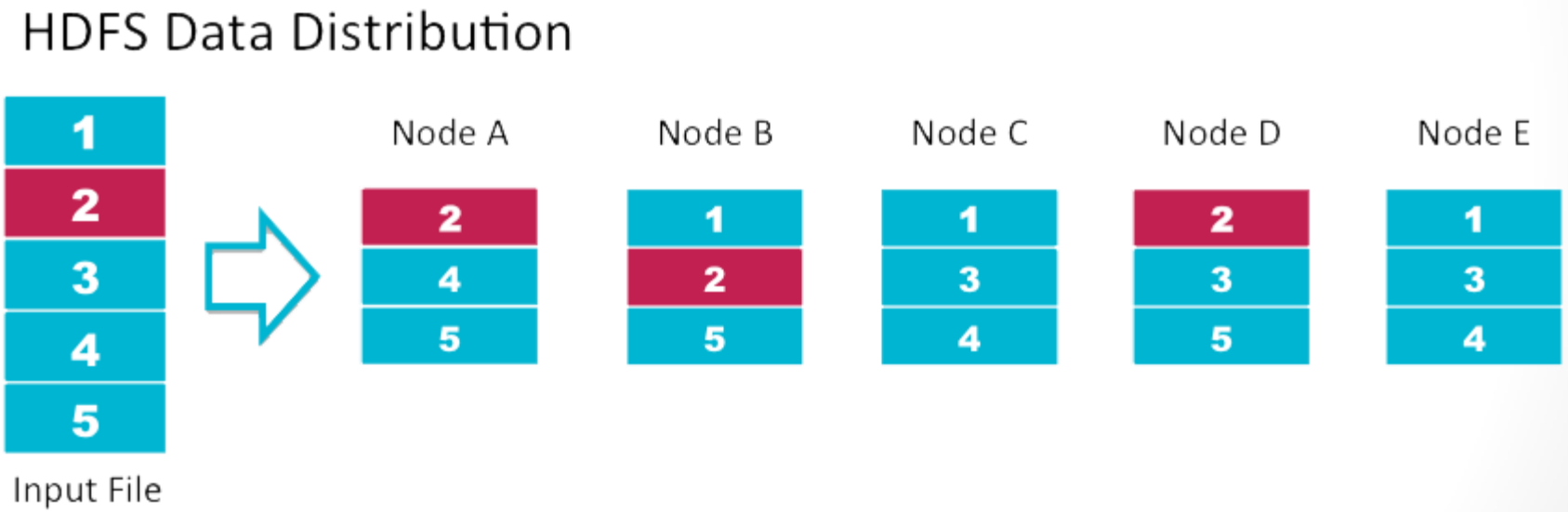
- HDFS works best with a smaller number of large files
  - Millions as opposed to billions of files
  - Typically 100MB or more per file
- Files in HDFS are write once
- Optimized for streaming reads of large files and not random reads

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



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

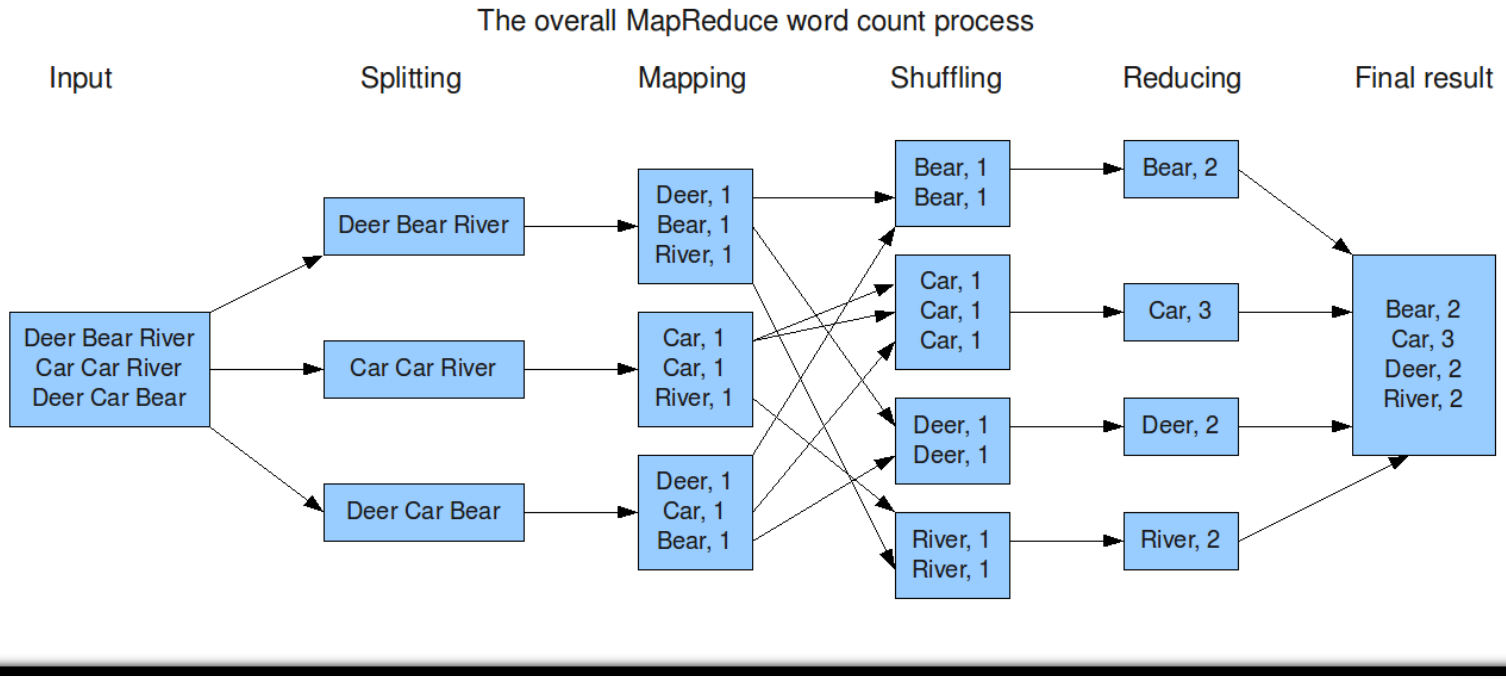
The Reducer

- 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

MR Architecture

- 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

MapReduce: Word Count



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

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

Thank you!!!