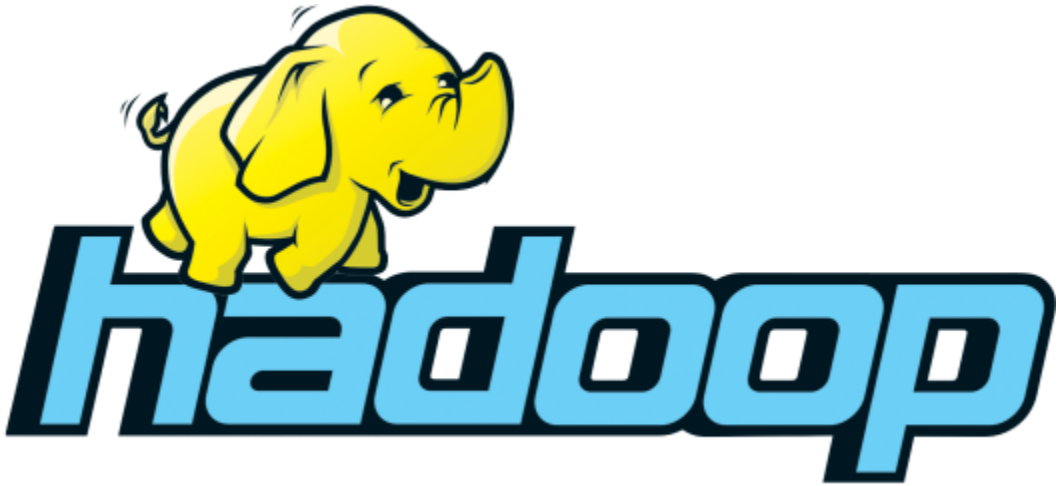
Big Data Technologies

Jnaneshwar Bohara

Chapter 6: Case Study Hadoop



Terminology

Google calls it:	**Hadoop equivalent:**
MapReduce	Hadoop
GFS	HDFS
Bigtable	HBase
Chubby	Zookeeper

Introduction

- 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.
- 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
- 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 Apache Software Foundation.

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.

Hadoop - Why ?

- Need to process huge datasets on large clusters of computers
- Very expensive to build reliability into each application
- Nodes fail every day
  - Failure is expected, rather than exceptional
  - The number of nodes in a cluster is not constant
- Need a common infrastructure
  - Efficient, reliable, easy to use
  - Open Source, Apache Licence

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?

What considerations led to its design

MOTIVATIONS FOR HADOOP

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

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
- Getting the data to the processors becomes the bottleneck

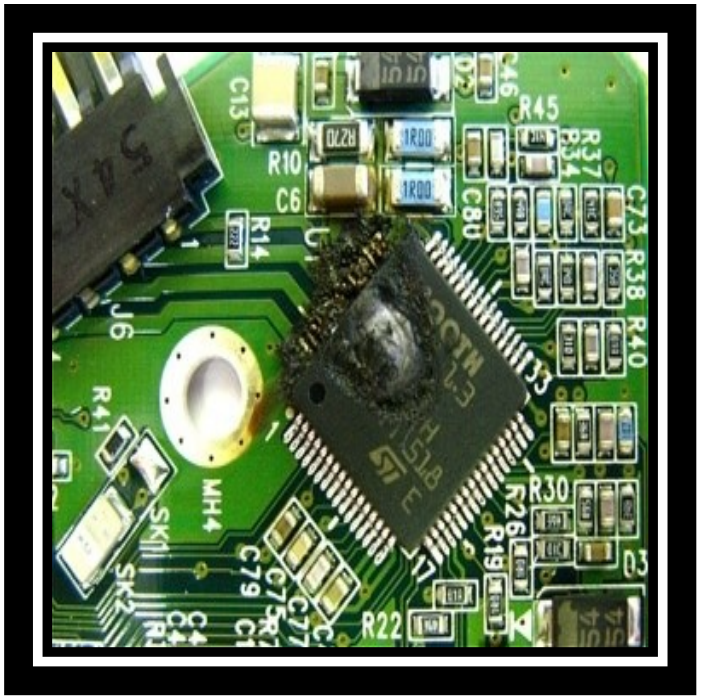
Requirements for Hadoop

- 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

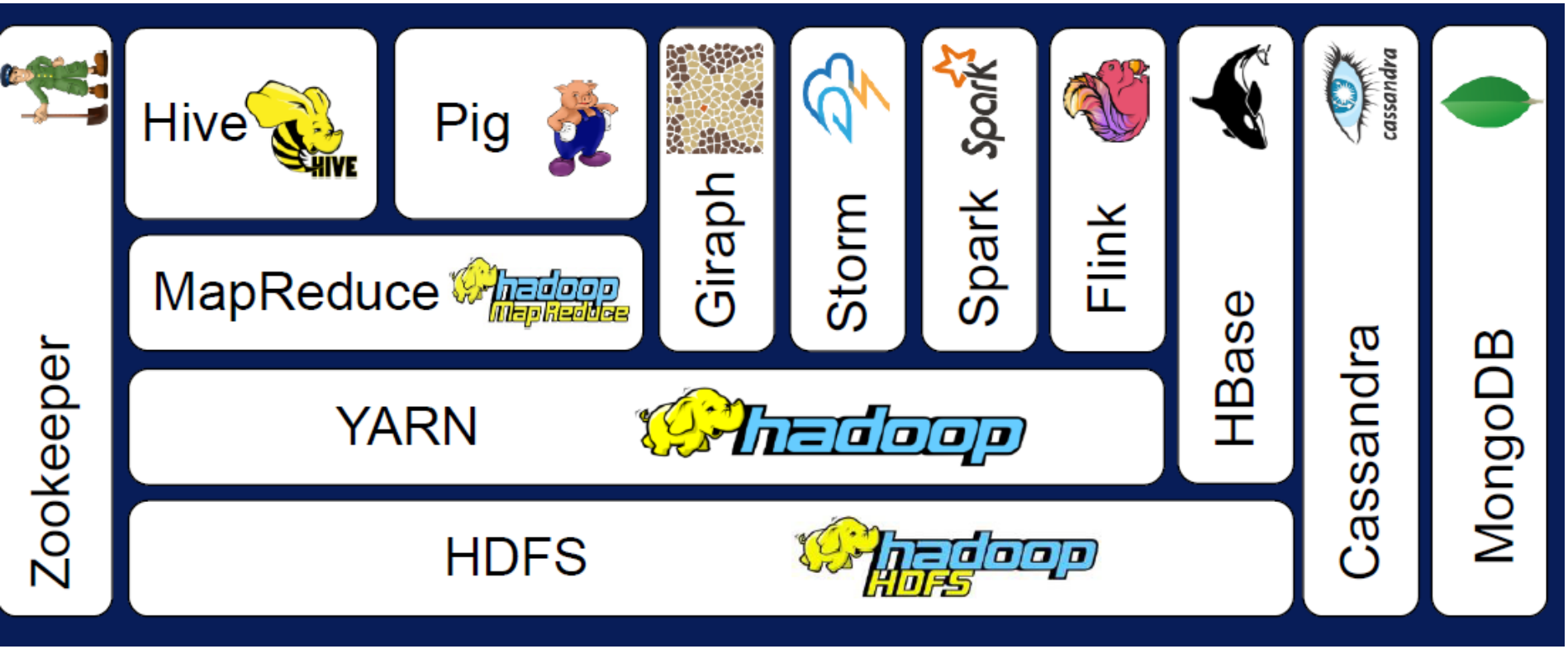
Fault Tolerance

- 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

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

The Hadoop Ecosystem



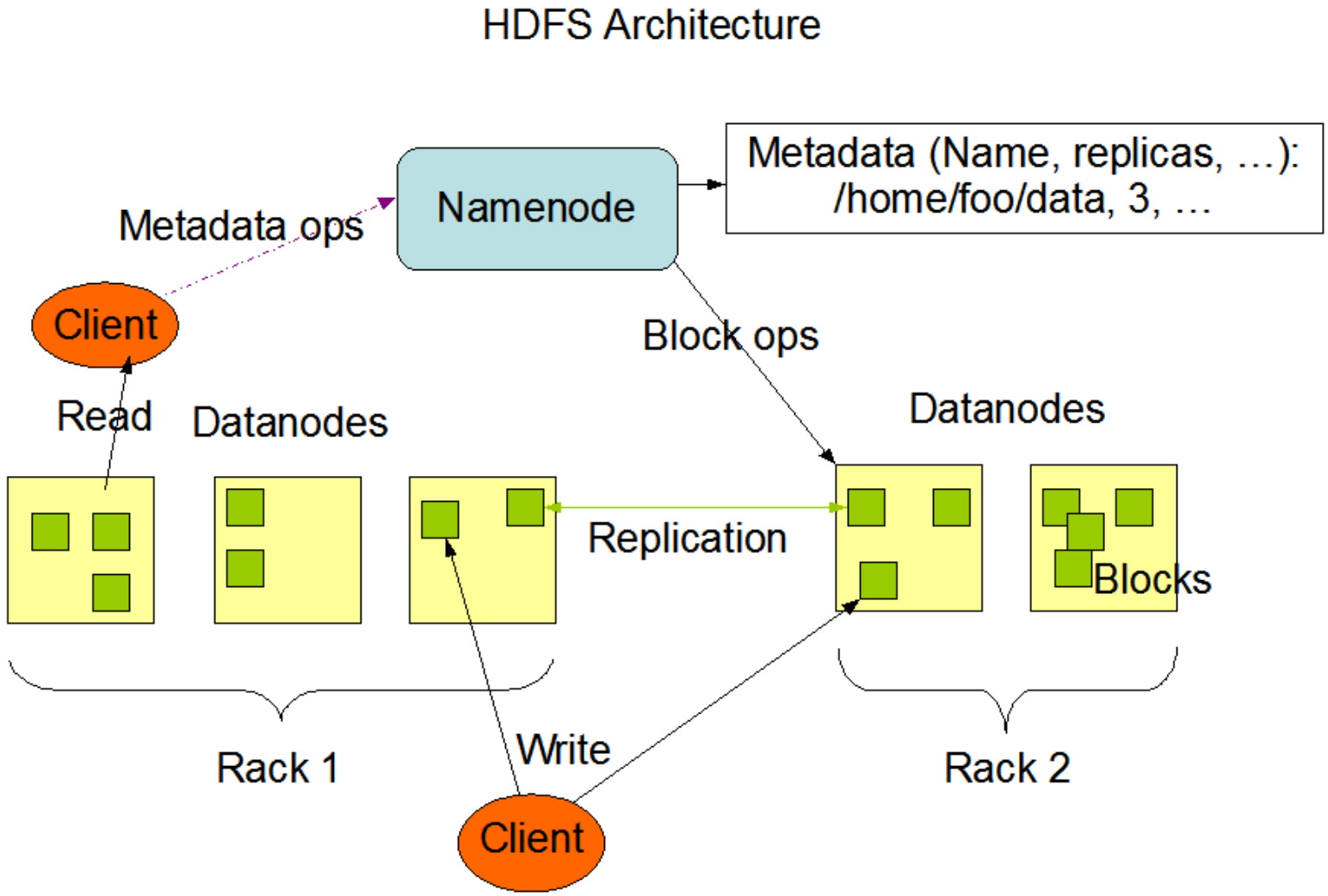
Hadoop Common

- The common utilities that support the other Hadoop modules.
- It is an essential part or module of the Apache Hadoop Framework, along with the Hadoop Distributed File System (HDFS), Hadoop YARN and Hadoop MapReduce.
- Hadoop Common is also known as Hadoop Core.

HDFS

- Part of the Apache Hadoop Core project.
- HDFS is a file system written in Java based on the Google's GFS
- Responsible for storing data on the cluster
- Provides redundant storage for massive amounts of data
- Data files are split into blocks and distributed across the nodes in the cluster
- Each block is replicated multiple times
- HDFS works best with a smaller number of large files
  - Millions as opposed to billions of files
  - Typically 100MB or more per file

HDFS Architecture

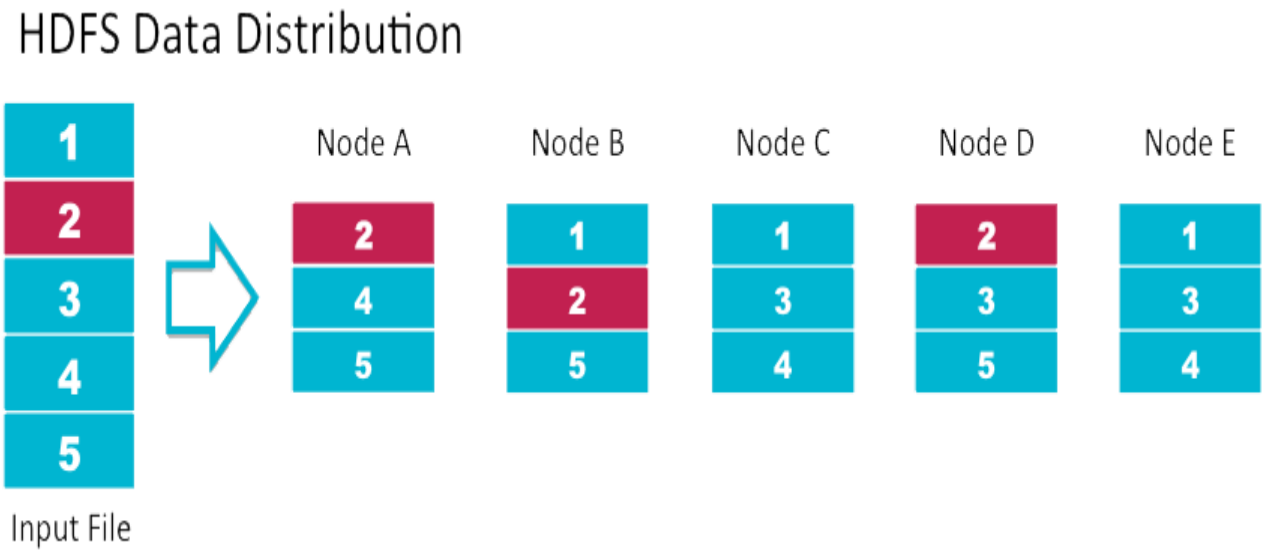


HDFS: How are Files Stored

- 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

HDFS: Data Replication

- Default replication is 3-fold



HDFS: 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

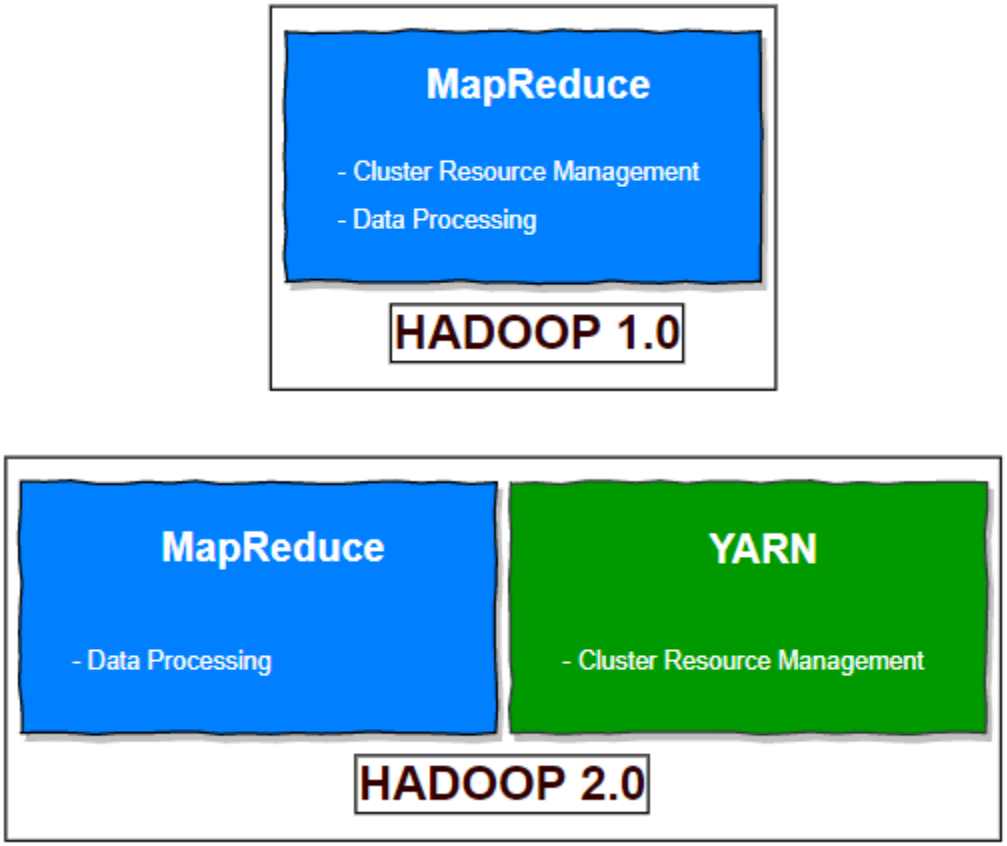
YARN

- Yet Another Resource Negotiator
- YARN Application Resource Negotiator(Recursive Acronym)
- Remedies the scalability shortcomings of “classic” MapReduce
- Is more of a general purpose framework of which classic mapreduce is one application.

Hadoop YARN

- A framework for job scheduling and cluster resource management.
- In 2012, YARN became a sub-project of the larger Apache Hadoop project.
- Sometimes called MapReduce 2.0, YARN is a software rewrite that decouples MapReduce's resource management and scheduling capabilities from the data processing component, enabling Hadoop to support more varied processing approaches and a broader array of applications.

Yet Another Resource Negotiator (YARN)



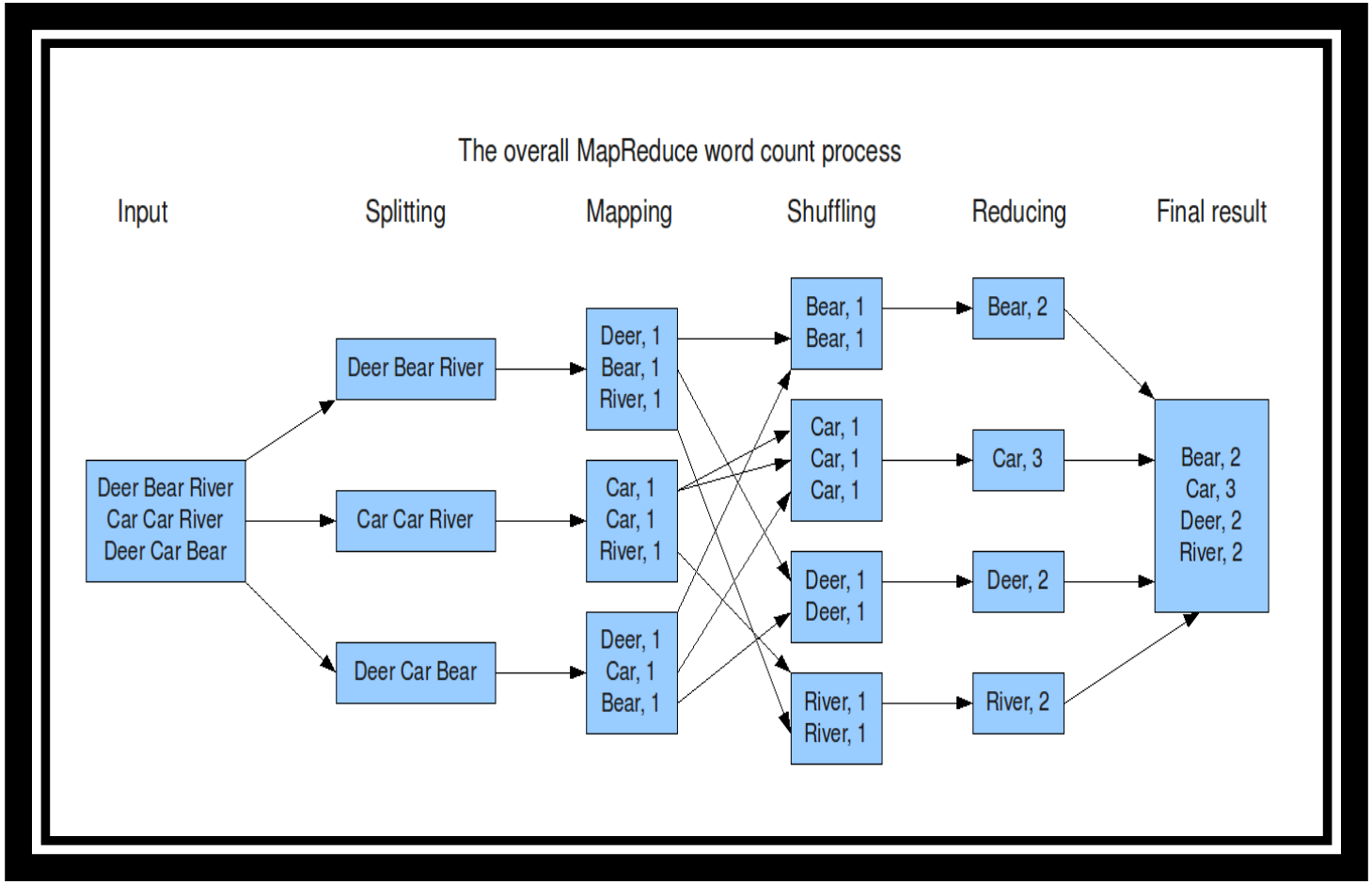
Yet Another Resource Negotiator (YARN)

- Yet Another Resource Negotiator (YARN) distributes a MapReduce program across different nodes and takes care of coordination
- Three important services
  - ResourceManager: a global YARN service that receives and runs applications (e.g., a MapReduce job) on the cluster
  - JobHistoryServer: keeps a log of all finished jobs
  - NodeManager: responsible to oversee resource consumption on a node

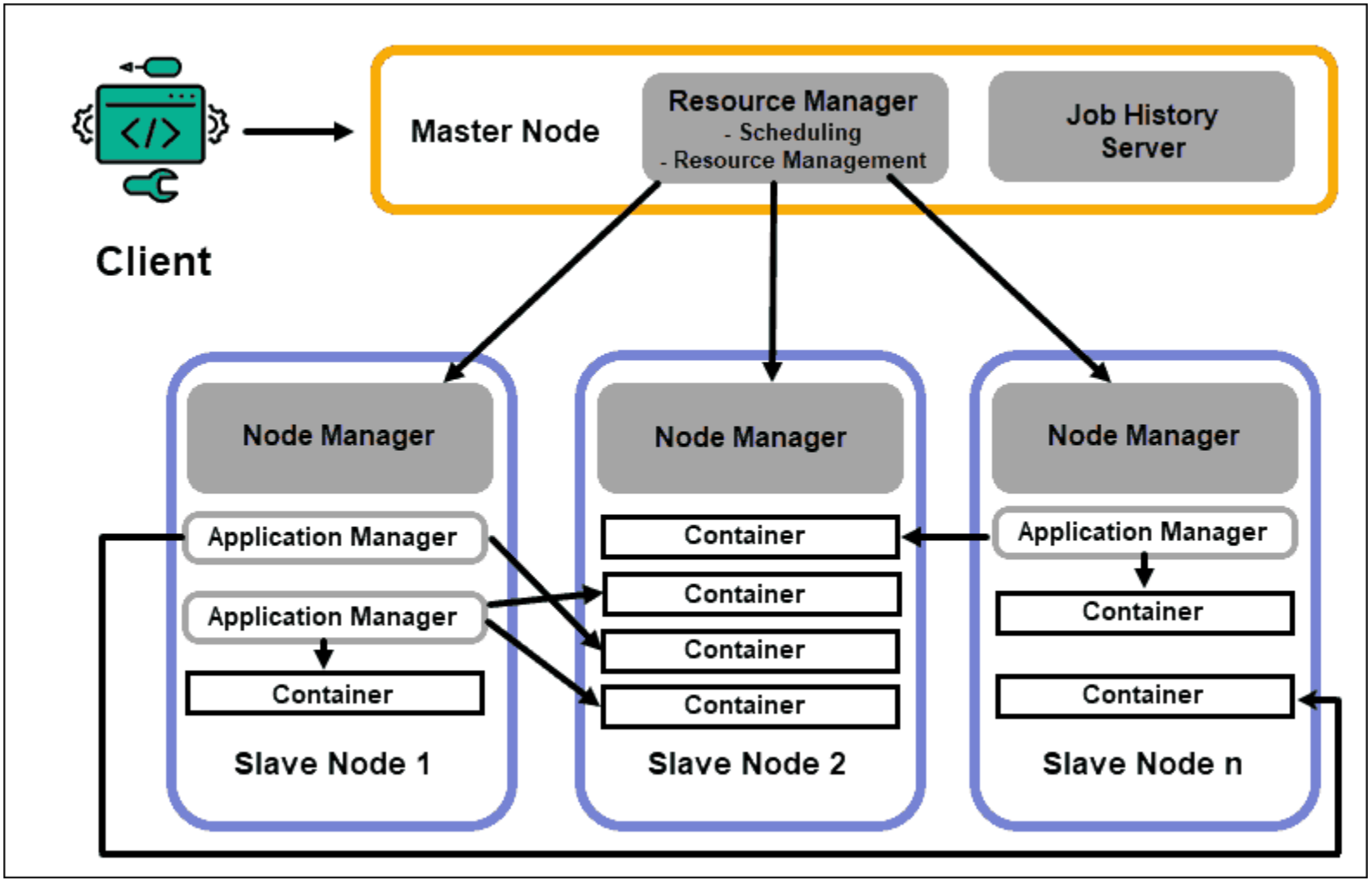
MapReduce

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

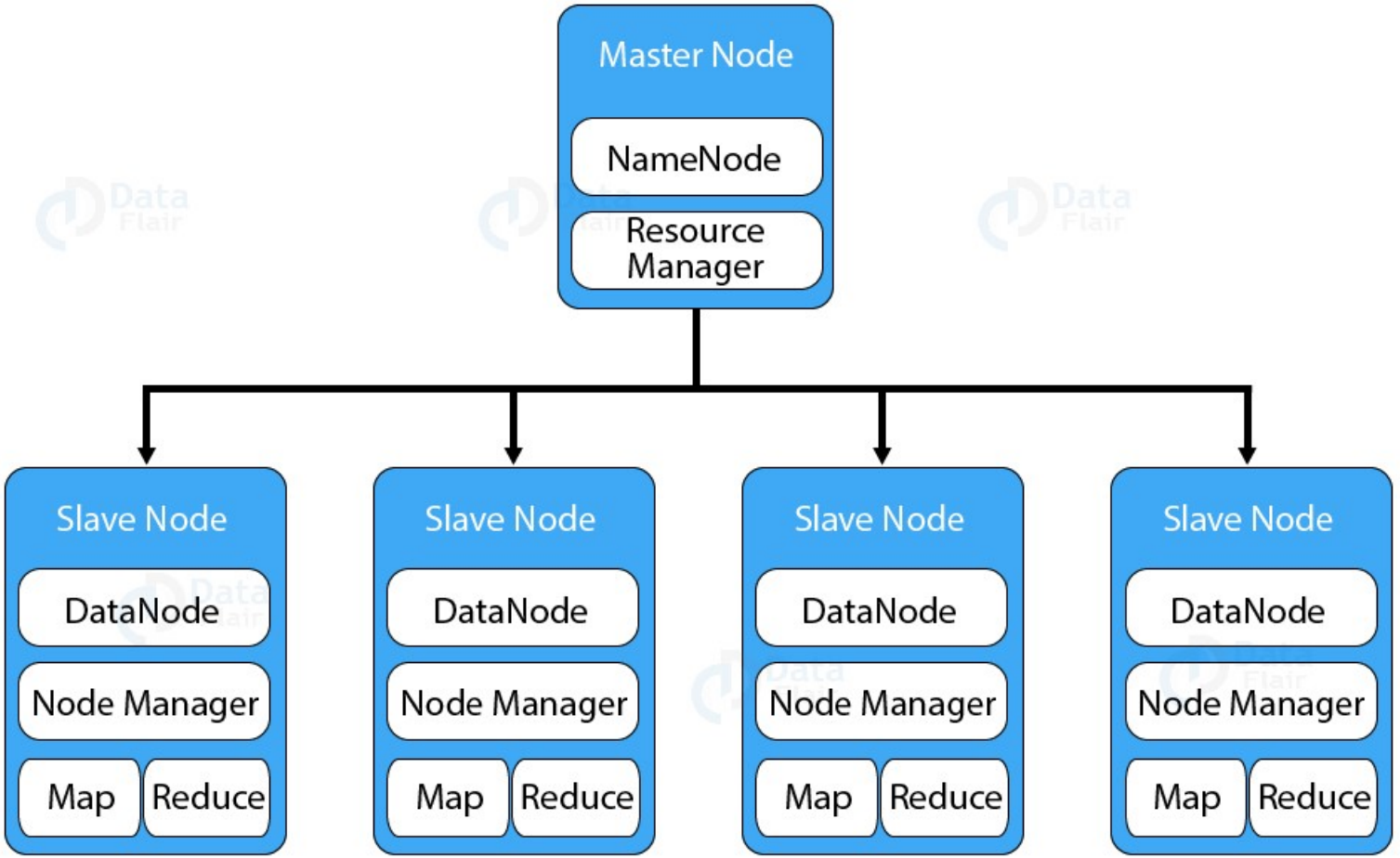
MapReduce: Word Count



Hadoop Master/slave Architecture



Hadoop Master/Slave Architecture



Hadoop Daemons

- Hadoop consist of five daemons.
  - **NameNode**
  - **DataNode**
  - **Secondary nameNode**
  - **Resource Manager**
  - **Node Manager**

- "Running Hadoop" means running a set of daemons, or resident programs, on the different servers in your network.
- These daemons have specific roles; some exist only on one server, some exist across multiple servers.

What is new in Hadoop 2.x Architecture?

- Hadoop 2.x has some common Hadoop API which can easily be integrated with any third party applications to work with Hadoop
- It has some new Java APIs and features in HDFS and MapReduce which are known as HDFS2 and MR2 respectively
- New architecture has added the architectural features like HDFS High Availability and HDFS Federation
- Hadoop 2.x not using Job Tracker and Task Tracker daemons for resource management now on-wards, it is using YARN (Yet Another Resource Negotiator) for Resource Management

HDFS High Availability (HA)

- **Problem:** As you know in Hadoop 1.x architecture Name Node was a single point of failure, which means if your Name Node daemon is down somehow, you don't have access to your Hadoop Cluster than after. How to deal with this problem?

HDFS High Availability (HA)

- **Solution:** Hadoop 2.x is featured with Name Node HA which is referred as HDFS High Availability (HA).
- Hadoop 2.x supports two Name Nodes at a time one node is active and another is standby node
- Active Name Node handles the client operations in the cluster
- StandBy Name Node manages metadata same as Secondary Name Node in Hadoop 1.x
- When Active Name Node is down, Standby Name Node takes over and will handle the client operations then after
- HDFS HA can be configured by two ways
  - Using Shared NFS Directory
  - Using Quorum Journal Manager

HDFS Federation

- **Problem:** HDFS uses namespaces for managing directories, file and block level information in cluster. Hadoop 1.x architecture was able to manage only single namespace in a whole cluster with the help of the Name Node (which is a single point of failure in Hadoop 1.x). Once that Name Node is down you loose access of full cluster data. It was not possible for partial data availability based on name space.

HDFS Federation

- **Solution:** Above problem is solved by HDFS Federation i Hadoop 2.x Architecture which allows to manage multiple namespaces by enabling multiple Name Nodes. So on HDFS shell you have multiple directories available but it may be possible that two different directories are managed by two active Name Nodes at a time.

HDFS 2.x Daemons

- **HDFS 2.x Daemons:** Name Node, Secondary Name Node (not required in HA) and Data Nodes
- **MapReduce 2.x Daemons (YARN):** Resource Manager, Node Manager

HDFS 2.x Daemons

- Hadoop 2.x allows Multiple Name Nodes for HDFS Federation
- New Architecture allows HDFS High Availability mode in which it can have Active and StandBy Name Nodes (No Need of Secondary Name Node in this case)
- Hadoop 2.x Non HA mode has same Name Node and Secondary Name Node working same as in Hadoop 1.x architecture

Name Node

- It is the master daemon that maintains and manages the DataNodes (slave nodes)
- It records the metadata of all the files stored in the cluster, e.g. The location of blocks stored, the size of the files, permissions, hierarchy, etc.
- There are two files associated with the metadata:
  - **FsImage:** It contains the complete state of the file system namespace since the start of the NameNode.
  - **EditLogs:** It contains all the recent modifications made to the file system with respect to the most recent FsImage.

Name Node

- It records each change that takes place to the file system metadata. For example, if a file is deleted in HDFS, the NameNode will immediately record this in the EditLog.
- It regularly receives a Heartbeat and a block report from all the DataNodes in the cluster to ensure that the DataNodes are live.
- It keeps a record of all the blocks in HDFS and in which nodes these blocks are located.
- The NameNode is also responsible to take care of the **replication factor** of all the blocks which we will discuss in detail later in this HDFS tutorial blog.
- In **case of the DataNode failure**, the NameNode chooses new DataNodes for new replicas, balance disk usage and manages the communication traffic to the DataNodes.

DataNode

- These are slave daemons or process which runs on each slave machine.
- The actual data is stored on DataNodes.
- The DataNodes perform the low-level read and write requests from the file system's clients.
- They send heartbeats to the NameNode periodically to report the overall health of HDFS, by default, this frequency is set to 3 seconds.

Secondary NameNode

- The Secondary NameNode is one which constantly reads all the file systems and metadata from the RAM of the NameNode and writes it into the hard disk or the file system.
- It is responsible for combining the EditLogs with FsImage from the NameNode.
- It downloads the EditLogs from the NameNode at regular intervals and applies to FsImage. The new FsImage is copied back to the NameNode, which is used whenever the NameNode is started the next time.

MapReduce 2.x Daemons (YARN)

- MapReduce2 has replace old daemon process Job Tracker and Task Tracker with YARN components Resource Manager and Node Manager respectively. These two components are responsible for executing distributed data computation jobs in Hadoop 2(Refer my post on YARN Architecture for further understanding).

Resource Manager

- This daemon process runs on master node (may run on the same machine as name node for smaller clusters)
- It is responsible for getting job submitted from client and schedule it on cluster, monitoring running jobs on cluster and allocating proper resources on the slave node
- It communicates with Node Manager daemon process on the slave node to track the resource utilization
- It uses two other processes named *Application Manager* and *Scheduler* for MapReduce task and resource management

Node Manager

- This daemon process runs on slave nodes (normally on HDFS Data node machines)
- It is responsible for coordinating with Resource Manager for task scheduling and tracking the resource utilization on the slave node
- It also reports the resource utilization back to the Resource Manager
- It uses other daemon process like Application Master and Container for MapReduce task scheduling and execution on the slave node

Hadoop Configuration Modes

- **Local (standalone) mode**
- By default, Hadoop is configured to run in a no distributed mode.
- It runs as a single Java process.
- Instead of HDFS, this mode utilizes the local file system.
- This mode useful for debugging and there isn't any need to configure core-site.xml, hdfs-site.xml, mapred-site.xml, masters & slaves.
- Stand-alone mode is usually the fastest mode in Hadoop.

Hadoop Configuration Modes

➢ **Pseudo-distributed mode**

- Hadoop can also run on a single node in a Pseudo Distributed mode.
- In this mode, each daemon runs on separate java process.
- In this mode custom configuration is required(core-site.xml, hdfs-site.xml, mapred-site.xml, yarn-site.xml).
- Here HDFS is utilized for input and output.
- This mode of deployment is useful for testing and debugging purposes.

Hadoop Configuration Modes

- **Fully distributed mode**
- This is the production mode of Hadoop.
- In this mode typically one machine in the cluster is designated as NameNode and another as Resource Manager exclusively. These are masters.
- All other nodes act as Data Node and Node Manager. These are the slaves.
- Configuration parameters and environment need to specified for Hadoop Daemons.
- This mode offers fully distributed computing capability, reliability, fault tolerance and scalability.

Other Hadoop-related projects at Apache

- Ambari
  - A web-based tool for provisioning, managing, and monitoring Apache Hadoop clusters
  - Includes support for Hadoop HDFS, Hadoop MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig and Sqoop.
  - Ambari also provides a dashboard for viewing cluster health such as heatmaps and ability to view MapReduce, Pig and Hive applications visually along with features to diagnose their performance characteristics in a user-friendly manner.

Other Hadoop-related projects at Apache

- Avro
  - A data serialization system.
- Cassandra
  - A scalable multi-master database with no single points of failure.
- Chukwa
  - A data collection system for managing large distributed systems.
- Hbase
  - A scalable, distributed database that supports structured data storage for large tables.

Other Hadoop-related projects at Apache

- Hive
  - A data warehouse infrastructure that provides data summarization and ad hoc querying.
- Mahout
  - A Scalable machine learning and data mining library.
- Pig
  - A high-level data-flow language and execution framework for parallel computation.

Other Hadoop-related projects at Apache

- Spark
  - A fast and general compute engine for Hadoop data.
  - Spark provides a simple and expressive programming model that supports a wide range of applications, including ETL, machine learning, stream processing, and graph computation.

Other Hadoop-related projects at Apache

- Spark
  - A fast and general compute engine for Hadoop data.
  - Spark provides a simple and expressive programming model that supports a wide range of applications, including ETL, machine learning, stream processing, and graph computation.

Other Hadoop-related projects at Apache

- Tez
  - A generalized data-flow programming framework, built on Hadoop YARN, which provides a powerful and flexible engine to execute an arbitrary DAG of tasks to process data for both batch and interactive use-cases.
  - Tez is being adopted by Hive™, Pig™ and other frameworks in the Hadoop ecosystem, and also by other commercial software (e.g. ETL tools), to replace Hadoop™ MapReduce as the underlying execution engine.
- Zookeeper
  - A high-performance coordination service for distributed applications.