Big Data Processing Framework - Hadoop In Data Science Projects

Manish Kalkar
Bellevue University
Bellevue, Nebraska 68005
mkalkar@my365.bellevue.edu

Rajasekhar Reddy Karna
Bellevue University
Bellevue, Nebraska 68005
rkarna@my365.bellevue.edu

Binay Jena
Bellevue University
Bellevue, Nebraska 68005
bpjena@my365.bellevue.edu

Juan Guevara
Bellevue University
Bellevue, Nebraska 68005
jguevara@my365.bellevue.edu

Abstract

Humanity's strive for better quality of life has led to evolution of data collection techniques from primitive tally marks to today's Big Data. Conventional data management methods have limitations with handling structured, unstructured, and semi-structured datasets which is inherent with trending SMAC, IoT and AI enabled devices. Variety of frameworks and technologies have evolved to capture, store, distribute and analyze large-sized datasets exhibiting complexity with structure and velocity aspects. Data Science as a multi-disciplinary area of expertise leverages architecture, algorithms and analytics at big data scale, diversity, and complexity to deliver the finesse required with value and insight derivation from the multitudes of information at disposal. Efficient, frugal processing of the diverse datasets at scale necessitates Hadoop like data processing ecosystem. Hadoop's data operations at scale, and open sourced collaborative platform has led to its popularity as processor for large datasets leveraging distributed computing efficiencies like MapReduce framework - where large data sets across high latency and availability clusters get distributed across multiple servers, process queries calculating partial results from each server, and then merge these partial results to generate aggregated response. This paper will focus on utilization of Hadoop as big data processing platform and how it is sacrosanct in delivering Data Science projects.

Author Keywords

Hadoop; Big Data; HDFS; MapReduce; YARN; Storage; Compute; Fault Tolerance

ACM Classification Keywords[26]

C.1.2 Multiple Data Stream Architectures; C.1.4 Distributed architectures; D.4.3 Distributed file systems; D.4.5 Fault-tolerance

1. Introduction

Moore's Law[21] holds good five decades since its postulation - moving on from microprocessors we're now in the era of nanotechnology, amongst smart devices, sensors, trackers, Artificial Intelligence (AI)/Internet of Things (IoT) devices. Coupled with the devices are Social, Mobile, Analytics and Cloud (SMAC) platforms that helped connecting, enhancing and capturing user experiences in variety of formats – text, videos, reviews, tweets, etc. Combined effect of this technology boom is spurt in data volumes at quintillion scale[14] daily. It's probably cliché now if we refer to indispensability of "data" driven decision making, not only for business and technology contexts, but also for governance and public policy planning, disease control, space research (even planning with personal expense, commute, vacation, maybe even choice with life-partner soon enough if human attributes could be quantified !!) – scope of data driven decision making has become as imaginative as it can get. Big Data analytics has positioned itself to serve insights and help decision making by processing data at any scale. When we say data at scale, we're not only talking about the volumes, we're dealing with velocity and variety formats too – videos, images, texts, etc.; a recent data snippet being - "Within a minute, 72 hours of videos are uploaded to YouTube, 100,000 tweets shared across Twitter, 200,000 images posted on Facebook"

From governments to business entities to individuals, reliance on data insights poses questions regarding the approaches, and 'how's' of processing and storage of petabytes scale data with real-time latency. The question stemmed from realization that with unprecedented spurt in data volume and velocity, traditional relational databases and business intelligence solutions are inadequate to meet the efficiency requirements of the insights. Further there are challenges with scalability, cost overruns, technology propriety and risk of breaching capacity thresholds. Our Research Questions are:

1. Storing data at scale: How do we make data storage infinitely scalable, while ensuring seamless retention, archival practices without affecting query retrieval and latency aspects?
2. Computing at scale: We are expecting computation scaling challenges with Petabyte scale data processing, the platform's computation capacities should be frugally responsive and scale along with data volume growth and corresponding processing needs. How to ensure platform's compute abilities scale while ensuring optimal data processing performance?
3. Resilience at scale: Given the scale of data processing complexities, the platform or framework is assumed to have task failures, so how does resource mapping, node allocation and execution tracking work in case of failovers in the framework?

In 2002, Apache Nutch[3] project intended to create a search engine to index 1 billion web pages, which confirmed the real challenges at big data scale with scaling, implementation, and maintenance costs. Thought process with the challenge resolution leverages distributed computing architectures. The Google File System[15] (2003) laid out the architecture details to deal with storage aspect of the problem, and MapReduce[10] (2004) paper did address the processing or computation aspect. The challenge lied with turning these whitepapers into an implementation framework – Hadoop framework. Hadoop framework (open sourced Java based application) works in an environment that provides distributed storage and computation across clusters of computers, running on cheap/commodity grade hardware. Two major layers in Hadoop are (1) Processing/Computation layer: MapReduce is a parallel programming model for coding distributed applications for efficient processing of multi-terabyte datasets on large clusters of commodity hardware in a reliable and fault-tolerant manner. (2) Storage layer: provides a distributed file system designed to run on commodity hardware. In addition, Hadoop framework includes two modules, Hadoop Common (Java libraries and utilities) and Hadoop YARN (Yet Another Resource Negotiator) (a framework for job scheduling and cluster resource management). Hadoop architecture becomes an automatic choice for data storage and processing platform due to its inherent fault/ failover tolerance, inherent scalability gained through distributed computing/parallel processing, open-source collaborative development and interoperability/compatibility with traditional legacy systems, thus helps derisk system level bottlenecks, code/technology interface dependencies.

Going beyond its original goal of searching and indexing billions of web pages, many organizations have implemented Hadoop as their next big data platform due to its compute and storage prowess. Hadoop has become founding technology for Big data processing, Analytics, and Data Science!

In this paper, we dive deeper into the most widely used big data framework - Hadoop, designed to cope with Big Data challenges and its computing, storage and retrieval complexities.

2. Big Data Characteristics and Hadoop – Overview[9]

This section focuses on Big Data and its characteristics and how Apache Hadoop help to solve Big Data problems and then we will talk about the Apache Hadoop framework and how it works.

2.1 Big Data

Big data is a term applied to data sets whose size or type is beyond the ability of traditional relational databases to capture, manage and process the data with low latency.

Big data is also a term that describes the large volume of data – both structured and unstructured – that inundates a business on a day-to-day basis. But it is not the amount of data that is important. It's what organizations do with the data that matters. Big data can be analyzed for insights that lead to better decisions and strategic business moves.

2.2 Big Data Analytics

Big data analytics is the use of advanced analytic techniques against very large, diverse data sets that include structured, semi-structured and unstructured data, from different sources, and in different sizes from terabytes to zettabytes.

2.3 Characteristics of Big Data

We differentiate Big Data characteristics from traditional data by one or more of the Volume, Velocity, Variety, and variability.

These points are called 4 Vs in the Big Data industry.

2.3.1 Volume

Volume is absolutely a slice of the bigger pie of Big data. The internet mobile cycle, delivering with it a torrent of social media updates, sensor data from tools and an outburst of e-commerce, means that all industry swamped with data, which can be amazingly valuable if users understand how to work on it.

2.3.2 Variety

Structured data collected in SQL tables are a matter of past. Today, 90% of data produced is 'unstructured,' arriving in all shapes and forms- from Geospatial data to tweets which can investigate for content and thought, to visual data such as photos and videos.

So, is this always comes in structured still or is it come in unstructured or semi-structured?

2.3.3 Velocity

Each minute of every day, users throughout the globe upload 200 hours of video on YouTube, send 300,000 tweets and carry over 200 million emails. And this keeps growing as the internet speed is getting faster.

2.3.4 Veracity

This applies to the uncertainty of the data available to marketers. This may also be referred to as the variability of data streaming that can be changeable, making it tough for organizations to respond quickly and more appropriately.

2.4 What is Hadoop?

Hadoop is an open-source, a Java-based programming framework that continues the processing of large data sets in a distributed computing environment.

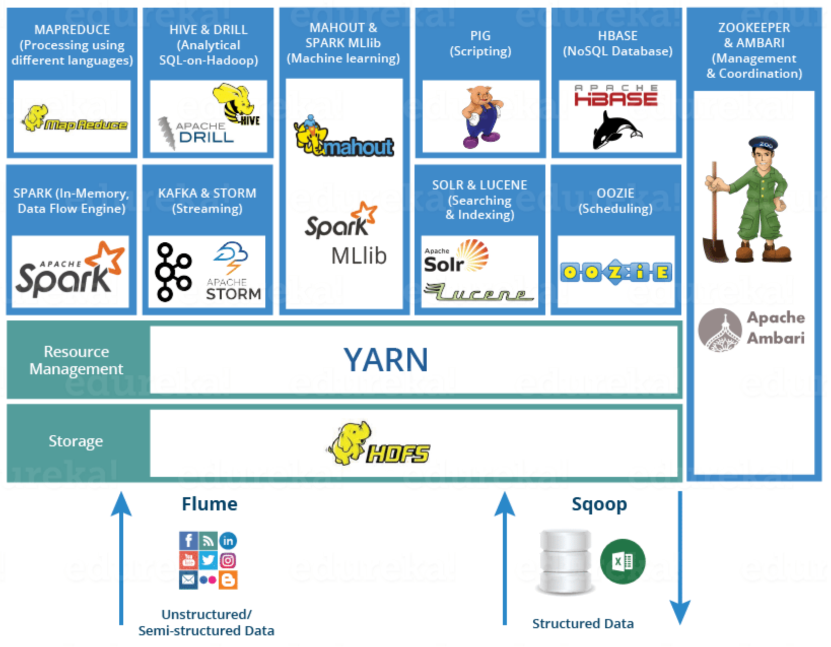
Hadoop library allows the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly available service on top of a cluster of computers.

2.5 Why Hadoop?

Hadoop runs few applications on distributed systems with thousands of nodes involving petabytes of information. It has a distributed file system, called Hadoop Distributed File System or HDFS, which enables fast data transfer among the nodes.

2.6 Hadoop Framework

The Hadoop framework application works in an environment that provides distributed storage and computation across clusters of computers.


MAPREDUCE (Processing using different languages)
HIVE & DRILL (Analytical SQL-on-Hadoop)
MAHOUT & SPARK MLlib (Machine learning)
PIG (Scripting)
HBASE (NoSQL Database)
ZOOKEEPER & AMBARI (Management & Coordination)
SPARK (In-Memory, Data Flow Engine)
KAFKA & STORM (Streaming)
SOLR & LUCENE (Searching & Indexing)
OOZIE (Scheduling)
Resource Management
YARN
Storage
HDFS
Apache Ambari
Flume
Unstructured/ Semi-structured Data
Sqoop
Structured Data

2.6.1 Hadoop Distributed File System (HDFS):

It provides a storage layer for Hadoop. It is suitable for distributed storage and processing, i.e. while the data is being stored it first get distributed & then it proceeds. HDFS Provides a command line interface to interact with Hadoop. It provides streaming access to file system data. So, it includes file permission and authentication. So, what store data here it is HBase who store data in HDFS.

2.6.2 HBase:

It helps to store data in HDFS. It is a NoSQL database or non-relational database. HBase mainly used when users need random, real-time, read/write access to your big data. It gives support to the high volume of data and high throughput. In HBase, a table can have thousands of columns.

2.6.3 Sqoop:

A Sqoop is a tool designed to transfer data between Hadoop and NoSQL. It is managed to import data from relational databases such as Oracle and MySQL to HDFS and export data from HDFS to relational database. If users want to ingest data such as streaming data, sensor data or log files, then you can use Flume.

2.6.4 Flume:

Flume distributed service for ingesting streaming data. So, Distributed data that collect event data & transfer it to HDFS. It is ideally suitable for event data from multiple systems. After the data transferred in the HDFS, it processed and one the framework that process data it is SPARK.

2.6.5 SPARK:

An open source cluster computing framework. It provides 100 times faster performance as compared to MapReduce.

For few applications with in-memory primitives as compared to the two-state disk-based MapReduce. Spark run in the Hadoop cluster & process data in HDFS. It also supports a wide variety of workload.

2.6.6 Hadoop MapReduce:

It is another framework that processes the data. The original Hadoop processing engine which primarily based on JAVA. Based on the Map and Reduce programming model. Many tools such as Hive, Pig build on Map Reduce Model. It is broad & mature fault tolerance framework. It is the most commonly used framework.

After the data processing, it is an analysis done by the open-source data flow system called Pig.

2.6.7 Pig:

It is an open-source dataflow system. It mainly used for Analytics. It covert pig script to Map-Reduce code and saving producer from writing Map-Reduce code. At Ad-Hoc queries like filter & join which is challenging to perform in Map-Reduce can be done efficiently using Pig. It is an alternate to writing Map-Reduce code.

Impala is also used to analyze the data.

2.6.8 Impala:

It is a high-performance SQL engine which runs on a Hadoop cluster. It is ideal for interactive analysis. It has a very low latency which can be measured in milliseconds.

It supports a dialect of SQL (Impala SQL). Impala supports a dialect of a sequel. So, data in HDFS modelled as a database table. Hive is used to implement Data Analysis.

2.6.9 Hive:

It is an abstraction cover on top of the Hadoop. It's very similar to the Impala. However, it preferred for data processing and ETL (extract, transform and load) operations. Impala preferred for ad-hoc queries, and hive executes queries using Map-Reduce. However, a user no needs to write any code in low-level Map-Reduce. Hive is suitable for structured data. After the data examined it is ready for the user to access what supports the search of data, it can be done using clutter is Search Cloudera.

2.6.10 Cloudera Search:

It is near real-time access products it enables non-technical users to search and explore data stored in or ingest it into Hadoop and HBase. In Cloudera, users don't need SQL or programming skill to use Cloudera search because it provides a simple full-text interface for searching. It is a wholly blended data processing platform.

Cloudera search does the flexible, scalable, and robust storage system combined with CD8 or Cloudera distribution including Hadoop. This excludes the need to move large data sets across infrastructure to address the business task. A Hadoop job such as MapReduce, Pig, Hive and Sqoop have workflows.

2.6.11 Oozie:

Oozie is a workflow or coordination method that users can employ to manage Hadoop jobs. Multiple actions occurred within the start and end of the workflow.

2.6.12 Hue:

Hue is an acronym for Hadoop user experience. It is an open-source web interface for analyzing data with Hadoop. Users can execute the following operations using Hue.

1. Upload and browse data
2. Query a table in Hive and Impala
3. Run Spark and Pig jobs
4. Workflow search data.

Hue makes Hadoop accessible to use. It also provides an editor for the hive, impala, MySQL, Oracle, Postgre SQL, Spark SQL and Solar SQL.

2.6.13 Hadoop YARN Architecture:

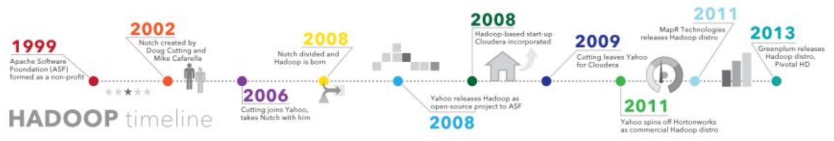
YARN stands for "Yet Another Resource Negotiator ". It was introduced in Hadoop 2.0 to remove the bottleneck on Job Tracker which was present in Hadoop 1.0.

The scheduler in Resource manager of YARN architecture allows Hadoop to extend and manage thousands of nodes and clusters.

3. Hadoop History and Timeline - Key Milestones[23]

As the World Wide Web grew in the late 1900s and early 2000s, search engines and indexes were created to help locate relevant information amid the text-based content. In the early years, search results were returned by humans. But as the web grew from dozens to millions of pages, automation was needed. Web crawlers were created, many as university-led research projects, and search engine start-ups took off (Yahoo, AltaVista, etc.).

One such project was an open-source web search engine called Nutch – the brainchild of Doug Cutting and Mike Cafarella. They wanted to return web search results faster by distributing data and calculations across different computers so multiple tasks could be accomplished simultaneously. During this time, another search engine project called Google was in progress. It was based on the same concept – storing and processing data in a distributed, automated way so that relevant web search results could be returned faster.


HADOOP timeline
1999
Apache Software Foundation (ASF) formed as a non-profit
2002
Nutch created by Doug Cutting and Mike Cafarella
2006
Cutting joins Yahoo, takes Nutch with him
2008
Nutch divided and Hadoop is born
2008
Yahoo releases Hadoop as open-source project to ASF
2008
Hadoop-based start-up Cloudera incorporated
2009
Cutting leaves Yahoo for Cloudera
2011
Yahoo spins off Hortonworks as commercial Hadoop distro
2011
MapR Technologies releases Hadoop distro
2013
Greenplum releases Hadoop distro, Pivotal HD

In 2006, Cutting joined Yahoo and took with him the Nutch project as well as ideas based on Google's early work with automating distributed data storage and processing. The Nutch project was divided – the web crawler portion remained as Nutch and the distributed computing and processing portion became Hadoop (named after Cutting's son's toy elephant). In 2008, Yahoo released Hadoop as an open-source project. Today, Hadoop's framework and ecosystem of technologies are managed and maintained by the non-profit Apache Software Foundation (ASF), a global community of software developers and contributors.

Use of the framework grew over the next few years, and three independent Hadoop vendors were founded: Cloudera in 2008, MapR Technologies a year later and Hortonworks as a Yahoo spinoff in 2011. In addition, AWS launched a Hadoop cloud service called Elastic MapReduce in 2009. That was all before Apache released Hadoop 1.0.0, which became available in December 2011 after a succession of 0.x releases.

4. Key Features and Advantages of Hadoop

4.1 Hadoop - Key Features[22]

4.1.1 Computing power

Hadoop's disseminated computing model enables it to process enormous quantities of data. The more nodes get utilized, more computing potential users have.

4.1.2 Versatility

Hadoop stores data without demanding any preprocessing. Store data—structured as well as unstructured data, for example, text, pictures, and even video now; and determine how to deal with it later. Hadoop is a perfect stage for executing the pre-processing productively and in a distributed way across huge datasets, utilizing tools like PIG, or HIVE, and scripting languages like Python or map-reduce.

4.1.3 Adaptation to internal failure

Hadoop naturally stores numerous duplicates of all data, and if one node fails while processing of data, tasks are diverted to different nodes and distributed computing proceeds.

4.1.4 Speed

Each company utilizes the platform to complete the job at a quicker rate. Hadoop empowers the organization to do only that with its data storing needs. It utilizes a storage framework wherein the information is put away on a distributed file framework. As the devices utilized for the data processing are situated on related servers as the data, the processing activity is likewise completed at a quicker rate. Thus, within few minutes systems can process terabytes of data with the help of Hadoop.

4.1.5 Minimal Cost

The open-source Hadoop system is free, and the data is saved on hardware.

4.1.6 Adjustability

Hadoop framework can be built without much of a stress, essentially by including more nodes. The Hadoop process brings down the danger of disastrous framework failure and unforeseen data loss, irrespective of if a noteworthy number of nodes end up defective. Thus, Hadoop immediately rose as an establishment for processing of huge data jobs, for instance, scientific analytics, processing large volumes of sensor data, and business and sales outlining.

4.2 Hadoop – Advantages[20]

Hadoop has a lot to offer and known for the following four benefits:

4.2.1 Flexibility

Hadoop stores data without requiring any preprocessing. Store data—even unstructured data such as text, images, and video—now; decide what to do with it later.

4.2.2 Fault Tolerance

Hadoop automatically stores multiple copies of all data, and if one node fails during data processing, jobs are redirected to other nodes and distributed computing continues.

4.2.3 Low Cost

The open-source framework is free, and data is stored on commodity hardware.

4.2.4 Scalability

Hadoop system can grow, simply by adding more nodes.

Although the development of Hadoop was motivated by the need to search millions of webpages and return relevant results, it today serves a variety of purposes. Hadoop's low-cost storage makes it an appealing option for storing information that is not currently critical but that might be analyzed later.

Hadoop storage is unencumbered by the schema-related constraints commonly found in SQL-based systems.

Organizations are using Hadoop to stage large amounts of raw, sometimes unstructured data for loading into enterprise data warehouses. Many of Hadoop's largest adopters use it for the real-time data analysis that enables web-based recommendation systems.

5. Hadoop Architecture

5.1 How Hadoop Works[17]

A client accesses unstructured and semi-structured data from sources including log files, social media feeds and internal data stores. It breaks the data up into "parts," which are then loaded into a file system made up of multiple nodes running on commodity hardware. The default file store in Hadoop is the Hadoop Distributed File System, or HDFS. File systems such as HDFS are adept at storing large volumes of unstructured and semi-structured data, as they do not require data to be organized into relational rows and columns.

Each "part" is replicated multiple times and loaded into the file system so that if a node fails, another node has a copy of the data contained on the failed node. A Name Node acts as facilitator, communicating information such as which nodes are available, where in the cluster certain data resides, and which nodes have failed back to the client.

Once the data is loaded into the cluster, it is ready to be analyzed via the MapReduce framework. The client submits a "Map" job - usually a query written in Java – to one of the nodes in the cluster known as the Job Tracker. The Job Tracker refers to the Name Node to determine which data it needs to access to complete the job and where in the cluster that data is located. Once determined, the Job Tracker submits the query to the relevant nodes. Rather than bringing all the data back into a central location, processing then occurs at each node simultaneously, in parallel.

When each node has finished processing its given job, it stores the results. The client initiates a "Reduce" job through the Job Tracker in which results of the map phase stored locally on individual nodes are aggregated to determine the "answer" to the original query, then loaded on to another node in the cluster. The client accesses these results, which can then be loaded into one of several analytic environments for analysis. The MapReduce job has now been completed.

Once the MapReduce phase is complete, the processed data is ready for further analysis by data scientists and others with advanced data analytics skills. Data scientists can manipulate and analyze the data using any of several tools for any number of uses, including to search for hidden insights and patterns or to create the foundation to build user-facing analytic applications. The data can also be modeled and transferred from Hadoop clusters into existing relational databases, data warehouses and other IT systems for further analysis and/or to support transactional processing.

5.2 Hadoop Architecture Components[4]

5.2.1 Hadoop MapReduce – an implementation of the MapReduce programming model

The system overview of MapReduce is illustrated in Figure 1 below.

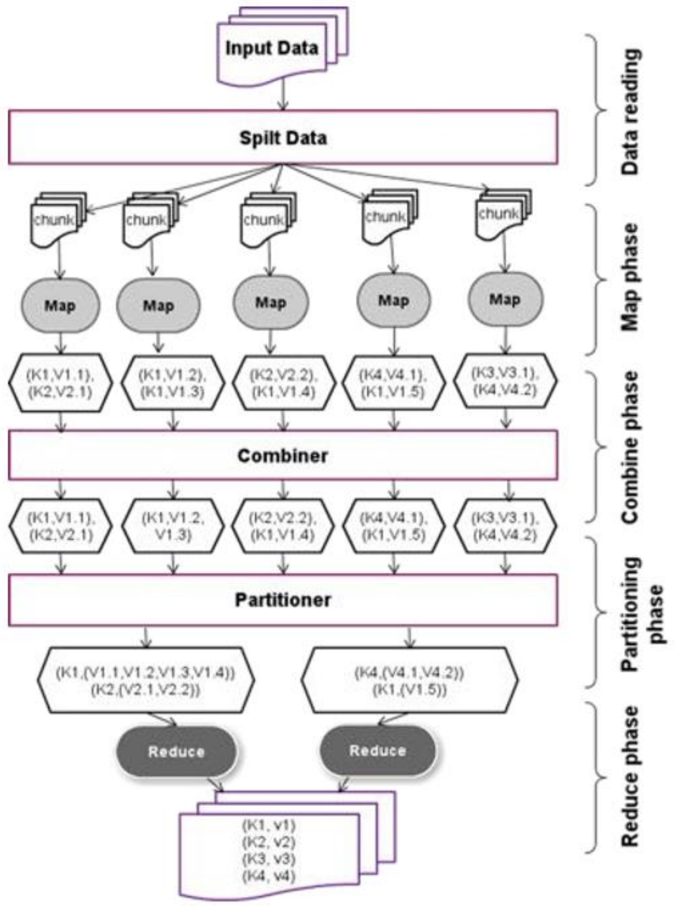


Figure 1: The MapReduce architecture

MapReduce is a programming model that was designed to deal with parallel processing of large datasets. MapReduce has been proposed by Google in 2004 as an abstraction that allows to perform simple computations while hiding the details of parallelization, distributed storage, load balancing and enabling fault tolerance.

The central features of the MapReduce programming model are two functions: Map and Reduce. The Map function takes a single key value pair as input and produces a list of intermediate key value pairs. The intermediate values associated with the same intermediate key are grouped together and passed to the Reduce function. The Reduce function takes as input an intermediate key and a set of values for that key. It merges these values together to form a smaller set of values.

As shown in Figure 1, the basic steps of a MapReduce program are as follows:

1. **Data reading:**
   In this phase, the input data is transformed to a set of key value pairs. The input data may come from various sources such as file systems, database management systems or main memory (RAM). The input data is split into several fixed size chunks. Each chunk is processed by one instance of the Map function.

2. **Map phase:**
   For each chunk having the key value structure, the corresponding Map function is triggered and produces a set of intermediate key value pairs.

3. **Combine phase:**
   This step aims to group together all intermediate key-value pairs associated with the same intermediate key.

4. **Partitioning phase:**
   Following their combination, the results are distributed across the different Reduce functions.

5. **Reduce phase:**
   The Reduce function merges key value pairs having the same key and computes a final result.

5.2.2 Hadoop Distributed File System (HDFS)

HDFS is an open source implementation of the distributed Google File System (GFS). It provides a scalable distributed file system for storing large files over distributed machines in a reliable and efficient way. In Figure 2 below, we show the abstract architecture of HDFS and its components.

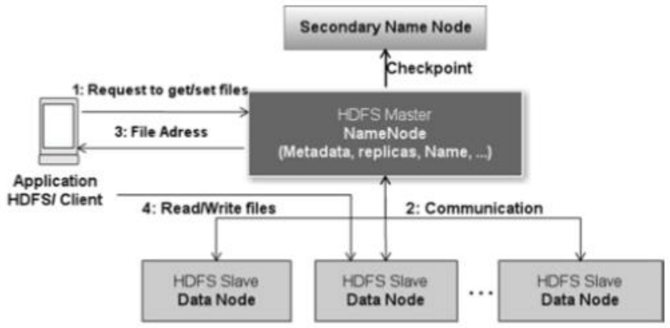


Figure 2: HDFS architecture

HDFS consists of a master/slave architecture with a Name Node being master and several Data Nodes as slaves.

The Name Node is responsible for allocating physical space to store large files sent by the HDFS client. If the client wants to retrieve data from HDFS, it sends a request to the Name Node.

The Name Node will seek their location in its indexing system and subsequently sends their address back to the client. The Name Node returns to the HDFS client the meta data (filename, file location, etc.) related to the stored files.

A secondary Name Node periodically saves the state of the Name Node. If the Name Node fails, the secondary Name Node takes over automatically.

5.2.3 YARN Architecture

In the second version of Hadoop called YARN, the two major features of the Job Tracker have been split into separate daemons:

1. A global Resource Manager
2. Per-application Application Master

In Figure 3 below, we illustrate the overall architecture of YARN.

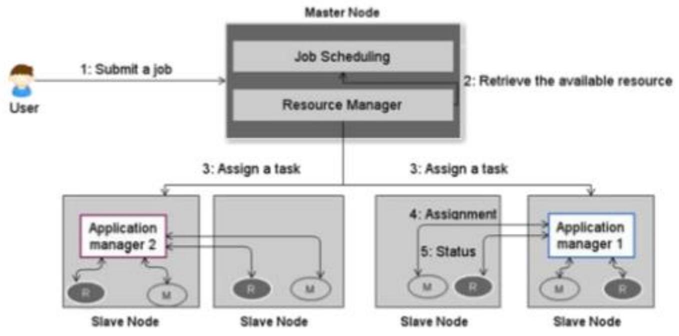


Figure 3: YARN architecture

The Resource Manager receives and runs MapReduce jobs. The per-application Application Master obtains resources from the Resource Manager and works with the Node Manager(s) to execute and monitor the tasks. In YARN, the Resource Manager (respectively the Node Manager) replaces the Job Tracker (respectively the Task Tracker).

Taking as examples Mesos and Zookeeper. Mesos is an open source cluster manager that ensures a dynamic resource sharing and provides efficient resources management for distributed frameworks. It is based on a master/slave architecture. The master node relies on a daemon, called master process. This later manages all executor daemons deployed in the slave nodes, on which user tasks are distributed and executed.

Apache ZooKeeper is an open source and fault tolerant coordinator for large distributed systems. It provides a centralized service for maintaining the cluster's configuration and management. It also ensures the data or service synchronization in distributed applications. Unlike YARN or Mesos, Zookeeper is based on a cooperative control architecture, where the same service is deployed in all machines of the cluster. Each client or application can request the Zookeeper service by connecting to any machine in the cluster.

6. Hadoop Case Study

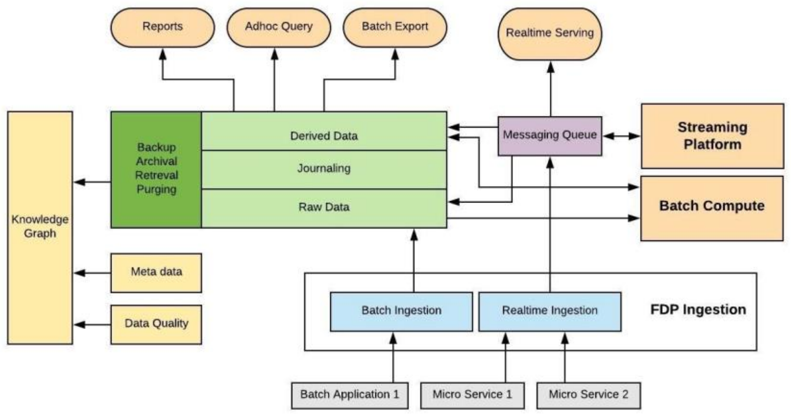
Flipkart Data Platform
India's biggest eCommerce Big Data Platform

6.1 Business Context

Amazon's growth has been a near-juggernaut affair, dominating competition through its competitive pricing, aggressive vendor negotiations claiming to be world's most customer centric company. There have been umpteen case studies on Amazon's supply chain optimization techniques and retail business dominance across countries where Amazon has expanded into. It was not a red-carpet affair for Amazon in India though. With a billion plus consumers (which is about 20% of entire world population), tapping into this consumer base has been every eCommerce player's dream. The competitive landscape is intense due to traditional grocery stores dominating the retail scene, while deep pocket investors as Amazon, Alibaba, Softbank and another retail behemoth Walmart, woo consumers to their eCommerce platforms. The competitive landscape is important because any slip in decision making gives an edge to the competitors in gaining consumer / market share. Hence speed and accuracy are implicit requirements of the insight's engines. Flipkart did prove to be a worthy challenger in Indian eCommerce space, being backed by Walmart as investor – we dig deeper into Flipkart's Data Platform components.

6.2 Flipkart Data Platform powered by Hadoop Ecosystem

Flipkart's exponential growth has been powered by its tech adoption and embracing of Service Oriented Architectures[11] (SOA), wherein each function results in a micro-service that drives user experiences. Some micro-service examples are catalog search, pricing engines, delivery estimations, product listings, etc. Each micro-service logs its own transactional database, which are spread across variety of platforms MySQL, HBase, ElasticSearch, Redis, etc. For the micro-services, there is an interdependency amongst them to run their businesses, while all of them together are required for Data Science, Insights and Analytics. Flipkart Data Platform (FDP), powered on Hadoop ecosystem, caters to this use case of addressing micro-services' dependencies as well as consolidating an aggregated view of entire business, thus equips the respective teams with capabilities to consume and act on the datasets.



FDP runs on 800-1000 node Hadoop cluster, storing over 40 PB data. Nearly 25000 compute data pipelines run on YARN clusters, with daily data ingestion being in TB scale, which does handle data spikes on occasions due to sales and promo events. FDP components are built on tech stack as Hadoop Distributed File System (HDFS), Hive, YARN, MapReduce, Spark, Storm and other API services supporting data meta layer.

FDP comprises of the following high-level components[8]:

6.2.1 FDP Ingestion System

Every team or micro-service ingests FDP data. Each ingested data chunk has a specified schema which can be created through self-serve user interface (UI). The tech stack for the system includes Messaging Queue (Kafka), Dropwizard, HDFS, Quartz, Azkaban & Hive. Ingestion happens through:

1. Specter – a Java library which sends the payload to FDP Kafka streams
2. Dart Service - This is a REST service which allows the payload to be sent over HTTP
3. File Ingestor – a CLI tool to dump data directly into FDP's HDFS

User creates a schema for which the corresponding Kafka topic is created. Using Specter or Dart client, user can start ingesting data into FDP. Ingestion system stores the data chunks or payloads in HDFS as raw Hive tables, on top of which journaling is run to make the data query able.

6.2.2 Batch Compute or Batch Data Processing System

Dimensional modeling concepts (primarily Star schema) is implemented here through Hive or Vertica tables. Facts/dimensions are created by defining a schema and Hive jobs to populate the schema.

6.2.3 Realtime Processing System / Flipkart Streaming Platform

Streaming Platform and Real-time Serving components are part of FSP which uses Apache Storm and Flink stack. Realtime ingestions via Specter or Dart can be directly consumed by Kafka topics. FSP enables plugging Spark jobs to these Kafka topics for analytical processing like rolling window aggregations at minute, hour, day grains or historic date ranges for any metrics.

6.2.4 Messaging Queue

This acts like a buffer or temporary storage system or an event bus when the destination system is busy or offline. Messages can be a plain text message, or prompt command for message queue to execute certain task, or byte array with headers indicating certain event. Producer generates, and Consumer captures the message from the queue depending on their functional micro-services. FDP uses Apache Kafka as stream processing software system for its message transaction logs.

6.2.5 Data Lake

Core storage platform of the FDP architecture leverages HDFS to store raw data, journaled data and derived data. Raw data is used by data science for developing data-oriented products (e.g. – pricing engines). Data is typically present in the form of batches or real-time streams. From the data lake, data is transferred through three routes primarily –

1. Reports – Canned reports typically produced off batch data, enable insight on website transactions, logs and readings.
2. Adhoc Query - Data Analysts are the typical users of these reports delivered through Business Intelligence (BI) tools.
3. Batch Exports – Bulk fetch and exports are sent out to further processing services (e.g. – seller category reconciliation)

6.2.6 Knowledge Graphs

Data lineage and metadata drives knowledge graph component of FDP. Knowledge graph leverages machine learning tools and libraries to explain and understand underlying semantics for newer fact derivation. Apache Spark's GraphX library is used as the tool for this component.

6.3 Challenges

FDP leverages almost the entirety of Hadoop ecosystem players. FDP challenges are owed to data veracity of its sources leading to data quality issues, 25000 compute pipelines can be optimized if data modeling practices were adopted org-wide leading to lowering of compute resources and costs. Realtime report generation as against N seconds latency is a technical debt with the Hadoop implementation, which can be addressed by websocket prioritization.

7. Conclusion

With the advent of big data, it would have been quite a daunting and costly affair for businesses, governments, all insight dependent entities had the Hadoop framework not come to the rescue. In a traditional environment, with the spurt in data volumes the choice would have been to purge older raw data given the fact that cost of storage scaling would probably never justify the frequency and value of insights derived; and simply forget processing the data volume, waiting for the processed results would have been a test of patience !

With Hadoop framework, scaling out and scaling up is a built-in feature of HDFS, and it leverages parallel processing and distributed computing architecture – so the hardware can be commodity grade, thus storage scaling is a matter of beefing nodes to a cluster or adding more clusters. Hadoop basically enabled infinite storage scaling. Similarly, for computing/processing, MapReduce processes Terabytes in seconds, and Petabytes in hours, which would be 'in-my-dreams' scale with traditional DW/ETL products. Performance and scaling are the benefits of Hadoop which are more relevant in big data, however, the most striking aspect of Hadoop framework is failure resilience or fault tolerance during task execution. When data is sent to an individual node, its simultaneously replicated to all other nodes as well – there always is another copy available for use in case the initial individual node fails to act on the data chunk. It is assumed that tasks will have multiple failures, which is where YARN component and MapR distribution enables No NameNode architecture to provide true high availability.

Big data was a 'natural' evolution, but Data Science would have been at the mercy of component sellers/providers, had Hadoop framework not evolved to what it is today. Open source collaborative development could ensure this reality. With Hadoop in the armory, Data Science experiments can focus on the problem definition, model development, training and deployment that worrying over data handling and processing constraints. It wouldn't be an understatement to state Hadoop as the most prized weapon in a data scientist's artillery.

8. References

1. Kuo, A., Chrimes, D., Qin, P., & Zamani, H. 2019. MapReduce Based Platform for Supporting Big Data Analytics Retrieved from https://www.ncbi.nlm.nih.gov/pubmed/30741201
2. By Anurag. 2018. How Hadoop Changed Data Science? Retrieved from https://www.newgenapps.com/blog/what-is-hadoop-and-how-it-changed-data-science/
3. Apache Nutch. Retrieved from https://en.wikipedia.org/wiki/Apache_Nutch
4. Sabeur Aridhi, Wissem Inoubli, Haithem Mezni. 2018. An experimental survey on big data frameworks. Retrieved from https://www.researchgate.net/publication/324682390_An_experimental_survey_on_big_data_frameworks
5. Sikha Bagui, Probal Chandra Dhar. 2019. Association rule mining in Hadoops's MapReduce environment. Retrieved from https://doi-org.ezproxy.bellevue.edu/10.1186/s40537-019-0238-8
6. Bilal Akil ; Ying Zhou ; Uwe Röhm. 2017. Usability of Hadoop MapReduce, Apache Spark & Apache flink. Retrieved from https://ieeexplore.ieee.org/document/8257938
7. Doug Cutting. 2006. Hadoop history or Evolution. Retrieved from https://www.geeksforgeeks.org/hadoop-history-or-evolution/
8. DataFlair Team. 2019. How Big Data is helping Flipkart to achieve the Milestone? Retrieved from https://data-flair.training/blogs/big-data-at-flipkart/
9. Sahil Dhankhad. 2019. Brief Summary of Apache Hadoop: A Solution of Big Data Problem. Retrieved from https://towardsdatascience.com/a-brief-summary-of-apache-hadoop-a-solution-of-big-data-problem-and-hint-comes-from-google-95fd63b83623
10. Jeffrey Dean, Sanjay Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. Retrieved from https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf
11. Rishabh Dua. 2018. Flipkart Data Platform — India's largest eCommerce Big Data Platform. Retrieved from https://tech.flipkart.com/overview-of-flipkart-data-platform-20c6d3e9a196
12. edureka Blog. 2019. Why do we need Hadoop for Data Science? Retrieved from https://www.edureka.co/blog/hadoop-for-data-science/
13. Tom Fawcett and Foster Provost. Data Science and its Relationship to Big Data Retrieved from https://www.liebertpub.com/doi/full/10.1089/big.2013.1508
14. Forbes' research. 2018. 2.5 quintillion bytes of data. Retrieved https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/#5375db7260ba
15. Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. 2003. The Google File System. Retrieved from https://static.googleusercontent.com/media/research.google.com/en//archive/gfs-sosp2003.pdf
16. Jaiswal, Er Shalika; Er Amandeep Singh Walia. 2017. Big Data and Hadoop challenges and issues. Retrieved from https://search.proquest.com/openview/3e15dbad558fb028b7a30bc26ea1c866/1?pq-origsite=gscholar&cbl=1606379
17. Jeff Kelly. 2014. Hadoop, Business Analytics. Retrieved from http://wikibon.org/wiki/v/Big_Data:_Hadoop,_Business_Analytics_and_Beyond
18. Sara Landset, Taghi M. Khoshgoftaar, Aaron N. Richter & Tawfiq Hasanin. 2015. Survey of open source tools for machine learning with big data in the Hadoop ecosystem. Retrieved from https://link.springer.com/article/10.1186/s40537-015-0032-1
19. James Le. 2019. An Introduction to Big Data: Distributed Data Processing. Retrieved from https://medium.com/cracking-the-data-science-interview/an-introduction-to-big-data-distributed-data-processing-36654202c6ce
20. Masters' In Data Science Org. Why use Hadoop? Retrieved from https://www.mastersindatascience.org/data-scientist-skills/hadoop/
21. Gordon E Moore. 1965. Moore's Law. Retrieved from https://en.wikipedia.org/wiki/Moore%27s_law
22. Margaret Rouse. 2014. Hadoop key features. Retrieved from https://searchdatamanagement.techtarget.com/definition/Hadoop
23. sas.com Hadoop and its timeline. Retrieved from https://www.sas.com/en_us/insights/big-data/hadoop.html
24. Savaliya, Rajesh; Saxena, Akash. 2019. Comparative Study of Andrew File System and Hadoop Distributed File System Framework to Manage Big Data. Retrieved from https://search.proquest.com/openview/cd09273ffe644a6206fe2ad90257e518/1?pq-origsite=gscholar&cbl=2030964
25. Jianwu Wang; Yan Tang; 2014. Scalable Data Science Workflow Approach for Big Data Bayesian Network Learning. Retrieved from https://ieeexplore.ieee.org/abstract/document/7321725
26. ACM - Association for Computing Machinery. The ACM Computing Classification System (1998). Retrieved from https://web.archive.org/web/20090126131933/http://acm.org:80/about/class/ccs98-html