Benchmarking Hadoop and Spark on Mutiple Platforms

Gregor von Laszewski
Indiana University
Smith Research Center
Bloomington, IN 47408
laszewski@gmail.com

Karan Kamatgi
Indiana University
School of Informatics, Computing, and Engineering
Bloomington, IN 47404
krkamatg@iu.edu

Karan Kotabagi
Indiana University
School of Informatics, Computing, and Engineering
Bloomington, IN 47404
kkotabag@iu.edu

Manoj Joshi
Indiana University
School of Informatics, Computing, and Engineering
Bloomington, IN 47404
manjoshi@iu.edu

Ramyashree DG
Indiana University
School of Informatics, Computing, and Engineering
Bloomington, IN 47404
rdasego@iu.edu

ABSTRACT

Big data and cloud computing computing embraces the potential benefits such as high volume, high velocity and high variety. Where as in distributed computing, a task is processed over two or more computers in the network within distributed system by breaking down the problem into many parallel smaller task. This method allows to combine computional power of many computer to execute a program involving large data or multiple iterations in parallel. Apache Spark and Hadoop are two frameworks which are trending technologies in the field of distributed computing. In this project we are comparing both frameworks by running a task implemented in different technology on a Raspberry pi distributed system setup with a single master and 4 workers. The final results are used to benchmark the performance in terms of efficiency and time elapsed. This analysis would help us realize scenarios and business use cases for each of these frameworks.

KEYWORDS

hid-sp18-406,hid-sp18-408,hid-sp18-410,hid-sp18-412, Spark,Raspberry Pi,Hadoop,Spark Cluster

1 INTRODUCTION

Bigdata is ruling today's business and with technology boom and digitalization, large amount data As we all know big data has gained wide range of importance in various aspects. Big data, as a word refers to huge amount of data which are both organized and disorganized. However, the main important thing here is not the amount of data, instead how this data is used by organizations. Big data, plays a vital role in having clear insights about the growth of business. Big data has determined its importance in various factors such as deducing the root cause of failed features, customizing coupon generation based on the customerfis buying habits, risk analysis of the business organization, determining irregular activities through data analysis which can affect the organizationfis growth. Big data has marked its importance in various organizations across practically every industry such as banking, education, government, health care, manufacturing and retail. To be specific Big data is used in various fields such as understanding and targeting customers, optimizing business process, personal qualification and performance optimization, improving health care and public health, Improvising business in sports, machine and device performance, improving security and law enforcement, financial trading and also in improving science and research. Thus, all these various applications of big data are not possible from the raw form of the data, it is required to process and analyze this data [34].

2 INTRODUCTION TO HADOOP

2.1 Why do we need Hadoop?

Data is generated continuously and rigorously from each and every application in almost all the platforms such as social media, mobile platform and many more. The solutions to handle this data should be very quick and also should consider business cost required for this analysis.The problem with early storage tools such as RDBMS is that it is unable to process the semi-structured and unstructured data such as text files, videos, audios, clickstream data etc and it is suitable only for structured data such as banking transaction, location information, etc. These both forms of data are completely different in ways required for processing the data. The main flaw in RDBMS is that it is unable to scale vertically with large number of CPU and other storages. It is a main problem if the main server is down, this created the need for distributed system which can be robust and handle scalability. Hadoop is trusted for its effective management of largely sized data which is both structred and unstructured in different forms such as XML, JSON, and also text at high fault-error tolerance. It is noticed that with clusters of many servers in the field of horizontal scalability, Hadoop has marked its significance by providing faster result rates from Big Data and also the unstructured data as Hadoop's architecture is based on the open source foundation. Other challenges faced with the big data where keeping the huge amount of data secure, analyzing the data without knowing the type of the data, dealing with data which has poor quality and is inconsistent and incomplete. Also, finding powerful algorithms to find patterns and insights was not an easy task. On the other hand, the organizations and enterprises realized that the amount of information which is used for analyzing is less and majority of the data is getting wasted. The main reason behind this is that the organizations lacks power and strong tools

to analyze such huge amount of data. Due to these limitations of processesing data, huge amount of valuable data was termed as unwanted and discarded. It is useful and important to collect and keep all the data in a safe storage, which the business organizations realized [24].

Hadoop emerged as a strongest tool to deal with the above challenges of the big data. Now, almost all organizations use big data analysis to improve the functionalities in each and every business unit. The various kinds of business units include research, design, development, marketing, advertising, sales and customer handling. Sharing such huge amount of data across different platforms is also challenging, in such scenarios hadoop is used to create a pond. Hadoop in this way represents repository of various sources of data which may be from intrinsic or extrinsic sources. Hadoop is a set of open source programs and procedures which can be used for big data operations. It is an open source framework from Apache which is comprised of Java based programming framework that can be used for data storage and processing of the data-sets clusters on commodity hardware. The basic idea behind hadoop is that to make the computation of the data faster. Hadoop MapReduce is composed of shared and integrated foundation where developers can include additional tools and enhance the framework. Additionally, scalability is the core of the hadoop system. The novel approach which emerged hadoop is that by storing the all types of data available, focus on organizing and analyzing the data in new interesting ways [27].

The growth of Hadoop has marked its significance in changing the perception of handling Big Data, specifically for the unstructured data. Using Hadoop we can enable excess data streamlining for any distributed processing system across clusters. Hadoop system helps scaling up from single server to a large number of servers, and also assuring local computation and storage space on every distributed server. Hadoop does not depend on hardware to provide high-availability; instead it makes its software efficient by building robust libraries. These libraries will handle the breakdown at the application layer and thus helping the service to be efficient along with the cluster of computers [37].

Hadoop as framework has established credibility for various factors. It allows the users to easily frame test cases on distributed systems. These test cases are efficiently deployed across various machines in turn and it also utilizes the parallelism of the CPU cores. Another interesting feature of hadoop is that it does not rely on hardware to take care of the fault tolerance, instead hadoop is designed to detect the failures and handle them at the application layer itself. Additionally, servers configuration, addition and removal is independent of the hadoop operation and these servers can be added and detached to the clusters dynamically during the operations. One of the major advantage and main reason for hadoop to be successful is that being an open source framework it is suitable for all the frameworks as it is Java based [32].

In simple words to state, hadoop provides efficient ways of storing enormous data sets by distributing it to various clusters and then execute distributed analysis application in each cluster. The other main important feature about hadoop is that, it is a framework which is modular and is compatible to integrate with any of the required modules. Thus, it allows swapping of the components according the need of software tool which results in flexible architecture, which makes hadoop robust and efficient. To the business organizations, the flexibility feature of hadoop is most advantageous, as it allows the business users to add or modify their data storage and analysis according to the business needs and suitable software.

2.2 Overview of Hadoop Modules

Apache's hadoop framework in composed of the four main modules:

- Hadoop Common: This holds the libraries and utilities required by the Hadoop modules.
- Hadoop Distributed File System: which manages to store data on commodity machines, which is helpful in providing efficient bandwidth across the cluster.
- Hadoop Yarn: this is basically a resource management platform which will be responsible managing compute resources in clusters and using them for scheduling of user's applications.
- Hadoop MapReduce: This is the programming module required for large scale data processing. The main principle behind the above mentioned modules of Hadoop is that the common scenarios of hardware failures is handled by software in the framework [25].

2.3 Importance of Yarn

Yet Another Resource Negotiator (YARN) is one of the most important components of Apache hadoop as it takes care of managing resources and scheduling tasks and thus forming the clustering platform. YARN is responsible for setting up the global and application specific components which are required for resource management. For any particular application, the required and suitable resources are allocated by the YARN. Initially, the application submission client will submit an application to the resource manager of YARN. Further, YARN takes the responsibility of scheduling application so that tasks are prioritized and big data analytics of the system can be managed. This results as the greater step of system architecture for collecting the data and sorting them and further conducting requirement specific queries such as data retrieval. Such type of information retrieval has found very great business values because the organizations can use data analyzing platforms for supply chain maintenance, product documentation, and various service operations such as maintaining customer information and also for maintaining automated processes of business [36].

Being an essential part of core hadoop projects, YARN is capable of allowing multiple data processing engines like interactive SQL, real-time streaming, data science and batch processing. Likewise YARN is growing to be a mandatory foundation for new generation hadoop tools and has become an important component which is required for realizing modern data architecture. Additionally, YARN is extending the capabilities of hadoop by providing support for the new technologies which will be found within the data center. This helps in various factors such as cost effective solutions, linear scalability for storing and processing. Further, YARN is still in progress for improvising factors specifically for the new engines which are emerging to interact with data storage and analysis. Thus,

YARN is one amongst the reliable architectural center of hadoop and definitely the most important foundational component [40].

2.4 Advantages of Hadoop

- Scalable : Hadoop is specifically designed to have a very flat scalability structure. Once After a Hadoop program is written and is functioning on ten nodes, very little work is required for that same program to run on a much larger setup. The underlying Hadoop platform will manage the data and hardware resources and provide dependable performance growth (but proportionate to the number of machines available). Also, it is a highly scalable storage platform, because it can store and distribute large data sets across hundreds of inexpensive servers that operate in parallel. Hadoop enables businesses to run applications on thousands of nodes involving thousands of terabytes of data [29].
- Advanced data analysis can be done in-house: With Hadoop environment its capable to work with large data sets and customize the outcome without having to outsource the task. Keeping operations in-house helps organizations be more agile, while also avoiding the ongoing operational expense of outsourcing.
- Organizations can fully leverage their data: With Hadoop systems, organizations can take full advantage of all their data fi?! structured and unstructured, real-time and historical.Earlier with traditional legacy systems, all the available data and inputs were not used to do analysis to support business activity. Leveraging adds more value to the data itself and improves the return on investment (ROI) for the legacy systems.
- Flexible architecture: Some of the tasks that Hadoop is being used for today were formerly run by expensive computer systems. Hadoop commonly runs on commodity hardware. Because of big data standard, Hadoop is supported by a large and competitive solution provider community, which protects customers from vendor lock-in [23].
- Cost Effective: Hadoop systems provides cost effective storage solution for businesses. The problem with traditional-systems is that it is extremely cost prohibitive to process such massive volumes of data. To reduce costs, many companies in the past would have had to down-sample data and classify it based on certain assumptions as to which data was the most valuable. The raw data would be deleted, as it would be too cost-prohibitive to keep. While this approach may have worked in the short term providing very cost-effective solution for expanding datasets. It is designed to scale-out architecture that can affordably store all company's data for use sometime later. This saves a lot of cost and improves the storage capability tremendously.
- Enhances Speed: Hadoop's unique storage method is based on a distributed file system that basically maps data wherever it is located on a cluster. The tools for data processing are often on the same servers where the data is located, resulting in much faster data processing. If youfire dealing with large volumes of unstructured data, it can efficiently process terabytes of data in just minutes [30].
- Great Data Reliability: Data reliability is one aspect that no organization wants to compromise on. Hadoop provides complete confidence and reliability; in a scenario where data loss happens on a regular basis, HDFS helps you solve the issue. It stores and delivers all data without compromising on any aspect, at the same time keeping costs down. Whether you are a start-up, a government organization, or an internet giant, Hadoop has proved its mettle when it comes to strong data reliability in a variety of production applications at full scale.
- Comprehensive Authentication and Security: All businesses are looking for software that makes their work safe, secure and authenticated. When it comes to authentication and security, Hadoop provides an advantage over other software. Its HBase [5] security, along with HDFS and MapReduce, allows only approved users to operate on secured data, thereby securing an entire system from unwanted or illegal access. Security is the top priority of every organization. Any unlawful access to data is sure to harm business dealings and operations [35].
- Flexible with Range of data sources: Hadoop enables businesses to easily access new data sources and tap into different types of data. The data collected from various sources will be of structured or unstructured form. This means organizations can use Hadoop to derive valuable business insights from data sources such as social media, clickstream data, or email conversations.A lot of time would need to be allotted to convert all the collected data into single format. Hadoop saves this time as it can derive valuable data from any form of data. It also has a variety of functions such as data warehousing, fraud detection, market campaign analysis etc [26].
- Resilient to failure: A key advantage of using Hadoop is its fault tolerance. When data is sent to an individual node, that data is also replicated to other nodes in the cluster, which means that in the event of failure, there is another copy available for use [28].

3 INTRODUCTION TO APACHE SPARK

Apache Spark is a cluster computing framework developed at the University of California, Berkeley. It is maintained as an open software by the Apache Software Foundation. It provides an interface for programming with data parallelization and also fault tolerance. Spark's architecture is based on Resilient Distributed Dataset (RDD) which is a read-only set of data items split over a cluster of machines. In short, instead of the code getting the data for computation, it goes to the location where the data is present [38].

3.1 RDD

RDD is a fundamental data structure of Spark. It is an immutable distributed collection of objects wherein each dataset is divided into logical partitions. Computation is performed on different nodes of the cluster. RDD's can have any type of objects like Java, Scala, Python objects or even user-defined class objects. An RDD is a

read-only, collection of records that are partitioned. RDD's can be created through operations on data on storage or using other RDD's. RDD's are a collection of elements which are fault-tolerant and operate parallely. RDD's can be created in two ways - parallelizing an existing collection in the driver program, or referencing a dataset in an external storage system [31].

Spark and its RDD's were developed to overcome the limitations of Map-Reduce in Hadoop which forces a linear dataflow on distributed programs. Map-Reduce programs read the data from the disk, map some function across the data, do the reduce operation on the results of the map and finally store the results from reduce onto a disk. On the other hand, Spark's RDD function by giving a working set for distributed programs and offer distributed shared memory in a restricted way.

Spark provides platform for implementation of iterative algorithms and interactive data analysis. Iterative algorithms visit the data set multiple times and interactive data analysis involves repeated querying of database. The latency of such applications can be reduced greatly when compared to a Map-Reduce implementation of the task.

Apache Spark requires a manger to handle the cluster load and also needs a distributed storage system. For cluster management, Spark supports standalone cluster, Hadoop Yarn and Apache Mesos. To provide distributed storage, Spark can be integrated with a variety of frameworks like Hadoop Distributed File System (HDFS), MapR File System (MapR-FS) [16], Cassandra [3], Kudu [8]. Also, a custom solution can be implemented and integrated into Spark [38].

3.2 Spark Core

Spark has a core component called the Spark Core that provides distributed task dispatching, Input-Output functionalities and scheduling exposed through an API centered on RDD abstraction. A driver program invokes parallel operations on an RDD by passing a function to Apache Spark which then schedules the function's execution in parallel on the whole cluster. Operations such as joins, take RDD's as an input and produce new RDD's. RDD's are immutable and fault-tolerance is achieved by keeping a track of the lineage of each RDD so that if there is a data loss, it can be reconstructed. RDD's support any type of Python, Scala or Java objects. Along with the RDD style of programming, Spark provides two restricted forms of shared variables - broadcast variables and accumulators. Broadcast variables reference read-only data and makes it available on all nodes. Accumulators can be used to program reduce part in an imperative style [38].

3.3 Spark SQL

Spark SQL is a component that is built on top of Spark Core that provides a data abstraction called DataFrame. DataFrame provides support for structured and semi-structured data. To support various languages like Scala, Java or Python, Spark SQL provides domain-specific language (DSL). Spark streaming makes use of Spark Core's fast scheduling capacity to perform streaming analytics. It takes in data in mini-batches and performs RDD transformations on those mini-batches of data. This enables the same set of application code written for batch analytics to be used for streaming analytics as well enabling easy implementation of lambda architecture. Spark Streaming has support to consume from Kafka [7], Twitter [19], Flume [4], Kinesis [1] and TCP/IP sockets [38].

3.4 Spark MLlib

Spark MLlib (Machine Learning Library) is a distributed machine learning framework that is built on top of Spark Core. Many common machine learning and statistical algorithms have been implemented in MLlib which facilitates large scale machine learning pipelines. Some of the supervised and unsupervised Machine Learning algorithms implemented are Support Vector Machine, Logistic Regression, Linear Regression, Decision Trees, Naive Bayes, Latent Dirichlect Allocation [38].

3.5 Spark GraphX

GraphX is a distributed graph processing framework built on top of Apache Spark. Since RDD's are immutable, graphs are also immutable and hence GraphX is unsuitable for graphs that need to be updated. GraphX provides two seperate application programming interface for implementing parallel algorithms - a Pregel abstraction and a MapReduce style API [38].

4 APPLICATIONS OF SPARK

The applications of Apache Spark range from Streaming Data, Machine Learning, Interactive Analysis and Fog Computing [13].

4.1 Spark Streaming Data

Spark's key use case is its ability to process continuously streaming data. Spark Streaming unifies different types of data processing capabilities allowing developers to use just a single framework to handle the processing needs. The most common ways that Spark Streaming is used today are Streaming ETL, Data Enrichment, Trigger Event Detection and Complex Session Analysis. Streaming ETL - Traditional ETL tools used for batch processing, read the data, convert it to a database compatible format and finally write it to the database. In Streaming ETL, data is cleaned and aggregated continually before it is pushed into data stores. Data Enrichment - This streaming capability enriches live data by combining it with static data. This allows user to conduct more real-time data analysis. Trigger event detection - Spark Streaming allows user to detect and respond quickly to unusual behaviors in real-time. Complex Session Analysis - Using the Spark Streaming feature, events with respect to live sessions such as user activity in a website or an application. These session informations can be used continuously to update the machine learning models. Machine Learning: Apache Spark comes integrated with a framework for performing advanced analytics. It helps users run repeated queries on data sets which amounts to processing machine learning algorithms. Spark supports machine learning through its framework called MLlib which can perform clustering, classification among many other algorithms [33].

4.2 Spark Interactive Analysis

One of the most notable features of Spark is its ability to provide interactive analytics. Unlike Hadoop, Spark performs exploratory queries without sampling. By integrating Spark with visualization tools, complex data sets can be processed visualized in an interactive way. Fog Computing: Fog computing decentralizes data processing

and storage instead of performing computation on the edge of the network . However, this creates a lot of complexities for processing decentralized data since it requires low latency along with massive parallel processing for running machine learning algorithms. But with Spark Streaming, real-time querying tool (Shark), MLlib and GraphX, Spark qualifies as a fog computing solution [33].

5 RASPBERRY PI

Raspberry Pi is a series of mini single-board computers. It was developed by Raspberry Pi Foundation to promote teaching of basic computer science. Till date, various Raspberry Pi's have been released with a system on chip and ARM compatible CPU along with on-chip GPU. The processor speed ranges from 700 MHz to 1.4 GHz and on-board memory ranges from 256 MB to 1 GB RAM. SD cards can be used to store operating system and program memory. The boards have USB ports ranging from one to four ports. For video output, HDMI is supported and 3.5 mm phono jack supports audio output. The Raspberry Pi Foundation provides Raspbian, a Debian-based Linux distribution along with Ubuntu, Windows 10 IoT Core and specialized media center distributions. It supports Python and Scratch as the main programming language and also supports many other languages [39].

6 DIFFERENCE BETWEEN SPARK AND MAPREDUCE

Spark keeps data in memory whereas MapReduce keeps shuffling things. MapReduce takes a long time to write things to disk and also to read them back. This makes MapReduce slow. For SQL queries, a chain of MapReduce operations are usually required and since MapReduce keeps shuffling things, it required a lot of Input-Output activity. On the other hand, when SQL queries are run on Spark, it executes faster since it has data in memory and requires less Input-Output operations. Spark has a Map and a Reduce function like MapReduce, but it also has richer features such as Filter, Join and Group-by. Hence development in Spark is easier and is flexible. Spark provides a lot of instructions at a higher level of abstraction than what MapReduce provides [22].

7 HADOOP ARCHITECTURE

7.1 General Introduction about the Hadoop Architecture

Apache Hadoop is distributed computation framework that that provides the storage and capability of running the programs on multiple master slave architecture in order to enable the distributed processing. The very basic implementation of the Apache Hadoop started with the Googlefis MapReduce programming model, which now has grown to an enormous ecosystem with related technologies such as Apache Hive, Apache Spark and Apache HBase. The architectural pattern of the Hadoop has made it so famous that it has been adopted by many of the companies such as Facebook, Yahoo, Adobe, Cisco, ebay etc [14].

7.2 HDFS Architecture

HDFS architecture was designed with the main goals of avoiding hardware failures, large datasets corresponding to around gigabytes and terabytes, portability and a simple coherent model [15].

The data is divided into the small units called as the blocks in hadoop cluster and subsequently data is distributed across the cluster. The duplication of data in blocks is done twice, and totally with the three copies. The two of copies are called as replicas and are stored at a different place in cluster. As a result of three copies, the data replication factor increase to three leading to increase in availability of data. When a certain copy is lost then the HDFS will again re-replicate the data in order to maintain the value of the replication factor to three [14].

There are mainly two models of the HDFS architecture which varies based on the characteristics and version of the Hadoop, they are as follows [14]:

- Vanilla HDFS
- High Availability HDFS

The main basic architecture of the HDFS follows a leader/follower pattern. A cluster would be specifically composed of a single Namenode, an optional secondary node and the random number of the data nodes [15].

The structuring of Vanilla Hadoop Deployment is shown in Figure 1.

[Figure 1 about here.]

7.2.1 Namenode and DataNodes. The HDFS cluster composes of a single Namenode, this node acts as the master server which handles filesystem namespace in order to control relevant access to files by the clients. There are random number of datanodes that enable storage of the data in the nodes they are attached to. In order to effectively allow the user data to be stored in files the HDFS exposes a file system namespace which enables the storage of user data in the files. Subsequently, file is further divided into one or more blocks which will be stored in the set of the data nodes. The execution of tasks with respect to the opening, closing, renaming files and directories and the other namespace functionalities are carried out by namenode. The namenode also serves in order to decide the mapping of blocks to the datanodes. The read and write requests from the clients of the file system are fulfilled by datanodes [15].

The HDFS architecture is shown in Figure 2.

[Figure 2 about here.]

The general platform that the namenodes and the datanodes run are on the GNU/Linux operating system. The main primary language used to build the HDFS is java. A typical machine that can support Java will be able to run namenode and the datanode. As system uses high portable java language HDFS can be easily deployed on wide range of the machines. With presence of the single namenode in the cluster there is simplification of HDFS architecture. The Namenode acts as an arbitrator and a repository for whole of the HDFS metadata. The system is designed by treating NameNode as black box and it never allows user data to flow out of the Namenode [15].

7.2.2 File System NameSpace. The file organization in HDFS is similar to the traditional hierarchical file organization. The HDFS filesystem provides functionalities to the users to create, add, rename, remove and move the files to respective directories. The HDFS file system provide effective support that enables user quotas and grants permission to access the file system [15].

The namespace of filesystem is managed by the Namenode. Any of the transactions with respect to the file system namespace or properties will be recorded by the NameNode. The replication factor can be controlled and changed by any of the respective client application. The data with respect to replication factor is also stored by NameNode [15].

7.2.3 fsimage and edit log. File system metadata is stored in two of the different files by NameNode, those are fsimage and the editlog. At the specific event of time the fsimage facilitates storage of file systems metadata. The few of changes that are going to be updated such as renaming, appending to a file are maintained in an edit log to ensure durability. This avoids redundant creation of the new fsimage every time when a namespace is modified [14].

Structuring of the secondary NameNode is shown in Figure 3.

[Figure 3 about here.]

The Figure 3 shows the structuring of the secondary NameNode. Secondary NameNode independently validates and updates replicas of fsimage subsequently to the changes that are made in edit log. If active node fails in system then NameNode needs to rebuild edit log on top of fsimage. Although, administrators of cluster can easily retrieve updated copy of fsImage from Secondary Namenode [14].

The `secondaryNameNode` also provides the advantage and benefit of the faster recovery in case of NameNode failure. Although, `SecondaryNameNode` will not provide automatic failover load balancing [14].

7.3 MapReduce

The Mapreduce is an effective framework that can be utilized to process large datasets in the distributed way. Basically, there are three operations in the MapReduce model, Map input dataset into the collective pairs, shuffle mapped data and handover the data to the reducers then subsequently reduce the over all pairs with same number of keys. The example of map-reduce job correspond to the WordCount example which is again illustrated in this paper that has been followed to analyse the performance of hadoop and spark on the different platforms. The high-level specification or unit in MapReduce framework is Job. The each of job composes of one or more map or the reduce tasks [14].

The following Figure 4 shows the perfect example of the MapReduce job with respect to WordCount application.

[Figure 4 about here.]

7.3.1 Related Frameworks. There are other frameworks that perform and accomplish the MapReduce jobs in a similar way as hadoop. The most used frameworks other than the Hadoop are Apache Spark and Apache Tez [11]. This paper also describes Apache spark and other relevant analysis that is performed with respect to Apache Spark with different computing platforms and different computing resources [14].

7.4 YARN

The major idea of the YARN is to divide the functionalities of resource management and scheduling/management into separate daemons. The very basic and the primary idea is to have the global resource manager (RM) and the per-application master. The application would be treated as a single job or Directed Acyclic graph of Jobs [20].

The following figure describes the architecture of the YARN 5.

[Figure 5 about here.]

7.4.1 Data Computation Framework. The data computation framework is formed by resource manager and node manager. The major authority is possessed by resource manager to assign resources to all applications in a system. In contrast, Node manager is framework that gets assigned to each machine and is responsible for monitoring of containers and resource allocation with respect to cpu, memory, disk and network [20].

7.4.2 Resource Manager. The following figure 6 describes architecture of the resource manager and displays high-level view of resource manager. The resource manager (RM) is assigned responsibility of tracking the resources in cluster and the scheduling jobs. The standby resource manager provides failover load balancing for active resource manager. The standby resource manager will be waiting to take over in case of any of disaster with respect to the active resource manager [20].

The following figure describes architecture of resource manager 6.

[Figure 6 about here.]

7.4.3 Node Manager. Node manager acts like a slave in a system infrastructure. When the node manager starts it indicates resource manager that it has gone live. At continuous periodic times node manager sends respective signals to Resource Manager indicating that it is live. At next level of depth, container is fraction of node manager and it is in turn used by client to run an application [21].

The following figure displays the division of the Node Manager into containers 7.

[Figure 7 about here.]

8 APACHE HADOOP ARCHITECTURE DEPLOYED ON THE RASPBERRY PI 3

The figure 8 depicts architecture of Apache Hadoop on Raspberry Pi nodes. There are basically 5 nodes which have been divided into the master and workers. YARN is deployed on top of Apache Hadoop. There are five nodes and 4 of them are treated and configured as workers and one node is configured as master. Further, Java and Python WordCount applications are run in order to benchmark, evaluate and compare the performance of Hadoop and spark applications.

Following figure 8 describes architecture of Apache Hadoop Deployed on Raspberry Pi 3.

[Figure 8 about here.]

9 DOCKER

The docker provides the containerization at operating systems level. The Docker virtualization system is developed by Docker. Inc. To

effectively develop and maintain the containers easily on the single Linux Instance docker provides effective framework to run different capability containers in a single machine [12]. The docker consists of three following components:-

9.1 Software

The docker has the major Daemon process called as `dockerd` which will effectively manage the docker containers and in turn handle container objects. Docker client called as docker helps users to manage and provides diverse range of options to interact with containers [12].

9.2 Objects

The various objects in docker are mainly images, containers and services. Docker container acts as a black-box that runs applications virtually. These containers can be managed by the docker API service. The docker images are templates that can be used to build containers and acts as a base framework. The docker service is a process that will effectively allow numerous containers to be extended across different docker Daemons [12].

9.3 Registries

Registries are of two types, public and private. At present the Docker hub and the Docker Cloud are the two main registries. Docker Hub provides different images to be tagged and pulled [12].

9.4 Docker-Spark

This project is mainly targeted to benchmark and evaluate the performance of Apache Hadoop and standalone spark on different platforms. Docker was one platform that was selected in this project to evaluate the performance of an application with these frameworks. The dockerized WordCount program is run on the Ubuntu 17.10 platform.

9.5 Docker-Spark Architecture

The following figure 9 describes the implementation of the docker spark architecture that is deployed on Ubuntu 17.10.

[Figure 9 about here.]

10 SPARK ARCHITECTURE

10.1 Overview

Spark offers great features in terms of speed, simplicity and high level support for development of current environments and storage systems. They have wide range of developers that embrace the ongoing development for one of the Apache's largest and most active platform with around 500 contributors who are responsible for their code in the software code release [18].

10.2 Support for programming languages

Spark provides very easy interface and comprehensive support for the enhancement and development languages so that we could easily learn and apply accordingly into our existing applications as flexibily as possible. Spark provides support for modern languages such as Java, Python, Scala, SQl and R. It has been a great concern for most people that Python performs sub-optimally when compared to its alternate language Java. This concern is less significant in a distributed environment mostly where Spark would be deployed. The slight loss in performance if introduced by the usage of Python language could be compensated elsewhere in the design and distributed operations of the cluster. One of the analysis carried out, informs us that familiarity of the user's chosen language is often times more important than the raw speed of code in that particular language.

Relational databases (RDBMS) which usually comprises of the structured query language (SQL) and the SQL queries are normally well understood by developers, Engineers, Data Scientists and Academia. The Apache Spark module Spark SQ offers native support for SQL. Spark provides excellent support and simplifies whole process of making query request to the data that is stored in the Spark's native RDD (Resilient Distributed Dataset) model along with data from other external sources like relational databases and data warehouses.

Spark also provides extensive support for data science enriched language called R. It has a package named SparkR which first appeared in the release of 1.4 apache spark. Due to enormous popularity in the data science, this package proves to be one of the important features of Apache Spark [18].

10.3 Spark deployment options

Spark is very easy to download and install on a laptop or virtual machine. It is designed to be able to deploy on both standalone as well as part of a cluster.Most of the production environment workloads require computations to be carried at large scale and Spark would then be deployed on existing big data cluster. These clusters are used for running Hadoop jobs and Hadoop's YARN resource manager will manage the hadoop cluster. There are several options for running spark such as running spark on YARN (Yet another Resource Negotiator) which is a cluster management system. Yarn could be used to execute arbitrary applications on a hadoop cluster. There needs to be one application master node and several arbitrary number of containers acting as workers. Workers are responsible for the execution of application where as the master makes requests to the workers and monitors the progress and status of the work carried out by the respective workers [18].

YARN consists of two component types

- Resource Manager is a unique component for the whole of big data cluster. The main job of Resource Manager is to grant the requested resources and balancing the load of the entire cluster.It also starts the application master only in the initial phase and restarts the master in case of any failure.
- Each computing node (Worker) has one node manager which is executed when the master initiates the request. The node manager starts and monitors the containers assigned to the current node in the cluster as well monitors the usage of its resources such as CPU and memory consumption.

The other option for running spark where people prefer alternative resource managers, Spark could be run on other clusters controlled by Apache Mesos [9]. Spark also provides series of scripts bundled with current releases of Spark simplifies the process of

launching spark on Amazon web services Elastic compute cloud EC2 [18].

10.4 Storage options for Spark

Spark can also easily integrate with myriad of commercial and open source 3rd party data storage systems such as MapR, Google Cloud, Amazon S3 [2] ,Apache Cassandra, Apache Hadoop, Apache Hbase and Apache Hive [6] [18].

10.5 Spark stack

The spark project comprises of stack components mentioned in the Figure 10 and mainly consists of Spark core and 4 main libraries that were optimized to address the requirements of several different use cases. Any application would technically require Spark Core and at least one of these four libraries.The flexibility and main benefit of spark would become evident when applications that require combination of two or more libraries run efficiently on top of Spark core [18].

[Figure 10 about here.]

10.6 Spark Core

The heart of the spark lies at the Spark Core and is responsible for management functions like task scheduling, monitoring and other watchdog services. Spark implements and is based on a programming abstraction known as RDD (Resilient Distributed Dataset) [18].

10.7 RDD Design flow

RDD is designed to support in-memory data storage and distributed across a cluster in manner that is fault tolerant and efficient. Fault tolerance was achieved in part by tracking lineage of transformations that were applied to coarse grained data sets.Efficiency was achieved through process parallelization across multiple nodes in the cluster and minimization of data replication between respective nodes. Once we load the data into RDD, two basic types of operations are carried out [10].

- Transformations - this creates a new RDD by changing by actually changing the original by mechanisms such as filtering and mapping.
- Actions- some of the operations such as measuring the count which do not change the original data and the original RDD remains untouched throughout the process. Here the chain of transformations from RDD1 to RDDm would be logged and this process could be repeated in the scenarios such as data loss or failure of cluster node.

Transformations tend to follow lazy evaluations in the sense that they would not be executed until a subsequent action/operation has an absolute need for result of previous action.This lazy evaluation helps in improving the performance as it could avoid unnecessary processing of the data. This would also introduce processing bottlenecks that could cause applications to stall while waiting for a processing action to conclude.

Spark tries to keep these RDDs in memory that greatly increase the performance of the cluster [10].

10.8 Reason for slowness in Data sharing on MapReduce

MapReduce is one of the most widely adopted technology for processing and generating large Datasets in parallel with a distributed algorithm on a cluster. Users were able to write parallel computations with help of high level operators and not worry about work distribution and fault tolerance. But in most of the current frameworks, the only way to reuse data between computations was write an external stable storage system such as HDFS. The fast growing user base needs interactive and interactive application that require data sharing at a faster pace. The sharing of data between jobs is slow in MapReduce due to replication, serialization and disk IO. Most of the hadoop applications tend to spend maximum time in reading and writing tasks in HDFS [10].

10.9 Iterative operations in MapReduce

The idea behind executing iterative tasks in MapReduce is to reuse intermediate results across multiple computations in multi storage applications. The Figure 11 clearly explains the working of the current framework in context of execution of interactive operations on Mapreduce. This clearly incurs substantial overheads due to I/O, replication of data and serialization that in turn degrades the system performance [10].

[Figure 11 about here.]

10.10 Interactive tasks in MapReduce

General users run ad-hoc queries on the same subset of data where each query would perform disk I/O on the stable storage that use most of the execution time. Figure 12 explains the current framework on execution of interactive queries on MapReduce [10].

[Figure 12 about here.]

10.11 Data sharing in Spark RDD

The key idea of spark is RDD which supports in-memory process computation.It stores the state of memory as object across the jobs and the object could be shared between respective jobs.Obviously data sharing is much faster than network and disk.

10.12 Iterative operations in Spark RDD

Illustration in the Figure 13 explains the iterative operations in spark RDD which stores the intermediate results in a distributed memory rather than stable storage such as HDFS and thus makes the system faster. Also if the distributed memory does not suffice to store the intermediate results then Spark stores those results on disk [10].

[Figure 13 about here.]

10.13 Interactive operations in Spark RDD

Figure 14 provides illustration of execution of interactive operation on Spark RDD where in if different queries are executed on same data set then that particular data is stored in memory for better execution times. Each transformed RDD might be re-computed each time we run an action against it.We could also persist an RDD in memory and spark would keep elements on the cluster for much faster access and increase the efficiency of whole process [10].

[Figure 14 about here.]

10.14 Spark Cluster Architecture

We have implemented spark cluster in two different ways

- Spark standalone cluster
- Running spark on Yarn on top of Hadoop

Figure (Master-worker diagram) explains the architectural set up of both mentioned ways of implementing spark cluster.

For the first method we did not install Hadoop and so basically does not consist of stable storage system (HDFS) . The basic cluster build configuration were followed and we assume that we have a working cluster with 1 master and 4 workers and each name node is able to ssh to other name nodes in the cluster.

All the spark related configuration were mentioned in the file spark-env.sh and all 4 workers were listed in the slaves file. The second method uses Hadoop on YARN configuration where YARN provides all the information required for spark cluster to run successfully.

11 RESULTS AND CONCLUSION

[Table 1 about here.]

[Table 2 about here.]

[Table 3 about here.]

The results clearly compare the performance of Hadoop and Spark on different platforms and using different languages such as Java and Python. Clearly, standalone Spark outperforms Hadoop(HDFS) because it persists the data in memory whereas the HDFS based Hadoop performs extra overhead operations such as shuffling of data and hence executes slower. Docker stores data in its own cache for subsequent runs, and hence the performace enhances for second and third run. The first run takes a long time on Docker since it has to register all the workers before execution.

At the outset it looks like Apache Spark seems to be performing better but at times HDFS Hadoop provides enough fault tolerance and stable storage which might help other applications. Hence both the frameworks have pros and cons and usage of these framework are greatly application specific.

12 WORK BREAKDOWN

Initial configuration of all the Raspberry Pi and network configuration required for the clusters were equally contributed by all the team members. Karan Kamatgi and Karan Kotabagi did the configurations and server setup for the Spark Standalone cluster. Ramyashree DG and Manoj Joshi did the configuration and server setup for Hadoop-Yarn-Spark cluster and Karan Kamatgi contributed for the Yarn configuration of this cluster. Python Docker setup was contributed by Karan Kotabagi. Spark-Java benchmarking program was contributed by Ramyashree DG, Spark-Python benchmarking program was contributed by Manoj Joshi. Hadoop-Java benchmarking program was done by Karan Kotabagi and the Hadoop-Python benchmarking program was contributed by Karan Kamatgi. Evaluation and benchmarking were carefully analysed by all the team members as a group. All the team members have equally contributed to the sections of project paper, configuration steps for the Raspberry Pi cluster and also have put equal team efforts to resolve the issues faced during the cluster configuration and evaluation process.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Gregor von Laszewski for his support and suggestions to write this paper.

REFERENCES

[1] [n. d.]. Amazon Kinesis. Web Page. ([n. d.]). https://aws.amazon.com/kinesis/
[2] [n. d.]. Amazon S3. Web Page. ([n. d.]). https://aws.amazon.com/s3/
[3] [n. d.]. Apache Cassandra. Web Page. ([n. d.]). http://cassandra.apache.org/
[4] [n. d.]. Apache Flume. Web Page. ([n. d.]). https://flume.apache.org/
[5] [n. d.]. Apache HBase. Web Page. ([n. d.]). https://hbase.apache.org/
[6] [n. d.]. Apache Hive. Web Page. ([n. d.]). https://hive.apache.org/
[7] [n. d.]. Apache Kafka. Web Page. ([n. d.]). https://kafka.apache.org/
[8] [n. d.]. Apache Kudu. Web Page. ([n. d.]). https://kudu.apache.org/
[9] [n. d.]. Apache Mesos. Web Page. ([n. d.]). http://mesos.apache.org/
[10] [n. d.]. Apache Spark RDD. Web Page. ([n. d.]). https://www.tutorialspoint.com/apache_spark/apache_spark_rdd.htm
[11] [n. d.]. Apache TEZ. Web Page. ([n. d.]). https://tez.apache.org/
[12] [n. d.]. Docker. Web Page. ([n. d.]). https://en.wikipedia.org/wiki/Docker_(software)
[13] [n. d.]. Fog Computing. Web Page. ([n. d.]). https://en.wikipedia.org/wiki/Fog_computing
[14] [n. d.]. Hadoop Architecture. Web Page. ([n. d.]). https://www.datadoghq.com/blog/hadoop-architecture-overview/
[15] [n. d.]. HDFS Architecture. Web Page. ([n. d.]). https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html
[16] [n. d.]. MapR-FS. Web Page. ([n. d.]). https://mapr.com/products/mapr-fs/
[17] [n. d.]. Resource Manager architecture. Web Page. ([n. d.]). https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/ResourceManagerHA.html
[18] [n. d.]. Spark Architecture Overview. Web Page. ([n. d.]). https://mapr.com/ebooks/spark/03-apache-spark-architecture-overview.html
[19] [n. d.]. Twitter. Web Page. ([n. d.]). https://twitter.com/
[20] [n. d.]. YARN architecture. Web Page. ([n. d.]). https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html
[21] [n. d.]. YARN Internal overview. Web Page. ([n. d.]). http://ercoppa.github.io/HadoopInternals/HadoopArchitectureOverview.html
[22] Nitin Bandugula. [n. d.]. The 5-Minute Guide to Understanding the Significance of Apache Spark. Web Page. ([n. d.]). https://mapr.com/blog/5-minute-guide-understanding-significance-apache-spark/
[23] bmc. 2016. *Advantages of using Hadoop.* bmc. http://www.bmcsoftware.in/guides/hadoop-benefits-business-case.html
[24] Infinity Consulting. 2015. *Benefits and Importance of Hadoop.* Infinity Consulting. http://infinitylimited.co.uk/benefits-and-importance-of-hadoop-big-data-platform/
[25] Haley. 2013. *Hadoop: What It Is And How It Works.* readwrite. https://readwrite.com/2013/05/23/hadoop-what-it-is-and-how-it-works/
[26] KnowledgeHut. 2016. *Top Pros and Cons of Hadoop.* knowledgehut. https://www.knowledgehut.com/blog/bigdata-hadoop/top-pros-and-cons-of-hadoop
[27] Davin Lewis. 2015. *Introduction to Hadoop.* Code club. http://www.abstract-thoughts.com/tech-talk/introduction-to-hadoop
[28] MindsMapped. 2016. *HADOOP ADVANTAGES AND DISADVANTAGES.* mindsmapped. http://blogs.mindsmapped.com/bigdatahadoop/hadoop-advantages-and-disadvantages/
[29] Michele Nemschoff. 2013. *Big data: 5 major advantages of Hadoop.* ITProPortal. https://www.itproportal.com/2013/12/20/big-data-5-major-advantages-of-hadoop/
[30] Ipsita Pattnaik. 2017. *Advantages of using Hadoop.* e-Zest. http://blog.e-zest.com/advantages-of-using-hadoop
[31] Tutorials Point. [n. d.]. Apache Spark - RDD. Web Page. ([n. d.]). https://www.tutorialspoint.com/apache_spark/apache_spark_rdd.htm
[32] Ritika Prasad. 2017. *Why is hadoop important.* Quora. https://www.quora.com/Why-is-Hadoop-important
[33] Qubole. [n. d.]. Top Apache Spark Use Cases. Web Page. ([n. d.]). https://www.qubole.com/blog/apache-spark-use-cases/
[34] Sas. 2015. *Big Data, What it is and why it matters.* Sas. https://www.sas.com/en_us/insights/big-data/what-is-big-data.html
[35] Technavio. 2016. *Top 5 Benefits of Using Hadoop.* technavio. https://www.technavio.com/blog/top-5-benefits-using-hadoop
[36] techopedia. 2017. *Hadoop Yarn.* techopedia. https://www.techopedia.com/definition/30154/hadoop-yarn
[37] Jagadish Thaker. 2016. *Big data Week.* BDW. http://blog.bigdataweek.com/2016/08/01/hadoop-important-handling-big-data/

[38] Wikipedia. [n. d.]. Apache Spark. Web Page. ([n. d.]). https://en.wikipedia.org/wiki/Apache_Spark
[39] Wikipedia. [n. d.]. Raspberry Pi. Web Page. ([n. d.]). https://en.wikipedia.org/wiki/Raspberry_Pi
[40] Horton Works. 2014. *Apache Hadoop Yarn*. Horton Works. https://hortonworks.com/apache/yarn/#section_1

A CHKTEX

```
make[2]: Entering directory '/home/manoj/hid-sp18-412/project-paper'
cd ../../hid-sample; git pull
Already up-to-date.
cp ../../hid-sample/paper/Makefile .
cp ../../hid-sample/paper/report.tex .
WRNING: line longer than 80 characters
('     ', '81:', 'high volume, high velocity and high variety. Where as in distributed computing, \n')
WRNING: line longer than 80 characters
('     ', '81:', 'a task is processed over two or more computers in the network within distributed\n')
WRNING: line longer than 80 characters
('     ', '81:', ' distributed system setup with a single master and 4 workers. The final results \n')
WRNING: line longer than 80 characters
('     ', '81:', ' are used to benchmark the performance in terms of efficiency and time elapsed. \n')
WRNING: line longer than 80 characters
('     ', '81:', "Bigdata is ruling today's business and with technology boom and digitalization, \n")
WRNING: line longer than 80 characters
('     ', '81:', 'large amount data As we all know big data has gained wide range of importance in\n')
WRNING: line longer than 80 characters
('     ', '81:', 'used in various fields such as understanding and targeting customers, optimizing\n')
WRNING: line longer than 80 characters
('     ', '81:', 'Data is generated continuously and rigorously from each and every application in\n')
WRNING: line longer than 80 characters
('     ', '81:', 'unstructured data such as  text files, videos, audios, clickstream data etc and\n')
WRNING: line longer than 80 characters
('     ', '81:', 'required for processing the data. The main flaw in RDBMS is that it is unable to\n')
WRNING: line longer than 80 characters
('     ', '81:', 'problem if the main server is down, this created the need for distributed system\n')
WRNING: line longer than 80 characters
('     ', '81:', 'which can be robust and handle scalability. Hadoop is trusted for its effective \n')
WRNING: line longer than 80 characters
('     ', '81:', 'different forms such as XML, JSON, and also text at high fault-error tolerance. \n')
WRNING: line longer than 80 characters
('     ', '81:', 'scalability, Hadoop has marked its significance by providing faster result rates\n')
WRNING: line longer than 80 characters
('     ', '81:', 'realized that the amount of information which is used for analyzing is less and \n')
WRNING: line longer than 80 characters
('     ', '81:', 'majority of the data is getting wasted. The main reason behind this is that the \n')
WRNING: line longer than 80 characters
('     ', '81:', 'organizations lacks power and strong tools to analyze such huge amount of data. \n')
WRNING: line longer than 80 characters
('     ', '81:', 'Due to these limitations of processesing data, huge amount of valuable data was \n')
WRNING: line longer than 80 characters
('     ', '81:', 'Hadoop emerged as a strongest  tool to deal with the above challenges of the big\n')
WRNING: line longer than 80 characters
('     ', '81:', 'units include research, design, development, marketing, advertising, sales  and \n')
WRNING: line longer than 80 characters
('     ', '81:', 'customer handling. Sharing such huge amount of data across different platforms  \n')
WRNING: line longer than 80 characters
('     ', '81:', 'is also challenging,  in such scenarios hadoop is used to create a pond. Hadoop \n')
WRNING: line longer than 80 characters
('     ', '81:', 'commodity hardware. The basic idea behind hadoop is that to make the computation\n')
WRNING: line longer than 80 characters
('     ', '81:', 'framework. Additionally, scalability is the core of the hadoop system. The novel\n')
WRNING: line longer than 80 characters
('     ', '81:', 'and thus helping the service to be efficient along with the cluster of computers\n')
WRNING: line longer than 80 characters
('     ', '81:', 'the parallelism of the CPU cores. Another interesting feature of hadoop is that \n')
WRNING: line longer than 80 characters
('     ', '81:', 'it does not rely on hardware to take care of the fault tolerance, instead hadoop\n')
WRNING: line longer than 80 characters
('     ', '81:', 'itself. Additionally, servers configuration, addition and removal is independent\n')
WRNING: line longer than 80 characters
('     ', '81:', 'clusters dynamically during the operations. One of the major advantage and main \n')
WRNING: line longer than 80 characters
('     ', '81:', 'reason for hadoop to be successful is that being an open source framework it is \n')
WRNING: line longer than 80 characters
('     ', '84:', '\\item Hadoop Common: This holds the libraries and utilities required by the Hadoop \n')
WRNING: line longer than 80 characters
('     ', '83:', '\\item Hadoop Yarn: this is basically a resource management platform which will be \n')
WRNING: line longer than 80 characters
('     ', '81:', 'responsible managing compute resources in clusters and using them for scheduling\n')
WRNING: line longer than 80 characters
('     ', '81:', '\\item Hadoop MapReduce: This is the programming module required for large scale \n')
WRNING: line longer than 80 characters
('     ', '81:', 'of Apache hadoop as it takes care of managing resources and scheduling tasks and\n')
WRNING: line longer than 80 characters
('     ', '81:', 'management. For any particular application, the required and suitable resources \n')
WRNING: line longer than 80 characters
('     ', '81:', 'multiple data processing engines like interactive SQL, real-time streaming, data\n')
WRNING: line longer than 80 characters
('     ', '81:', 'foundation for new generation hadoop tools and has become an important component\n')
WRNING: line longer than 80 characters
('     ', '81:', 'which is required for realizing modern data architecture. Additionally, YARN is \n')
WRNING: line longer than 80 characters
('     ', '81:', 'machines available). Also, it is a highly scalable storage platform, because it \n')
WRNING: line longer than 80 characters
('     ', '81:', 'can store and distribute large data sets across hundreds of inexpensive servers \n')
WRNING: line longer than 80 characters
('     ', '82:', 'With Hadoop systems, organizations can take full advantage of all their data \xe2\x80\x93 \n')
WRNING: line longer than 80 characters
('     ', '81:', 'to support business activity. Leveraging adds more value to the data itself and \n')
WRNING: line longer than 80 characters
('     ', '81:', 'expensive computer systems. Hadoop commonly runs on commodity hardware. Because \n')
WRNING: line longer than 80 characters
('     ', '81:', 'expanding datasets. It is designed to scale-out architecture that can affordably\n')
WRNING: line longer than 80 characters
('     ', '81:', 'authenticated. When it comes to authentication and security, Hadoop provides an \n')
WRNING: line longer than 80 characters
('     ', '81:', 'along with HDFS and MapReduce, allows only approved users to operate on secured \n')
WRNING: line longer than 80 characters
('     ', '81:', 'Security is the top priority of every organization. Any unlawful access to data \n')
WRNING: line longer than 80 characters
('     ', '85:', 'is sure to harm business dealings and operations~\\cite{hid-sp18-406-hadoop-intro11}.\n')
WRNING: line longer than 80 characters
('     ', '81:', 'variety of functions such as data warehousing, fraud detection, market campaign \n')
WRNING: line longer than 80 characters
('     ', '81:', 'A key advantage of using Hadoop is its fault tolerance. When data is sent to an \n')
WRNING: line longer than 80 characters
('     ', '81:', 'at the University of California, Berkeley. It is maintained as an open software \n')
WRNING: line longer than 80 characters
('     ', '82:', 'by the Apache Software Foundation. It provides an interface for programming with \n')
WRNING: line longer than 80 characters
('     ', '81:', "data parallelization and also fault tolerance. Spark's architecture is based on \n")
WRNING: line longer than 80 characters
('     ', '81:', 'Resilient Distributed Dataset (RDD) which is a read-only set of data items split\n')
WRNING: line longer than 80 characters
('     ', '91:', 'computation, it goes to the location where the data is present~\\cite{hid-sp18-408-Spark}. \n')
WRNING: line longer than 80 characters
('     ', '81:', "RDD's can be created through operations on data on storage or using other RDD's.\n")
WRNING: line longer than 80 characters
('     ', '81:', "Spark and its RDD's were developed to overcome the limitations of Map-Reduce in \n")
WRNING: line longer than 80 characters
('     ', '81:', 'programs read the data from the disk, map some function across the data, do the \n')
WRNING: line longer than 80 characters
('     ', '81:', "reduce onto a disk. On the other hand, Spark's RDD function by giving a working \n")
WRNING: line longer than 80 characters
('     ', '81:', 'set for distributed programs and offer distributed shared memory in a restricted\n')
WRNING: line longer than 80 characters
('     ', '82:', 'solution can be implemented and integrated into Spark~\\cite{hid-sp18-408-Spark}. \n')
WRNING: line longer than 80 characters
('     ', '81:', 'Spark has a core component called the Spark Core that provides distributed task \n')
WRNING: line longer than 80 characters
('     ', '81:', 'dispatching, Input-Output functionalities and scheduling exposed through an API \n')
WRNING: line longer than 80 characters
('     ', '81:', 'centered on RDD abstraction. A driver program invokes parallel operations on an \n')
WRNING: line longer than 80 characters
('     ', '82:', "execution in parallel on the whole cluster. Operations such as joins, take RDD's \n")
WRNING: line longer than 80 characters
('     ', '106:', 'nodes. Accumulators can be used to program reduce part in an imperative style~\\cite{hid-sp18-408-Spark
WRNING: line longer than 80 characters
('     ', '82:', 'Spark SQL is a component that is built on top of Spark Core that provides a data \n')
WRNING: line longer than 80 characters
('     ', '81:', 'Spark SQL provides domain-specific language (DSL). Spark streaming makes use of \n')
WRNING: line longer than 80 characters
('     ', '81:', 'from Kafka~\\cite{hid-sp18-408-Kafka}, Twitter~\\cite{hid-sp18-408-twitter}, Flume\n')
WRNING: line longer than 80 characters
('     ', '82:', 'GraphX is a distributed graph processing framework built on top of Apache Spark. \n')
WRNING: line longer than 80 characters
('     ', '81:', 'Analysis. Streaming ETL - Traditional ETL tools used for batch processing, read \n')
WRNING: line longer than 80 characters
('     ', '81:', 'the data, convert it to a database compatible format and finally write it to the\n')
WRNING: line longer than 80 characters
('     ', '82:', 'database. In Streaming ETL, data is cleaned and aggregated continually before it \n')
WRNING: line longer than 80 characters
('     ', '82:', 'is pushed into data stores. Data Enrichment - This streaming capability enriches \n')
WRNING: line longer than 80 characters
('     ', '81:', 'to detect and respond quickly to unusual behaviors in real-time. Complex Session\n')
WRNING: line longer than 80 characters
('     ', '81:', 'Machine Learning: Apache Spark comes integrated with a framework for performing \n')
WRNING: line longer than 80 characters
('     ', '81:', 'queries without sampling. By integrating Spark with visualization tools, complex\n')
WRNING: line longer than 80 characters
('     ', '92:', 'and GraphX, Spark qualifies as a fog computing solution~\\cite{hid-sp18-408-Spark-Use-Case}.\n')
WRNING: line longer than 80 characters
('     ', '81:', '1 GB RAM. SD cards can be used to store operating system and program memory. The\n')
WRNING: line longer than 80 characters
('     ', '81:', 'boards have USB ports ranging from one to four ports. For video output, HDMI is \n')
WRNING: line longer than 80 characters
('     ', '81:', 'takes a long time to write things to disk and also to read them back. This makes\n')
WRNING: line longer than 80 characters
('     ', '81:', 'Input-Output activity. On the other hand, when SQL queries are run on Spark, it \n')
WRNING: line longer than 80 characters
('     ', '89:', 'higher level of abstraction than what MapReduce provides~\\cite{hid-sp18-408-Difference}.\n')
WRNING: line longer than 80 characters
('     ', '81:', 'Architecture~\\cite{hid-sp18-412-ResourceManager_Architecture}}\\label{s:archires}\n')
WRNING: line longer than 80 characters
('     ', '81:', 'Spark offers great features in terms of speed, simplicity and high level support\n')
WRNING: line longer than 80 characters
('     ', '81:', " Apache's largest and most active platform with around 500 contributors who are \n")
WRNING: line longer than 80 characters
('     ', '97:', ' responsible for their code in the software code release~\\cite{hid-sp18-410-spark-architecture}.\n')
WRNING: line longer than 80 characters
('     ', '81:', 'Spark provides very easy interface and comprehensive support for the enhancement\n')
WRNING: line longer than 80 characters
('     ', '81:', 'Spark provides support for modern languages such as Java, Python, Scala, SQl and\n')
WRNING: line longer than 80 characters
('     ', '81:', "whole process of making query request to the data that is stored in the Spark's \n")
WRNING: line longer than 80 characters
('     ', '81:', 'Spark also provides extensive support for data science enriched language called \n')
WRNING: line longer than 80 characters
('     ', '81:', 'be carried at large scale and Spark would then be deployed on existing big data \n')
WRNING: line longer than 80 characters
('     ', '110:', 'progress and status of the work carried out by the respective workers~\\cite{hid-sp18-410-spark-archit
WRNING: line longer than 80 characters
```

```
('     ', '85:', '\\item Resource Manager is a unique component for the whole of big data cluster. The \n')
WRNING: line longer than 80 characters
('     ', '85:', '\\item Each computing node (Worker) has one node manager  which is executed when the \n')
WRNING: line longer than 80 characters
('     ', '101:', 'managers, Spark could be run on other clusters controlled by Apache Mesos~\\cite{hid-sp18-410-mesos}.\n')
WRNING: line longer than 80 characters
('     ', '88:', 'party data storage systems such as MapR, Google Cloud, Amazon S3~\\cite{hid-sp18-410-s3}\n')
WRNING: line longer than 80 characters
('     ', '88:', ',Apache Cassandra, Apache Hadoop, Apache Hbase and Apache Hive~\\cite{hid-sp18-410-hive}\n')
WRNING: line longer than 80 characters
('     ', '81:', 'The heart of the spark lies at the Spark Core and is responsible for management \n')
WRNING: line longer than 80 characters
('     ', '81:', 'coarse grained data sets.Efficiency was achieved through process parallelization\n')
WRNING: line longer than 80 characters
('     ', '109:', 'Once we load the data into RDD, two basic types of operations are carried out~\\cite{hid-sp18-410-spark-RDD}.\n')
WRNING: line longer than 80 characters
('     ', '85:', '\\item Transformations - this creates a new RDD by changing by actually changing the \n')
WRNING: line longer than 80 characters
('     ', '87:', '\\item Actions- some of the operations such as measuring the count which do not change \n')
WRNING: line longer than 80 characters
('     ', '81:', 'the original data and the original RDD remains untouched throughout the process.\n')
WRNING: line longer than 80 characters
('     ', '81:', 'generating large Datasets in parallel with a distributed algorithm on a cluster.\n')
WRNING: line longer than 80 characters
('     ', '81:', 'require data sharing at a faster pace. The sharing of data between jobs is slow \n')
WRNING: line longer than 80 characters
('     ', '108:', 'applications tend to spend maximum time in reading and writing tasks in HDFS~\\cite{hid-sp18-410-spark-RDD}.\n')
WRNING: line longer than 80 characters
('     ', '81:', 'The idea behind executing iterative tasks in MapReduce is to reuse intermediate \n')
WRNING: line longer than 80 characters
('     ', '81:', 'Mapreduce. This clearly incurs substantial overheads due to I/O, replication of \n')
WRNING: line longer than 80 characters
('     ', '100:', 'data and serialization that in turn degrades the system performance~\\cite{hid-sp18-410-spark-RDD}.\n')
WRNING: line longer than 80 characters
('     ', '81:', '  \\centering\\includegraphics[width=\\columnwidth]{images/iterative-MapReduce.png}\n')
WRNING: line longer than 80 characters
('     ', '82:', 'would perform disk I/O on the stable storage that use most of the execution time.\n')
WRNING: line longer than 80 characters
('     ', '81:', 'Figure~\\ref{fig:interactive-MapRed} explains the current framework on execution \n')
WRNING: line longer than 80 characters
('     ', '83:', '  \\centering\\includegraphics[width=\\columnwidth]{images/interactive-MapReduce.png}\n')
WRNING: line longer than 80 characters
('     ', '81:', 'memory rather than stable storage such as HDFS and thus makes the system faster.\n')
WRNING: line longer than 80 characters
('     ', '81:', 'interactive operation on Spark RDD where in if different queries are executed on\n')
WRNING: line longer than 80 characters
('     ', '81:', ' execution times. Each transformed RDD might be re-computed each time we run an \n')
WRNING: line longer than 80 characters
('     ', '81:', 'For the first method we did not install Hadoop and so basically does not consist\n')
WRNING: line longer than 80 characters
('     ', '81:', 'All the spark related configuration were mentioned in the file spark-env.sh and \n')
WRNING: line longer than 80 characters
('     ', '81:', 'The second method uses Hadoop on YARN configuration where YARN provides all the \n')
WRNING: line longer than 80 characters
('     ', '81:', ' might help other applications. Hence both the frameworks have pros and cons and\n')
WRNING: line longer than 80 characters
('     ', '81:', 'Karan Kamatgi and Karan Kotabagi did the configurations and server setup for the\n')
WRNING: line longer than 80 characters
('     ', '81:', 'and server setup for Hadoop-Yarn-Spark cluster and Karan Kamatgi contributed for\n')
WRNING: line longer than 80 characters
('     ', '82:', 'Karan Kotabagi. Spark-Java benchmarking program was contributed by Ramyashree DG,\n')
WRNING: line longer than 80 characters
('     ', '82:', 'benchmarking were carefully analysed by all the team members as a group. All the \n')
Warning 26 in content.tex line 248: You ought to remove spaces in front of punctuation.
\item Scalable :
             ^

Warning 39 in content.tex line 327: Double space found.
analysis etc ~\cite{hid-sp18-406-hadoop-intro12}.
             ^

Warning 26 in content.tex line 461: You ought to remove spaces in front of punctuation.
of performing computation on the edge of the network . However, this creates a
                                                     ^

Warning 39 in content.tex line 765: Double space found.
The figure ~\ref{s:archihadoop} depicts
          ^

Warning 39 in content.tex line 951: Double space found.
   ~\cite{hid-sp18-410-spark-architecture}}\label{fig:spark-stack}
   ^

Warning 39 in content.tex line 1034: Double space found.
   ~\cite{hid-sp18-410-spark-RDD}}\label{fig:iterative-MapRed}
   ^

Warning 39 in content.tex line 1068: Double space found.
   ~\cite{hid-sp18-410-spark-RDD}}\label{fig:interactive-MapRed}
   ^

Warning 39 in content.tex line 1103: Double space found.
   ~\cite{hid-sp18-410-spark-RDD}}\label{fig:iterative-spark}
   ^

Warning 39 in content.tex line 1133: Double space found.
   ~\cite{hid-sp18-410-spark-RDD}}\label{fig:interactive-spark}
   ^

Warning 26 in content.tex line 1153: You ought to remove spaces in front of punctuation.
 of stable storage system (HDFS) . The basic cluster build configuration were
                                 ^

Warning 44 in content.tex line 1178: User Regex: -2:Vertical rules in tables are ugly.
    \begin{tabular}{ | c | c | c |}
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Warning 44 in content.tex line 1179: User Regex: -2:Use \toprule, midrule, or \bottomrule from booktabs.
    \hline
    ^^^^^^

Warning 44 in content.tex line 1180: User Regex: -2:Use \toprule, midrule, or \bottomrule from booktabs.
    Data Size & Technology & Execution time (seconds) \\ \hline
                                                      ^^^^^^

Warning 44 in content.tex line 1181: User Regex: -2:Use \toprule, midrule, or \bottomrule from booktabs.
    3.6 MB & Java & 36 \\ \hline
                       ^^^^^^

Warning 44 in content.tex line 1182: User Regex: -2:Use \toprule, midrule, or \bottomrule from booktabs.
    3.6 MB & Python & 23\\ \hline
                       ^^^^^^

Warning 44 in content.tex line 1183: User Regex: -2:Use \toprule, midrule, or \bottomrule from booktabs.
    1.5 MB & Java & 30 \\ \hline
                       ^^^^^^

Warning 44 in content.tex line 1185: User Regex: -2:Use \toprule, midrule, or \bottomrule from booktabs.
    \hline
    ^^^^^^

Warning 44 in content.tex line 1193: User Regex: -2:Vertical rules in tables are ugly.
    \begin{tabular}{ | c | c | c |}
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Warning 44 in content.tex line 1194: User Regex: -2:Use \toprule, midrule, or \bottomrule from booktabs.
    \hline
    ^^^^^^

Warning 44 in content.tex line 1195: User Regex: -2:Use \toprule, midrule, or \bottomrule from booktabs.
    Data Size & Technology & Execution time (seconds) \\ \hline
                                                      ^^^^^^

Warning 44 in content.tex line 1196: User Regex: -2:Use \toprule, midrule, or \bottomrule from booktabs.
    144 KB & Java & 42   \\ \hline
                         ^^^^^^

Warning 44 in content.tex line 1197: User Regex: -2:Use \toprule, midrule, or \bottomrule from booktabs.
    144 kB & Python & 67 \\ \hline
                         ^^^^^^

Warning 44 in content.tex line 1198: User Regex: -2:Use \toprule, midrule, or \bottomrule from booktabs.
    3.6 MB & Java &  320  \\ \hline
                           ^^^^^^

Warning 44 in content.tex line 1200: User Regex: -2:Use \toprule, midrule, or \bottomrule from booktabs.
    \hline
    ^^^^^^

Warning 44 in content.tex line 1208: User Regex: -2:Vertical rules in tables are ugly.
    \begin{tabular}{ | c | c | c |}
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Warning 44 in content.tex line 1209: User Regex: -2:Use \toprule, midrule, or \bottomrule from booktabs.
    \hline
    ^^^^^^

Warning 44 in content.tex line 1210: User Regex: -2:Use \toprule, midrule, or \bottomrule from booktabs.
    Data Size & Iteration & Execution time (seconds) \\ \hline
                                                     ^^^^^^

Warning 44 in content.tex line 1211: User Regex: -2:Use \toprule, midrule, or \bottomrule from booktabs.
    3.6 MB & One & 67  \\ \hline
                       ^^^^^^

Warning 44 in content.tex line 1212: User Regex: -2:Use \toprule, midrule, or \bottomrule from booktabs.
    3.6 MB & Two & 24 \\ \hline
                       ^^^^^^

Warning 44 in content.tex line 1213: User Regex: -2:Use \toprule, midrule, or \bottomrule from booktabs.
    3.6 MB & Three & 22 \\ \hline
                       ^^^^^^

Warning 36 in content.tex line 1220: You should put a space in front of parenthesis.
Clearly, standalone Spark outperforms Hadoop(HDFS) because it persists
                                                 ^

make[2]: Leaving directory '/home/manoj/hid-sp18-412/project-paper'
```

List of Figures

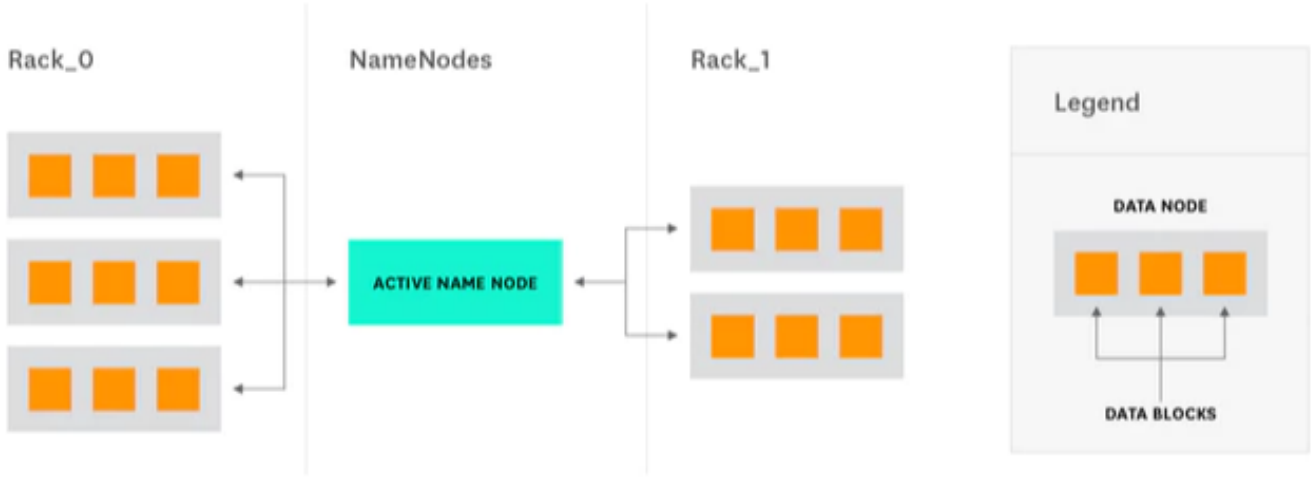

Figure 1: Vanilla Hadoop Architecture [14]

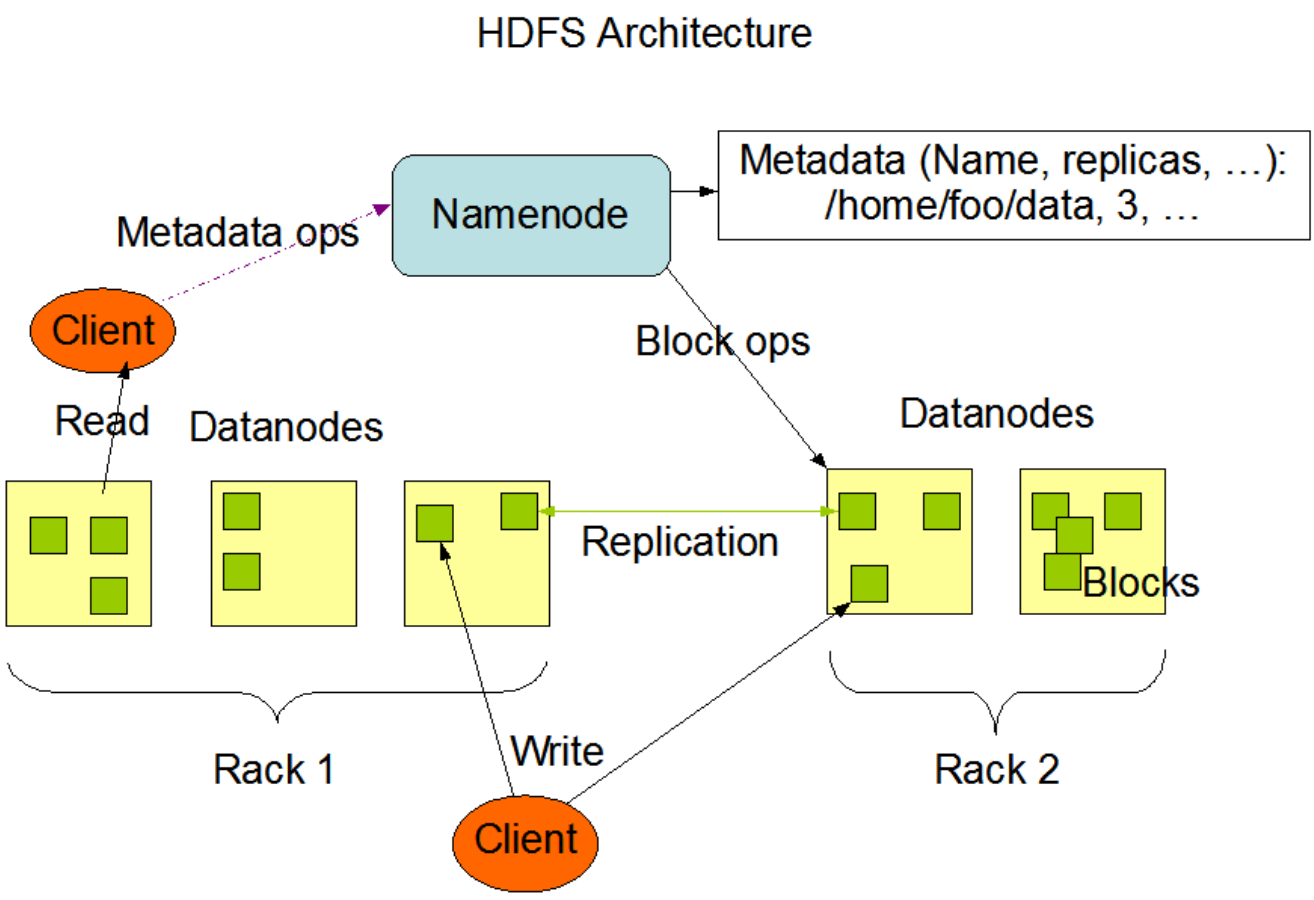

Figure 2: HDFS Architecture [15]

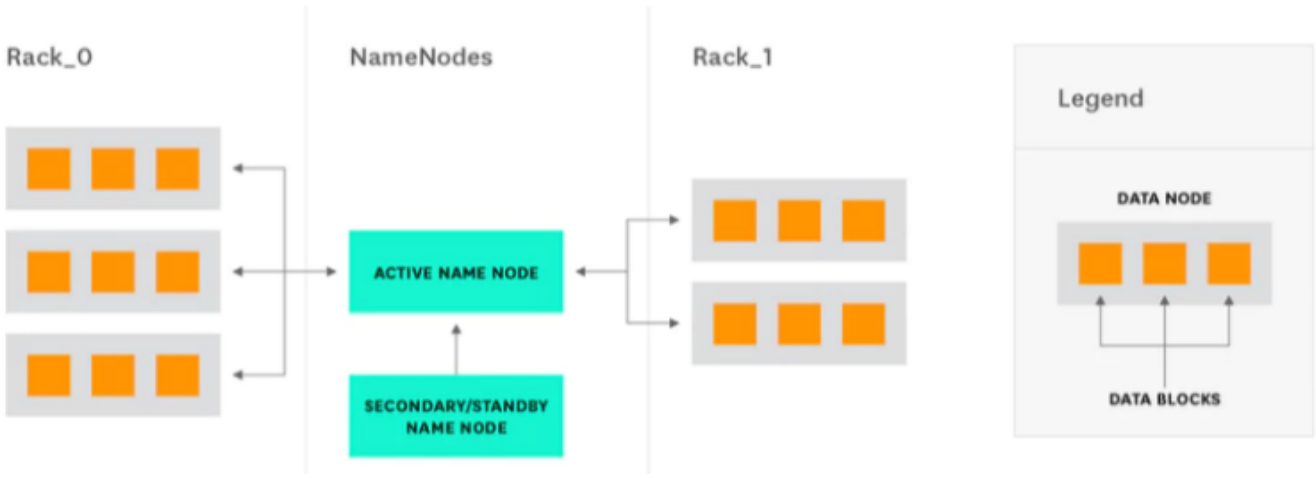

Figure 3: Structuring of Secondary NameNode [14]

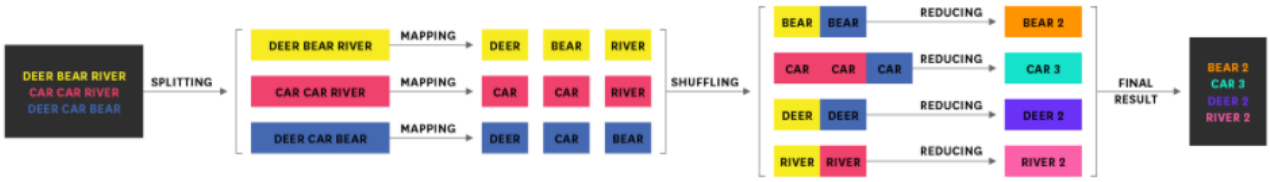

Figure 4: Word Count MapReduce Job [14]

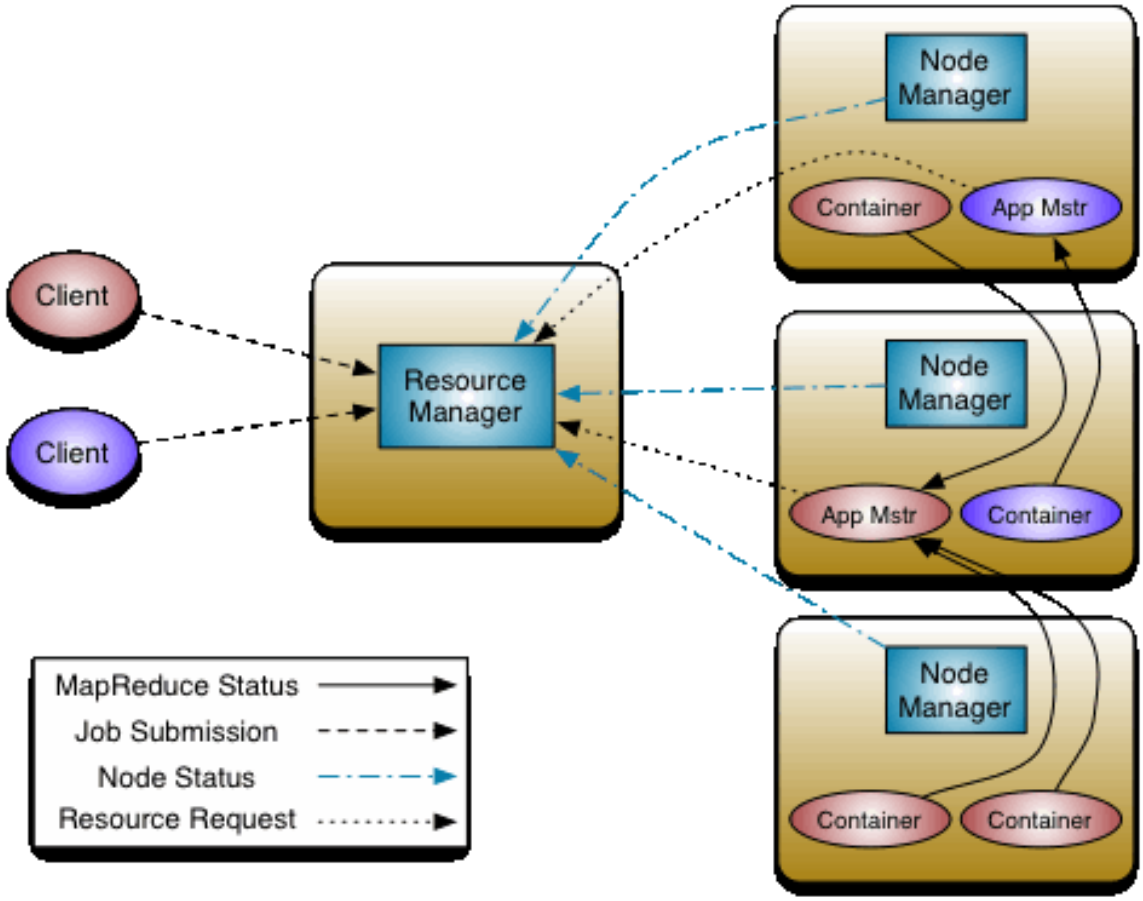

Figure 5: YARN Architecture [20]

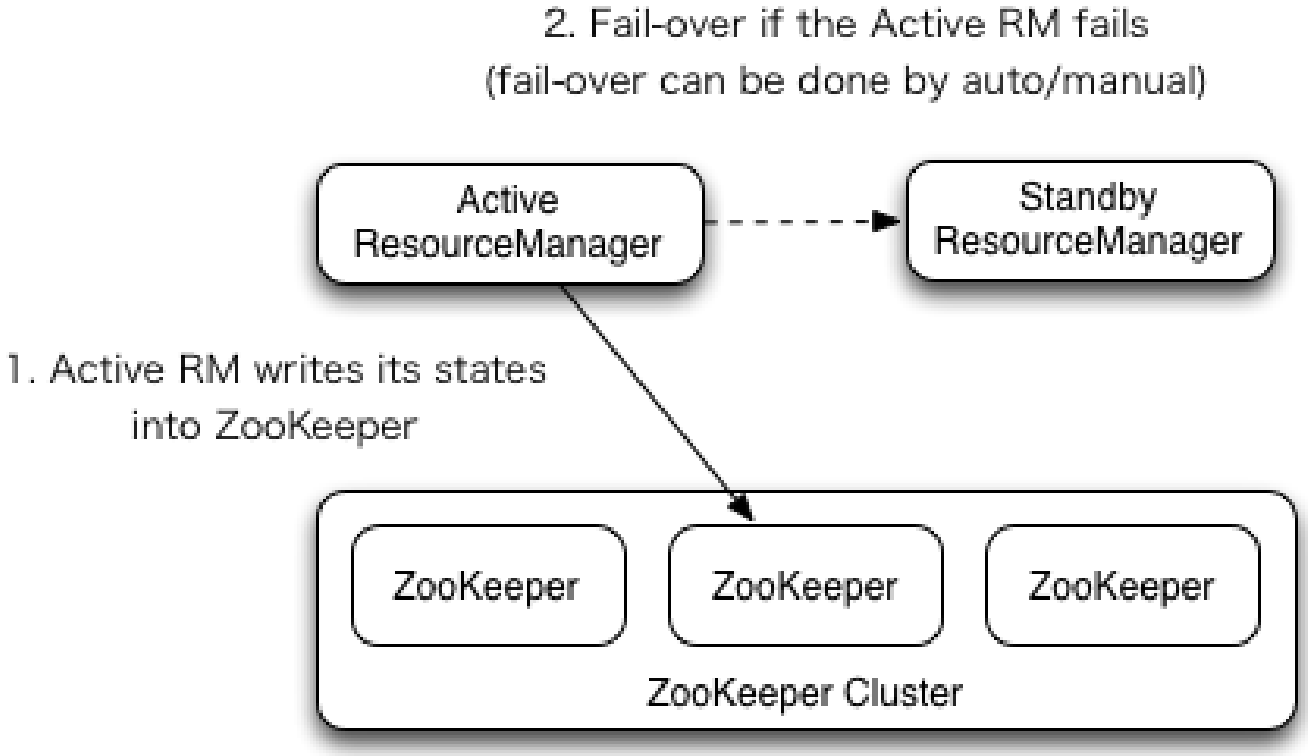

Figure 6: Resource Manager Architecture [17]

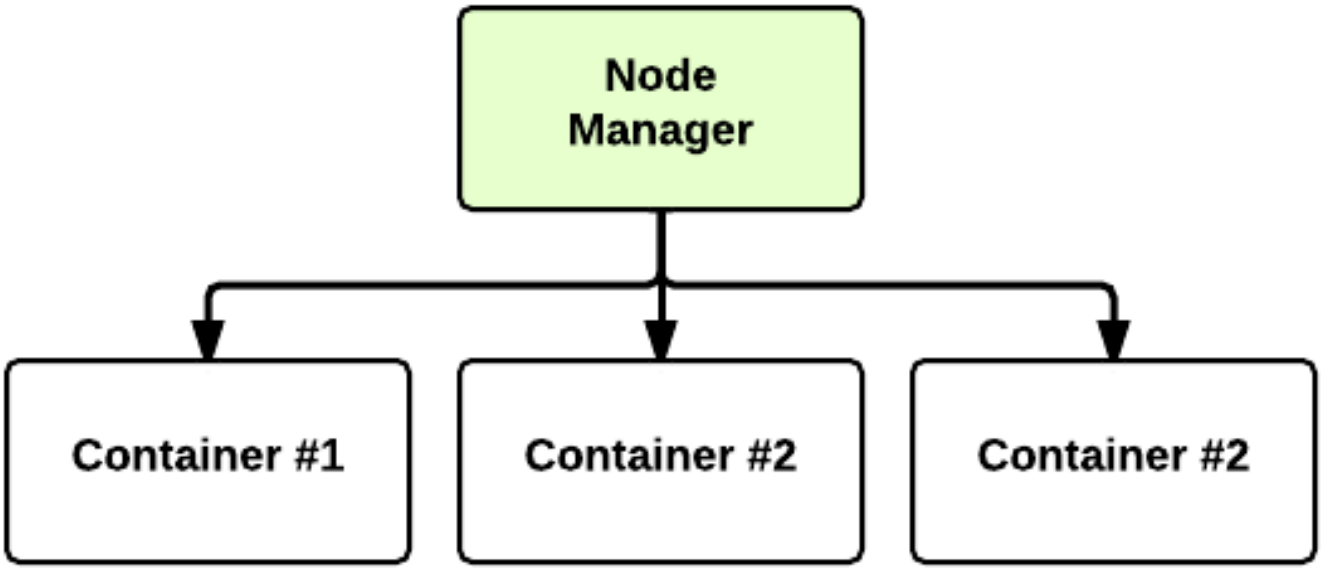

Figure 7: Node Manager Division [21]

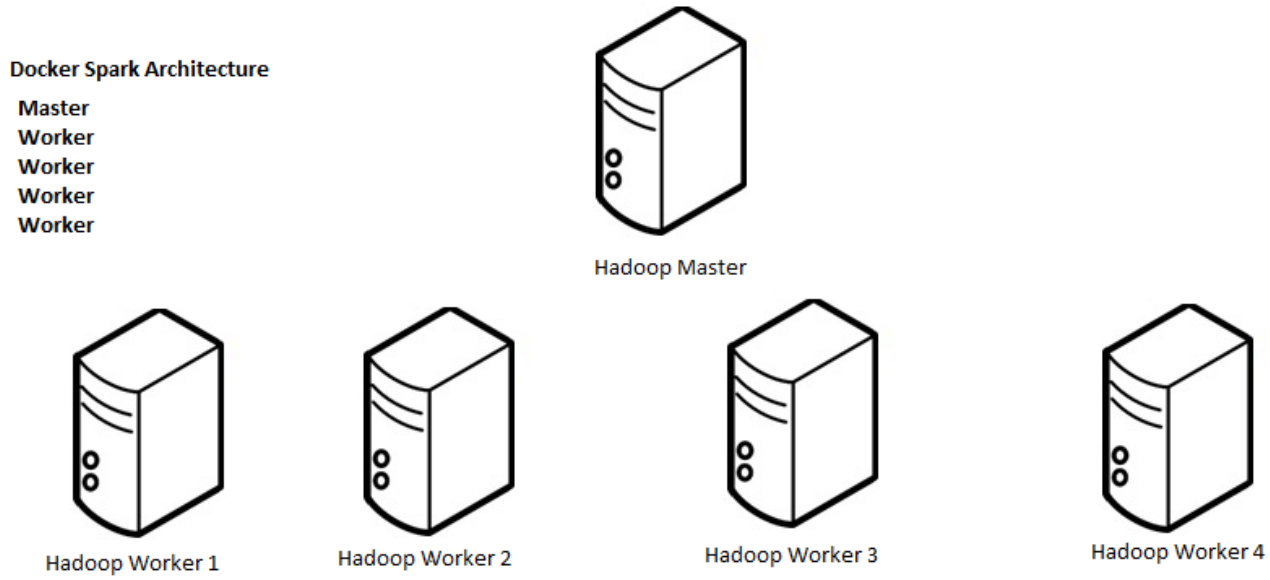

Figure 8: Apache Hadoop Deployment Architecture on RaspberryPi 3

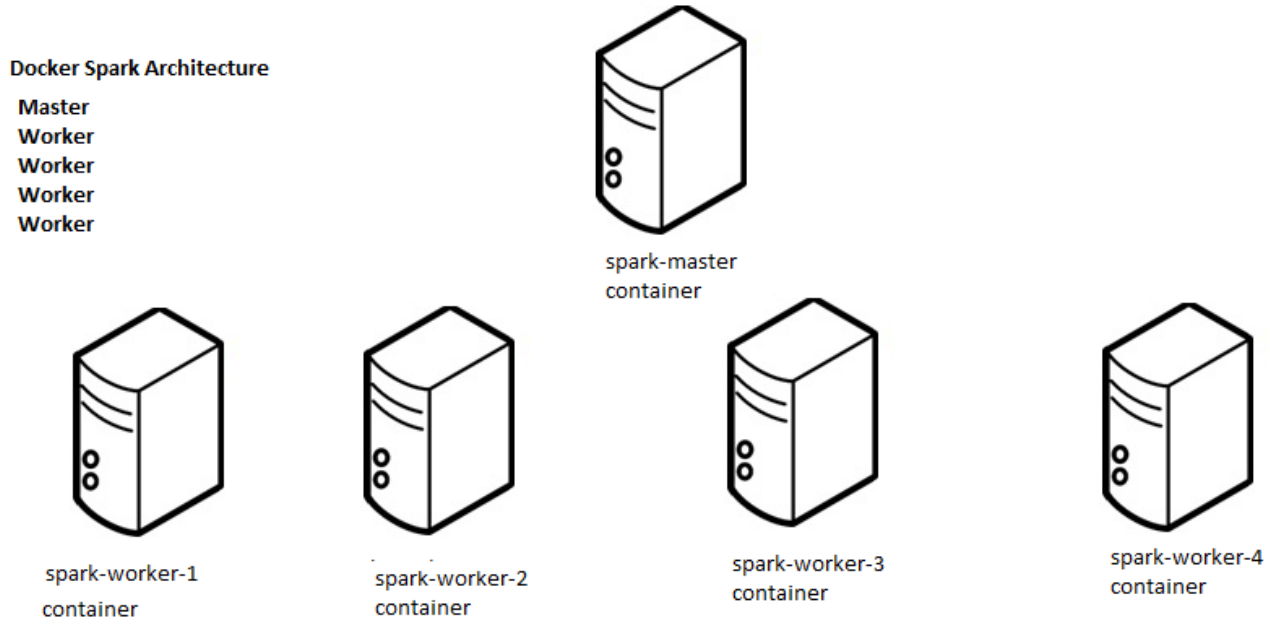

Figure 9: Docker Spark

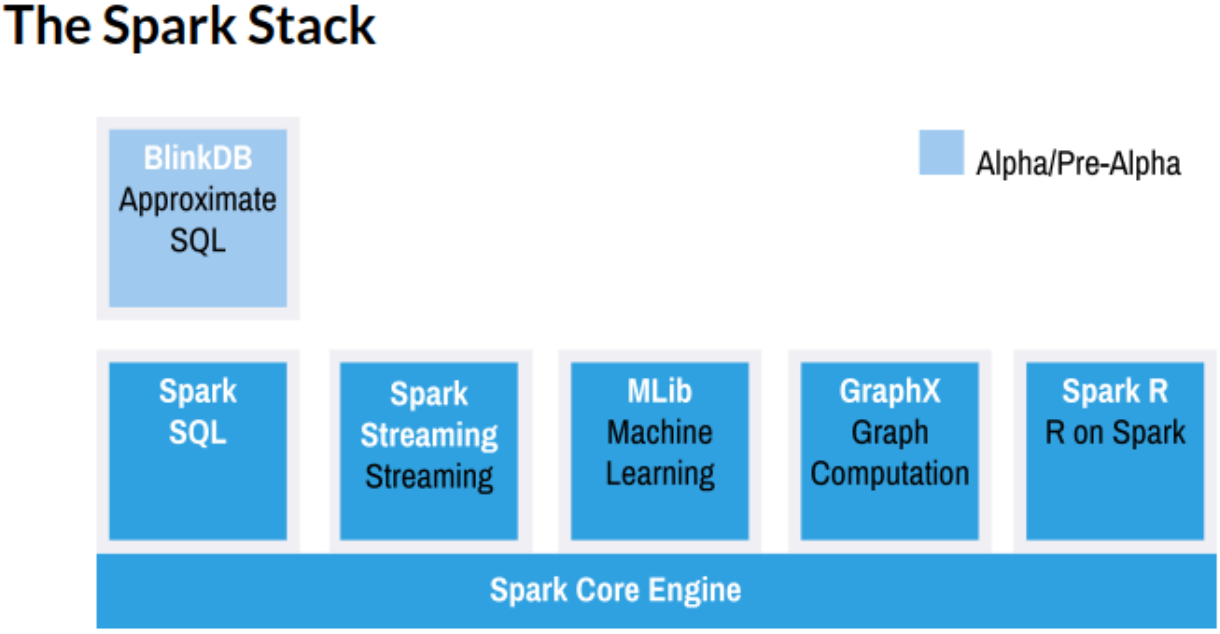

Figure 3-1. Spark Stack Diagram

Figure 10: Spark Stack [18]

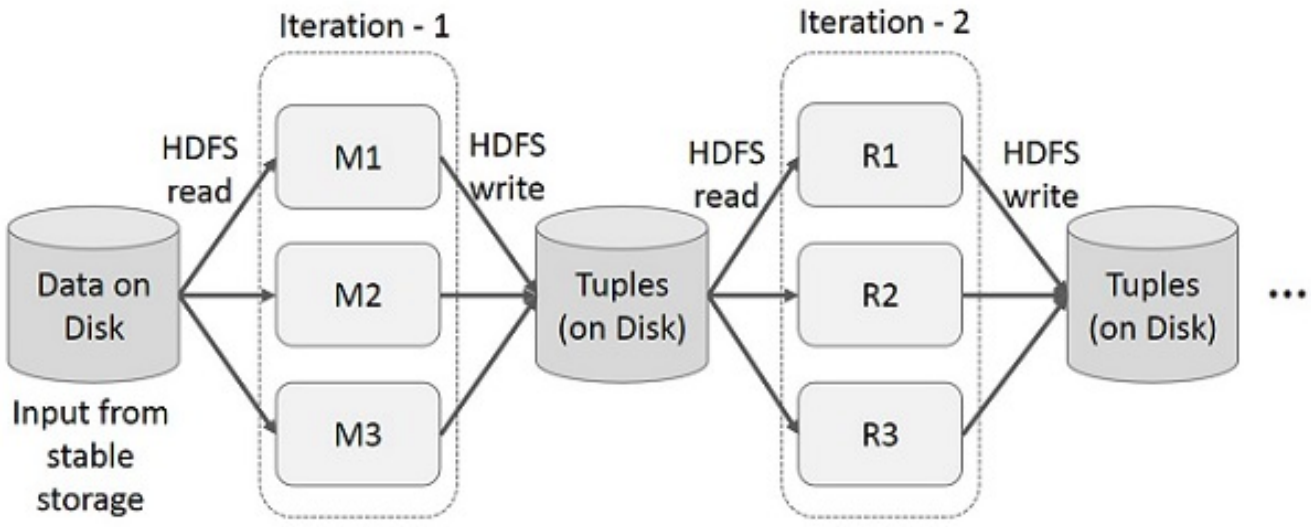

Figure 11: Iterative Operations on MapReduce [10]

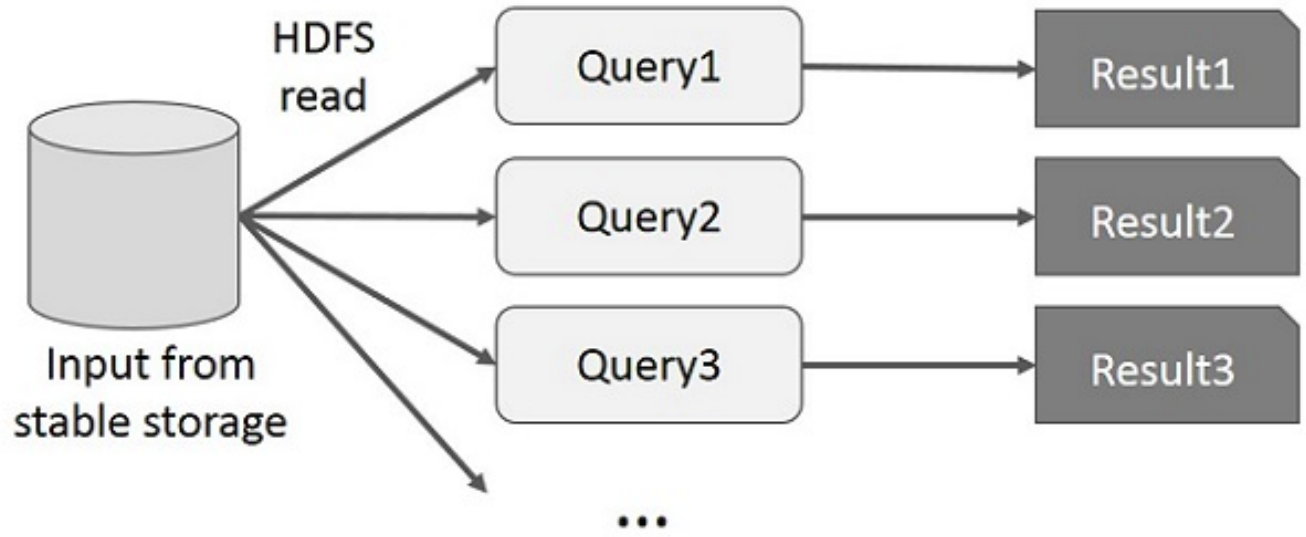

Figure 12: Interactive Operations on MapReduce [10]

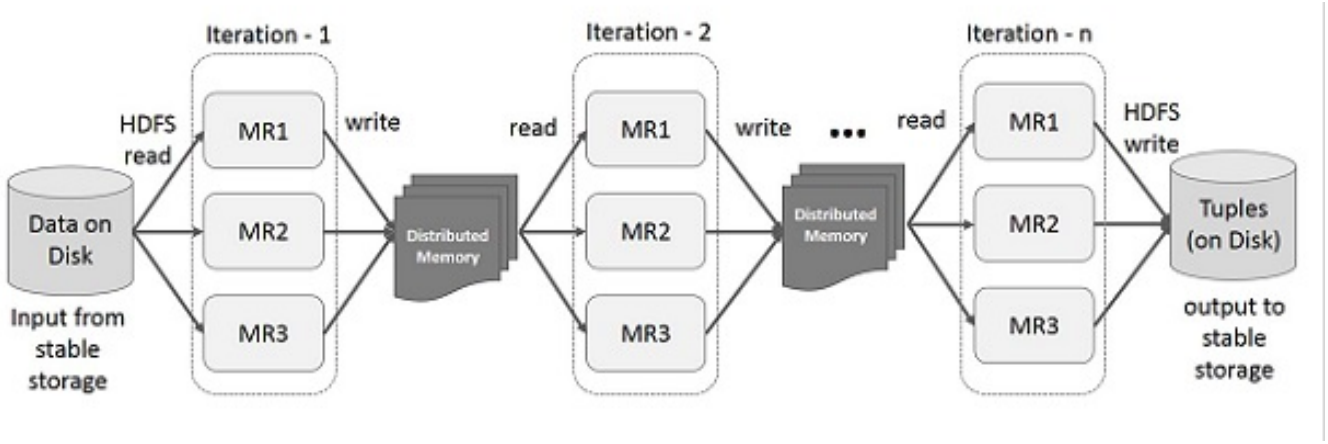

Figure 13: Iterative Operations on Spark RDD [10]

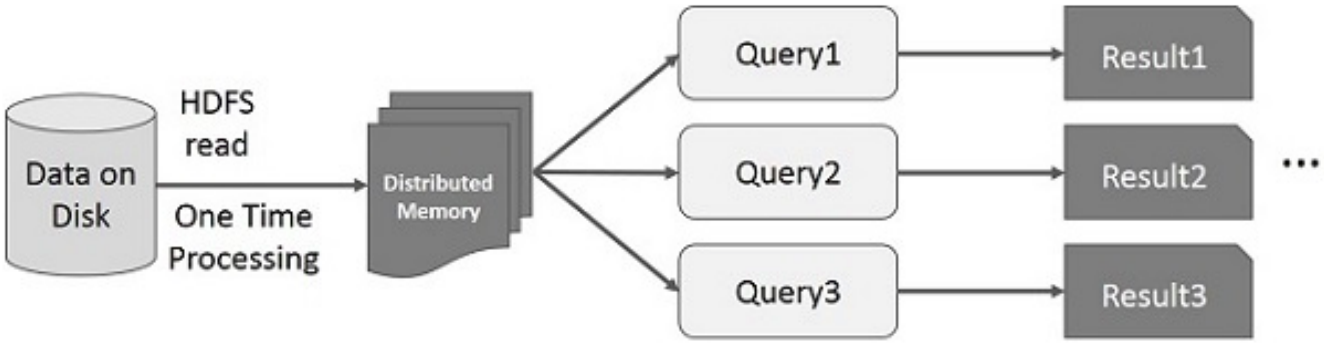

Figure 14: Interactive Operations on Spark RDD [10]

List of Tables

Table 1: Stand Alone Spark Performance

Data Size	Technology	Execution time (seconds)
3.6 MB	Java	36
3.6 MB	Python	23
1.5 MB	Java	30
1.5 MB	Python	18

Table 2: Hadoop-Yarn-Spark Performance

Data Size	Technology	Execution time (seconds)
144 KB	Java	42
144 kB	Python	67
3.6 MB	Java	320
3.6 MB	Python	450

Table 3: Stand Alone Docker Python Performance

Data Size	Iteration	Execution time (seconds)
3.6 MB	One	67
3.6 MB	Two	24
3.6 MB	Three	22