Chapter 1
Introduction

Prevailing variety of applications are generating a huge amount of data regularly, which is much beyond our imagination. According to an IBM study, nearly 2.5 quintillion bytes of data is created every year, so much that about 90% of the data in the world today has been created in the last two years itself (Ibm.com., 2019a). Directly or indirectly, each individual plays a big role in generating this tremendous amount of data, through Social Networking Sites by posting or creating self images or sending audio/video messages etc. This huge amount of data may be referred as "Big Data". The term "Big Data" was coined for the data which is difficult to store, process, and analyze using conventional approaches. The Big Data can be characterised using four V's (Laney, 2001; IBM Big Data & Analytics Hub, 2018): Volume, Velocity, Variety, and Veracity. The first two 'V's i.e. Volume and Velocity relate to the quantitative part of data while the remaining two 'V's i.e. Variety and Veracity relate to the qualitative part of the data. Conceptually, Volume refers to a massive size of data and Velocity refers to the pace with which data is generated. The Variety refers to diversity of data (i.e. unstructured, semi-structured, and structured data) and Veracity refers to trustworthiness of data source.

Big Data storage, processing and its analytics, have played a major role in the growth of the industry, business, health care, research and many more areas. These days, data that is generated is not available merely in structured format, but it is also available in unstructured and semi-structured format. Moreover, data these days is no more static in nature, but it is very dynamic as it is growing exponentially. Therefore, the challenge lies not just in storing Big Data but also in processing Big Data optimally. For data-intensive computational processing, we consider a highly reliable and invariably accepted distributed computing model.

Distributed computing architecture allows scalability in a transparent way by bringing together many computing units via a high-speed network. Distributed

computing enables division of Big Data problems in context of processing, storage and analysis, into smaller sub-problems, so as to solve it in an optimum way (Gilbert and Lynch, 2002). In distributed computing one of the major challenges for managing Big Data is the latency, while measuring the performance parameters such as data access time, transmission time, and throughput etc. It is insignificant to implement Big Data processing platform in a high latency infrastructure, if performance is important. Therefore, it is important to leverage the distributed computing platform to reduce latency. Many researcher and scientists have proposed various tools and techniques, to reduce the latency of distributed computing infrastructure. Many distributed framework such as Hadoop, Spark, Flink, Storm, Samza etc support Big Data processing. Due to less expensive hardware requirement and wide adaptability compared to rest of the models (I2.wp.com, 2018), we chose Apache Hadoop for our research implementation.

Apache Hadoop (Hadoop.apache.org, 2018a) is an open source, scalable, and fault-tolerant framework with the immense potential of large-scale data processing. Hadoop has it all that we expect from the distributed computing environment. Firstly, for distributed file storage Hadoop uses the Hadoop Distributed File System (HDFS). Secondly, for multiple resource management and simultaneous processing on multiple nodes, Hadoop uses Yet Another Resource Negotiator (YARN) and MapReduce programming model respectively. All these three components are helpful in providing fully distributed infrastructure for Big Data processing.

Hadoop distributed file system provides highly scalable, fault-tolerant, reliable and vastly available data storage. HDFS is effortlessly scalable across multiple low-cost commodity hardware machines and maintains replicas to achieve reliability, availability, and fault-tolerance. Hadoop uses HDFS block placement policy for data placement and replica management. YARN handles the resources of the cluster and schedules the Map-Reduce applications for effective utilization of cluster. YARN does that with the Resource Manager and Application Master respectively. YARN provides middle-level service between data storage (i.e. HDFS) and application processing (MapReduce) model. MapReduce support high-level concurrent processing to perform parallel computation across distributed data and

resources. In the MapReduce model, Map phase maps the data blocks from multiple resources and generates key-value pairs. Hereafter key-value pairs are processed in Reduce phase by a shuffle and sorting to produce output and store it on HDFS. In MapReduce job placement and scheduling is important to gain optimum performance. Therefore, node labeling and scheduling plays a vital role in job and task level assignment.

In the case of Hadoop, by implementing Node Labelling, we can actually select the nodes for running specific applications. This feature is important as we can choose nodes based on application characteristics and requirement of resources. At the same time scheduling is equally important for effective use of available resources in Hadoop cluster. Various Hadoop schedulers and their performance comparison (Shah and Padole, 2018) are discussed in detail in section 3 and 4 respectively.

The rest of the sections, we discuss our motivation to carry out this study, the research objective and research methodology that was followed for our research work. Since our approach is based on block rearrangement policy, the reader can find a detailed overview of existing block placement strategy and proposed model in later part of the thesis.

1.1 Motivation

Distributed computing is inevitable to store and process Big Data. Distributed computing allows managing distributed data storage and processing on multiple machines. Hadoop framework provides a distributed computing infrastructure for Big Data storage and processing. Hadoop framework facilitates the distributed storage using the Hadoop Distributed File System (HDFS). HDFS is one of the core components of Hadoop, which is used for efficient distributed computing. HDFS uses block placement policy for placing data blocks (i.e. large files split into blocks of smaller size) on multiple machines.

In HDFS default block placement policy, we do not have control over block placement and cannot choose nodes for storage and processing which may result in

poor load distribution for heterogeneous cluster and may affect MapReduce performance. In MapReduce, load unbalancing may create two major problems: First, if MapReduce containers are running on the datanodes which have low processing capability, then the job will get delayed. Second, If MapReduce containers do not get data blocks where tasks are processing, then blocks will automatically transfer from the nodes of same rack or other racks to the running containers. In both the cases, overall execution time will increase due to high latency and less data locality respectively.

HDFS block placement policy fails to achieve optimum performance specifically for the heterogeneous cluster. In today's era of computing, one cannot envisage having cluster nodes made up of similar configuration called homogeneous cluster. Heterogeneous nodes are having different processing capability and few nodes can be slow due to lower CPU/memory, bad disk, and network congestion. There is a need for better block placement approach in Hadoop which leverages over the heterogeneous cluster and remove these slow nodes from processing to achieve better performance. Hence, Hadoop default HDFS block placement policy needs to be improvised, so as to consider node processing capability or heterogeneity of the system. If processing capability of nodes or heterogeneity of the system is taken into consideration for block placement, then better performance can be achieved, compared to the default policy.

1.2 Research Objective

The prime objective of this research work is to build an ecosystem for big data processing which can work efficiently for the Hadoop heterogeneous cluster. We strive towards making policy to improve Hadoop data locality and load balancing by implementing block rearrangement scheme to optimize the performance of Big Data processing using heterogeneous distributed computing. Our concern is to work on two aspects Hadoop, to improve performance for Big Data processing. Firstly, to improve file management in the HDFS and secondly, to improve process management for Big Data job processing.

In order to optimize Big Data processing time, it is important to optimize Hadoop performance. More specifically, we focus on to optimize the performance of heterogeneous distributed cluster as, Hadoop by default, has limited performance outcome for data-intensive jobs. To accomplish our objective, we do not want to rely on results which are application dependent, but we focus on wide and generalised applicability and adaptability of existing cluster hardware and software configuration. Therefore, simply customizing Hadoop default parameters such as tuning number of reducers, combiners, compression algorithms, and JVM reuse to get better performance is not sufficient. Instead, we focus on, to design an approach which can use existing hardware support, computation capability and effectively use node labelling and scheduling schemes to meet our prime goal.

In order to achieve the objective, we define following sub goals for optimizing Hadoop performance for Big Data processing.

- **Improve data locality:** Data locality refers to the concept, "Moving Computation is Cheaper than Moving Data" (Hadoop.apache.org, 2018f). In order to meet our objective, we find that data locality plays a vital role in optimizing Hadoop performance. We strive towards making a model, which can achieve more data locality than one that works by default.

- **Remove stragglers:** It is important to note that the slower machine slows the entire job (Coursera, 2019). These slow nodes are called as stragglers. Nodes can be slow due to bad disk, network bandwidth, and slower CPU / memory. The performance of Hadoop also effects due to stragglers. Therefore, we present an approach to remove straggler from MapReduce processing.

- **Load balancing:** In order to achieve load balancing, it is important to tweak default HDFS block placement policy. Default HDFS block placement policy does not consider optimization of the heterogeneous cluster. Therefore, there is a need for better block placement policy, which can leverage the heterogeneity of nodes. We propose to design a block rearrangement scheme which can evenly balance the load amongst heterogeneous nodes.

- **Application transparent:** In case of data-intensive applications, it is important that the applied optimization technique should be adaptable to the application in consideration. It is always preferable that users take the advantage of newly designed optimization technique without rewrite or recompile the application. We try towards making our optimization model such a way that it should be application independent and leverages the existing application for wide applicability.

- **System independent:** To achieve better performance in the heterogeneous cluster, it is important to design an optimization model which is system independent. Therefore, we strive towards making policy which does not change with the system and easily able to scale the optimization model as and when required.

- **Less client involvement:** We firmly believe that Hadoop system tuning to achieve better performance require a lot of efforts and in-depth knowledge of Hadoop cluster and configuration. In order to meet our goal, we propose a model which requires minimal client involvement, to make a decisions regarding node selection for processing, based on the available hardware capability of the cluster.

1.3 Research Methodology

To carry out this research, we followed an empirical approach. The analytical or mathematical model for optimization did not suit our approach. Our approach was based on improving the performance of Hadoop heterogeneous cluster for Big Data processing. Therefore, the only reliable performance measure was the practical implementation of our research problem. First, we conducted an in-depth research study about past work done in the areas of distributed scheduling algorithms, distributed filesystem, Hadoop schedulers, and performance optimization approaches in Hadoop. The outcome of this literature study helped us to establish a Hadoop heterogeneous cluster. It also gave us insight into performance bottlenecks in Hadoop and various strategies already established by researchers and scientists.

Once we studied HDFS block placement policy and performance issues in the heterogeneous cluster, we found an opportunity to design a block rearrangement policy which can leverage the heterogeneous cluster, without compromising with the performance of Hadoop. Next, as proposed, we implemented "Saksham" – a block rearrangement policy to eliminate the limitations of the default policy. While trying to get better data locality, we used, node labelling and scheduling along with the proposed "Saksham" algorithm, for block rearrangement. Finally, we tested the "Saksham" model by comparing the results with default Hadoop and default node labelling approach and saw significant improvement in Hadoop performance. We used data locality, job completion time, and latency as our performance measuring parameters, for data-intensive applications.

For this research experiments, we implemented Hadoop cluster on commodity hardware as well as on highly configured system environment. We chose Hadoop benchmark applications for our experiment and publicly available datasets. We used Grid'5000 large-scale distributed infrastructure for our experiments.

1.4 Thesis Organization

The rest of the thesis is organized as follows.

Chapter-2 gives an overview of Big Data. It also discusses distributed computing and heterogeneous distributed computing. Distributed computing performance highly relies on a distributed file system (DFS) and scheduling algorithms. Therefore, we discuss it in detail in this chapter.

Chapter-3 gives in-depth information about Hadoop. First, we introduce Hadoop and three core components of its architecture. Second, we also describe an overview of two important aspects of model i.e. node label and scheduler. Finally, we discuss some of the challenges of Hadoop on which researchers can target upon.

Chapter-4 presents the literature review. This section briefs about various distributed scheduling algorithms, different types of distributed file systems, practical analysis of scheduling algorithms and various performance improvement strategies in Hadoop.

Chapter-5 discusses about our research contribution and proposed algorithm called 'Saksham' for defining the block placement policy in Heterogeneous Cluster. The chapter also discusses, as proposed, the node labelling concept that considers the storage and processing capability of the node, before placing the file blocks in the given node. This helps in optimally scheduling the task on the given node, based on the availability of the data and the processing capability of the node.

Chapter-6 summarizes the test results of two Hadoop benchmark applications i.e. WordCount and TeraSort. We demonstrate the block rearrangement policy and present how we leverage node processing capability for enhancing the performance of Hadoop. Our results show that our proposed model can achieve high data locality, less job completion time, and load balancing while incurring negligible rearrangement time overhead.

Chapter-7 concludes the research carried out and discusses the future work.