Improving Performance of Map Reduce using DLAJS Algorithm

Balaji Siva Jyothi[#1], Dr. P. Radhika Raju[#2], Dr.A.Ananda Rao[#3]
[1]*M.Tech Scholar,* [2]Ad-hoc Assistant Professor, [3]Professor *Department of CSE, JNTUACEA, Ananthapuramu, A.P, India*

Abstract

Cloud Computing provides different services to the users with regard to processing data. The main concepts in cloud computing are big data and big data analysis. Hadoop framework is used to process big data in parallel processing mode. Job scheduling and optimized resource allocation can help improve performance of Hadoop. In the existing system Hadoop architecture has been enhanced in order to reduce computational complexity while processing big data. It also takes care of efficient resource allocation and processing textual data such as DNA sequence. Their architecture was named as H2Hadoop that improves the ability of NameNode to assign jobs to the TaskTrackers (DataNodes) in a given cluster. By adding control features to NameNode, their architecture can intelligently assign tasks to the DataNodes where required data is present thus reducing resource utilization pertaining to CPU time, number of read operations etc. However, the existing system can be improved to have more focused approach by considering data locality awareness to the job scheduling process. In the proposed system, an algorithm is proposed to have data locality aware job scheduling. This algorithm is named as Data Locality Aware Job Scheduling (DLAJS) algorithm. The algorithm explores the data locality aware to know how far efficient job scheduling. Thus, consuming less cloud resources such as CPU, memory and execution time.

Keywords — *Cloud computing, Big data, Hadoop, MapReduce framework,Data-locality,Job scheduling*

I. INTRODUCTION

The exponential growth of data led to big data of late which in turn demanded the distributed programming frameworks like Hadoop to process voluminous data in short span of time. Data intensive applications can be run in distributed, scalable and parallel-processing environment. Hadoop is the framework that supports MapReduce programming paradigm. Big data needs Hadoop and MapReduce environment and the problem with Hadoop is that it has provision for assigning mappers based on the data availability and data locality. When data locality is known, the Hadoop framework assigns mappers accordingly so as to ensure faster processing of data thus reducing network overhead.

The problem identified with MapReduce framework is that Hadoop does not consider data locality in the case of assigning reducers to worker nodes in Hadoop clusters. This will result in delay in processing, increased latency and decreased throughput. To overcome this problem, data locality aware approach in assigning mappers is explored in this paper. The existing approaches can be improved further with practical implementation. In this paper is detailed the issue in Section 4 and proposed an algorithm to solve it. The contributions in this paper are as follows.

The proposed algorithm named Data Locality Aware Job Scheduling (DLAJS) for assigning reducers to worker nodes based on the data locality. This will reduce computational complexity besides reducing network overhead. It reduces latency and increases throughput. The prototype application is built to demonstrate proof of the concept. Experiments are made with CloudEra test bed that supports Hadoop and MapReduce frameworks. Experimental results revealed the significance of data-locality aware approach in MapReduce framework.

The remainder of the paper is structured as follows. Section 2 reviews related literature. Section 3 provides the need for big data and the MapReduce programming framework associated with Hadoop. Section 4 formulates the problem addressed in this paper. Section 5 provides the proposed solution to the problem with an underlying algorithm. Section 6 provides experimental results while section 7 concludes the paper and gives directions for future work.

II. RELATED WORKS

This section provides review of literature on MapReduce programming paradigm and the improvements that can be made. The problem of big data and how the problem is solved with the introduction of distributed programming frameworks is explored in [1]. Exploiting Meta data of related jobs and improving performance of MapReduce framework is the focus in [2]. There are many performance models related to Hadoop as studied in [3]. Optimization of Hadoop with possible data import is examined in [4], [5]. Locality aware resource allocation so as to optimize resource usage is the main research carried out in [6], [7]. The concept of mobile networks and the utility of them in the context of distributed programming are explored in [8]. Storage of data, analysis of data and other issues related to data in the context of high performance computing are analyzed in [9], [10]. Hadoop kind of distributed programming frameworks need the cloud eco system to function well [11].

Pervasive computing environments need big data processing [12]. Mining big data associated with mobile phones can be done with a probabilistic approach [13]. Technical issues and challenges associated with big data are explored in [14], [15]. Large scale data management and analysis of data with distributed computational solutions is found in [16]. Parallel process of algorithms with Hadoop platform is the main focus in [17]. In the presence of big data processing and MapReduce programming paradigm, query optimization and parallel processing of massive amounts of data is studied in [18]. In this paper the proposed algorithm to solve the problem of data-locality aware assignment of reducers to worker nodes to improve Hadoop performance.

III. PRELIMINARIES

This section provides important information that leads to understanding the proposed work in this paper. It includes big data and Hadoop MapReduce.

A. Big Data and Need for It

Big data, as the name implies, is voluminous data (V). There are other Vs associated with it. They include variety, value and velocity. Volume indicates that the data is very huge and cannot be accommodated in local machines generally. It is measured in peta bytes. Velocity is another attribute that informs that the big data keeps growing continuously (streaming data). Variety attribute on the other hand informs us that bit data is in many forms. They are known as structured format, unstructured format and semi-structured format. When sources of input are from different places or branches of a company, the data needs to be processed as a whole.

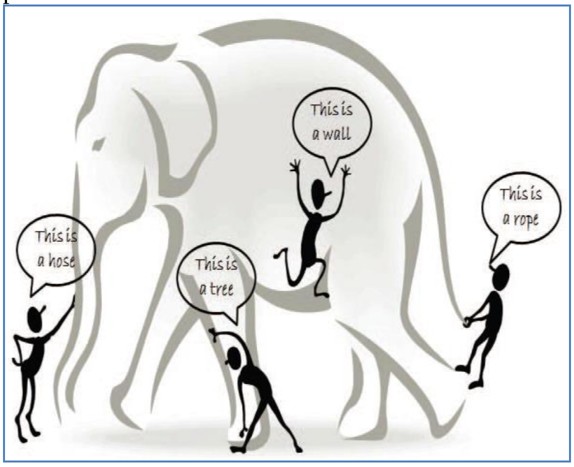

Figure 1: Shows the importance of considering big data for gaining unbiased conclusions

As shown in Figure 1, it is evident that big data has to be considered for gaining complete business intelligence. Processing some part of data provides biased conclusions. Unbiased conclusions can be obtained by considering big data. The rationale behind this is that the big data contains complete data that can provide comprehensive intelligence when mined. So as to process such gigantic measure of information and even to store it, distributed computing foundation alongside disseminated programming structures like Hadoop are required.

B. Hadoop's MapReduce Framework

Hadoop is one of the distributed programming frameworks that support big data storage and processing. With its associated Hadoop Distributed File System (HDFS), it can store and handle big data. Hadoop supports MapReduce programming approach that is new. It can take voluminous data as input and split it in the form of subsets of data containing key/value pairs. Keys are uniquely identified while the values may have duplicates.

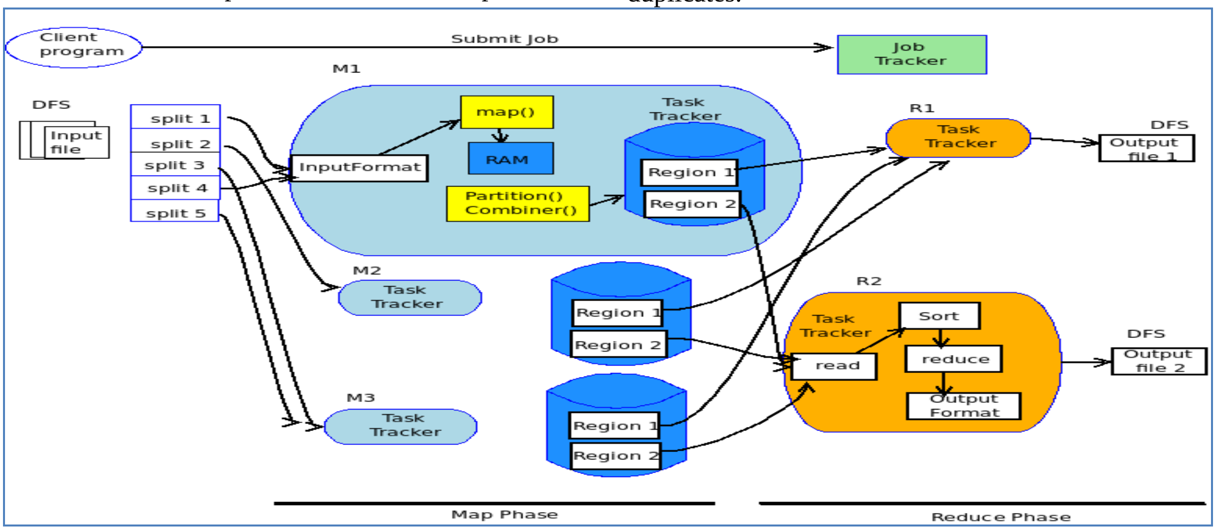

Figure 2: MapReduce framework of Hadoop

As presented in Figure 2, it is evident that the Map phase and Reduce phase are part of the framework as the name implies. Map phase performs actually intended job. It does processing before that it takes data from the HDFS in the form of chunks of data. In other words input data from HDFS is taken by the framework and it is split into number of parts and given to many commodity computers where map task runs. Map task takes care of processing given data and the intermediate results are given to Reduce phase in the form of key/value pairs. The Reduce phase takes care of producing final output. In this context the problem definition is provided in the section 4.

IV. PROBLEM FORMULATION

Hadoop supports MapReduce programming paradigm. According to this jobs are assigned by Job Tracker to Task Tracker. In the process, the whole input data is split into number of chunks. Then each chunk of data is given to a mapper (worker node in the Hadoop cluster). Mapping takes care of intended functionality. However, one mapper cannot produce the whole output. The intermediate results of all mappers are to be properly clubbed and output needs to be produced.

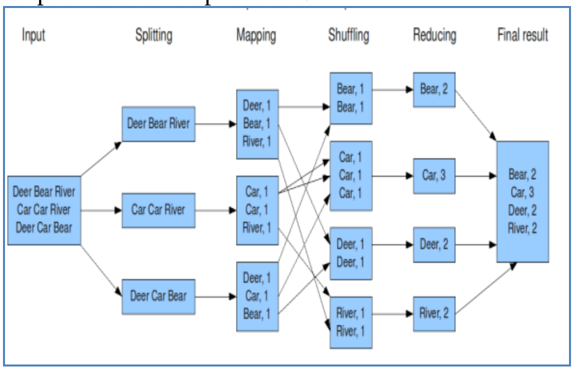

Figure 3: Illustrates workflow of MapReduce with WordCount benchmark

As presented in Figure 3, it is evident that the map phase is able to count the occurrence of words in the given chunk of data. The result of mapping is given for shuffling.

shuffling phase sorts data in ascending order. Then the reduce phase is making the summary of count of words. Afterwards, the final result is produced. In this context, the problem is that Hadoop assigns map tasks to nearly worker nodes based on data locality. However, Hadoop framework does not consider data locality while assigning reduce tasks to worker nodes. This can lead to issues related to performance. Therefore data locality-aware scheduling of jobs to Reduce worker nodes provides significant performance benefits. The proposed system to achieve is explored in section 5.

V. PROPOSED SYSTEM

In the proposed system, a new algorithm is proposed to have data locality aware job scheduling. This algorithm is named as Data Locality Aware Job Scheduling (DLAJS) algorithm. The calculation abuses the information territory mindful skill for effective employment planning in this way devouring less cloud assets, for example, CPU, memory and execution time. A model application is worked to show verification of the idea. The proposed solution for enhancing execution of MapReduce is given in Figure 4.

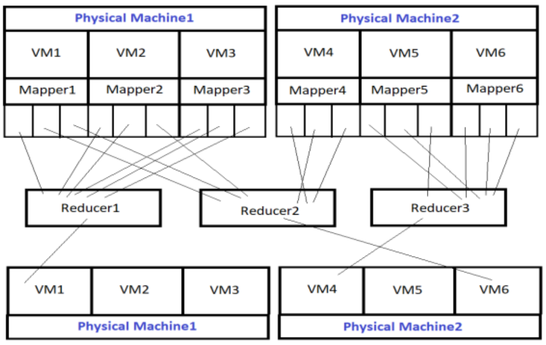

Figure 4: Architectural overview of the proposed system

Data locality is measured by the total amount of data stored locally on the physical machine for each virtual machine.When data is local, it can be accessed faster. Data locality aware scheduling is therefore a reflection of performance of MapReduce programming paradigm.

A. Data Locality Aware Job Scheduling Algorithm

This algorithm takes care of data locality while assigning reduce tasks in Hadoop distributed framework. Thus it can bring about performance by reducing network overhead.

Algorithm: Data Locality Aware Job Scheduling
Input: Set of physical machines PM, set of virtual machines VM, reducer index i, partition index j
Output: Data locality aware mapping of reducers

1. R=getAllReducers()
2. For each pm on PM
3. For each mapper on pm
4. For i=1 to NoOfReducers
5. For j=1 to NoOfPartitions
6. Compute reducer i's partition size
7. End For
8. End For
9. End For
10. End For
11. //Assign reducers to physical machine based on data locality
12. For each pm from PM
13. VM = virtual machines of pm
14. R = reducers of pm
15. Sort VM based on speed
16. //best fit reducer assignment
17. For i=1 to number of reducers in PM
18. VM[i]=reducers[i]
19. End For
20. End For

When the algorithm is applied to the case shown in Figure 1, it does like this. Each physical machine has 3 virtual machines. Each VM has a mapper associated. Each mapper has 3 data partitions. Based on the data locality and the speed of VM, the reducers are assigned to physical machines appropriately.

VI. EXPERIMENTAL RESULTS

Experiments are made with CloudEra which is one of the test beds for making experiments with Hadoop framework. It runs in virtual environment using Oracle VM Virtual Box or VMware. Observations are made in terms of number of read operations with given benchmark application.

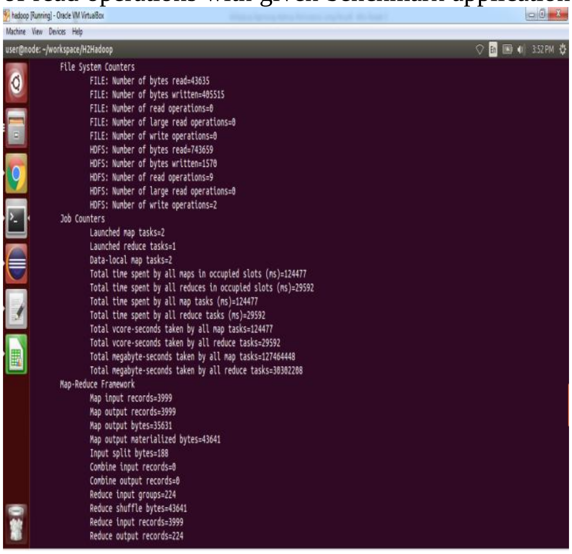

Figure 5: Shows the initiation of jobs with MapReduce framework

As shown in Figure 5, it is evident that the MapReduce framework with CloudEra environment is loaded and it performs its intended operations. It shows the MapReduce programming paradigm and its dynamics and statistics in the console.

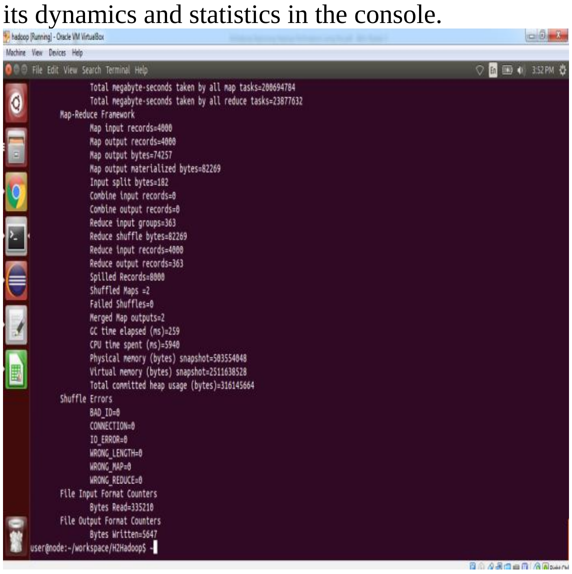

Figure 6: Shows the initiation of jobs with MapReduce framework along with I/O and shuffle errors

As shown in Figure 6, it is clear that the MapReduce structure with CloudEra condition is stacked and it plays out its planned tasks. It shows the MapReduce programming paradigm and its dynamics and statistics in the console with details like shuffle phase parameters and file input and output details besides shuffle errors if any.

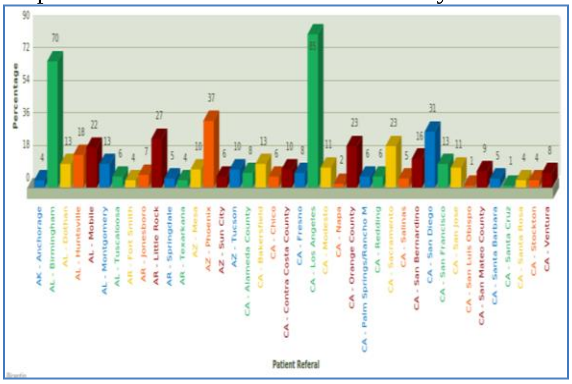

Figure 7: Showing results of patient referral dynamics with healthcare benchmark

As presented in Figure 7, it is evident that the MapReduce programming produced percentage of patient referral on different healthcare units as part of healthcare benchmark application that is run with Hadoop. In this process, there are other observations that are related to data locality-aware job scheduling. The results are as follows.

Table 1: Number of read operations to show performance difference

Common Feature	HDFS: Number of Read Operations		
	Native Hadoop	***H2Hadoop***	***Proposed***
Sq1	100	100	80
Sq2	100	15	10
Sq3	100	68	50
Sq4	100	41	30
Sq5	100	16	10

As presented in Table 1, it is evident that the proposed system and existing systems are presented in terms of number of read operations against five common features.

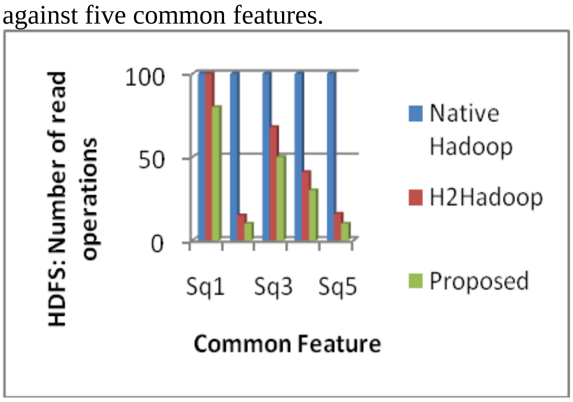

Figure 7: Performance comparison in terms of number of read operations

As shown in Figure 7, , it is clear that the basic highlights are exhibited in flat pivot while the vertical hub demonstrates the quantity of read tasks. The proposed framework beat the current frameworks. Both H2Hadoop and the proposed algorithm demonstrated preferred execution over local Hadoop because of the thought of information territory mindful occupation planning.

Table II: CPU time in seconds to show performance difference

Common Feature	CPU Time in Seconds		
	Native Hadoop	***H2Hadoop***	***Proposed***
Sq1	370	385	360
Sq2	397	50	40
Sq3	390	270	250
Sq4	392	149	130
Sq5	410	61	50

As presented in Table 2, it is evident that the proposed system and existing systems are presented in terms of CPU time in seconds against five common features.

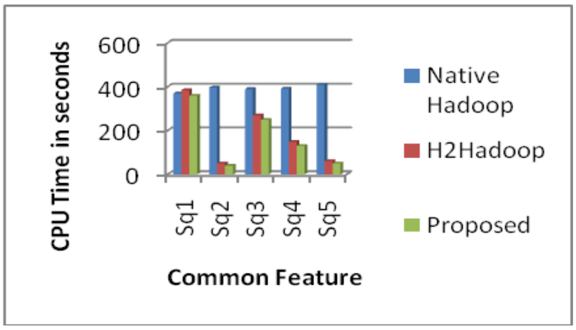

Figure 8: Performance comparison in terms of CPU time

As shown in Figure 8, , it is clear that the regular highlights are displayed in flat hub while the vertical hub demonstrates the CPU time right away. The proposed framework beat the current frameworks. Both H2Hadoop and the proposed algorithm demonstrated preferable execution over local Hadoop because of the thought of information territory mindful employment planning.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, the proposed algorithm named Data Locality Aware Job Scheduling (DLAJS) in order to solve the problem of assigning reducers with data locality-aware job scheduling. The problem with native Hadoop framework is that it takes care of the data locality-aware approach in assigning map tasks to worker nodes. However, while assigning reducers, it does not consider data locality aware approach. This has resulted in reduction of throughout, increase in latency and increase in overall network overhead. Hadoop clusters contain thousands of commodity computers. Appropriate allocation of the computing resources can lead to performance gain. In this manner, it is fundamental to enhance execution of Hadooop as it will have huge impact on both service providers and service consumers. The implemented proposed algorithm using CloudEra test bed. The experimental results revealed significance improvement in the performance of Hadoop.

In future the work intend to have an architectural modeling framework to explore other possibilities in optimizing Hadoop performance.

REFERENCES

[1] Patel, A.B., M. Birla, and U. Nair. Addressing big data problem using Hadoop and Map Reduce. in Engineering (NUiCONE), 2012 Nirma University International Conference on. 2012.

[2] HamoudAlshammari, Jeongkyu Lee and Hassan Bajwa. (2016). H2Hadoop: Improving Hadoop Performance using the Metadata of Related Jobs. IEEE TRANSACTIONS ON Cloud Computing, p1-11.

[3] Herodotou, H., Hadoop performance models. arXiv preprint arXiv:1106.0940, 2011.

[4] Xu, W., W. Luo, and N. Woodward. Analysis and optimization of data import with Hadoop. IEEE.

[5] P.Radhika Raju, Dr. A.Ananda Rao, Optimization of program invariants, ACM SIGSOFT Software Engineering Notess, Vol.39, Issue 1, January 2014.

[6] Palanisamy, B., et al. Purlieus: locality-aware resource allocation for MapReduce in a cloud. in Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. ACM.

[7] Hammoud, M. and M.F. Sakr. Locality-Aware Reduce Task Scheduling for MapReduce. in Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on. 2011.

[8] Chen, M., S. Mao, and Y. Liu, Big Data: A Survey. Mobile Networks and Applications, 2014. 19(2): p. 171-209.

[9] Buck, J.B., et al. SciHadoop: Array-based query processing in Hadoop. in High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for. 2011.

[10] Condie, T., et al.MapReduce Online.in NSDI.2010

[11] Schatz, M.C., B. Langmead, and S.L. Salzberg, Cloud computing and the DNA data race. Nature biotechnology, 2010. 28(7): p. 691.

[12] Changqing, J., et al. Big Data Processing in Cloud Computing Environments. in Pervasive Systems, Algorithms and Networks (ISPAN), 2012 12th International Symposium on. 2012.

[13] Farrahi, K. and D. Gatica-Perez, A probabilistic approach to mining mobile phone data sequences. Personal Ubiquitous Comput., 2014. 18(1): p. 223-238.

[14] Jagadish, H., et al., Big data and its technical challenges. Communications of the ACM, 2014. 57(7): p. 86-94.

[15] Marx, V., Biology: The big challenges of big data. Nature, 2013. 498(7453): p. 255-260.

[16] Schadt, E.E., et al., Computational solutions to large-scale data management and analysis. Nature Reviews Genetics, 2010. 11(9): p. 647-657.

[17] Ming, M., G. Jing, and C. Jun-jie. Blast-Parallel: The parallelizing implementation of sequence alignment algorithms based on Hadoop platform. in Biomedical Engineering and Informatics (BMEI), 2013 6th International Conference on. 2013.

[18] Wu, S., et al. Query optimization for massively parallel data processing. in Proceedings of the 2nd ACM Symposium on Cloud Computing. 2011. ACM.