International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 3 | ISSN: 2456-3307

Hadoop Periodic Jobs Using Data Blocks to Achieve Efficiency

Sujit Roy[1], Subrata Kumar Das*[2], Indrani Mandal[3]
Department of Computer Science and Engineering, Jatiya Kabi Kazi Nazrul Islam University, Trishal, Mymensingh, Bangladesh

ABSTRACT

To manage, process, and analyze very large datasets, HADOOP has been a powerful, fault-tolerant platform. HADOOPis used to access big data because it is effective, scalable and is well supported by large trafficker and user communities. This research paper proposed a new approach to process the data in HADOOP to achieve the efficiency of data processing by using synchronous data transmission, sending block of data from source to destination. Here a method has been shown how to divide the data blocks in achieving optimal efficacy by adjusting the split size or using appropriate size of staffs. As the effective HADOOP hardware configuration matches the requirements of each periodic task, so this allows our system to the data blocks increasing data efficiency as well as throughput.

Finally, experiments showed the effectiveness of these methods with high data efficiency (around 22% more than existing system), low installation cost and the feasibility of this method.

Keywords: Hadoop, Map Reduce, Data efficiency, Data blocks, HDFS.

I. INTRODUCTION

HADOOP [7] is the popular open source implementation of map reduce, a powerful tool designed for deep analysis and transformation of very large data sets. It is capable to connect and coordinate thousands of nodes inside a cluster. The HADOOP framework has two components such as HADOOP Distributed file system (HDFS) [6] and Map Reduce [8]. HDFS is designed for data storage and Map Reduce for data processing. The HDFS layer provides a virtual file system to client application. An important feature of HDFS is data replication and each file is divided into blocks and replicated among nodes. At first, all data are stored in HDFS. From the point of data efficiency, this essay proposed a method to measure the effectiveness of periodic tasks that run on HADOOP platform, and it also studied how to divide the data blocks , and it servers in the cluster contains data from different blocks, the analysis of measurement methods.

The step to study the energy efficiency of the cloud is to have evaluation criteria. The main task of this research paper, to manage how data file are divided and stored across the clusters by using HADOOP distributed file system. Data is divided into blocks, and individual server in the cluster contains data from different blocks. The HADOOP is a distributed framework that makes it easier to process large data sets that reside in clusters of computers. Because it is a framework, HADOOP is not a single technology or product. It can work with any distributed file system. However the HDFS is the primary means for doing so and is the heart of HADOOP technology. HDFS manages how data files are divided and stored across the cluster. Data blocks are divided into, each server in the cluster contains several data from different blocks.

Map Reduce

With the Map Reduce programming model, programmers only need to specify two functions:

Map and Reduce. Map whichtakes an input pair and creates a set of intermediate key/value pairs. The library groups of Map Reduce together all intermediate values associated with the same intermediate key I and passes them to the Reduce function. The Reduce function accepts an intermediate key I and a set of values of that key. It aggregates together those values to form a possibly smaller set of values.

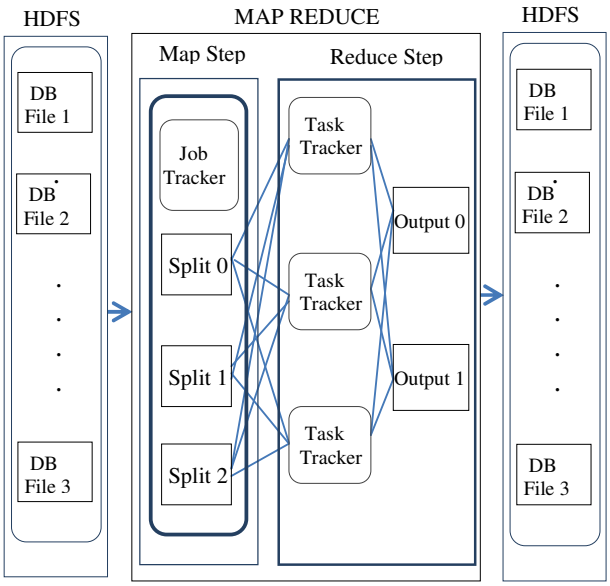

Figure 1. HDFS and Map Reduce Flow

In Figure 1 the map step has two nodes one master node and other worker nodes. The master node splits the data and assigns the key value pairs. The master node picks the idle workers and assigns each one a job to set the intermediate key value pairs and send back to master node. The master node keeps the details information about the location of the data. The Reduce function receives the intermediate key value pairs and reduces to a smaller solution.

The remainder of the paper is structured in the following manner. Section 2 presents the related work in this area and shows the contribution of this work. In section 3, we describe the proposed map reduce architecture. Section 4 depicts the methods of measuring efficiency of the systems. Section 4 presents our experimental results and discussion and section 5 concludes this paper.

II. RELATED WORK

Now, we present some related research work. One of the biggest challenges in distributed systems like HADOOP is to guarantee the data stability even if some nodes fail. Thus, it is important to maintain redundancy for stored data in HADOOP system. A more interesting research, by Leverichet al. [9] who develop a Mechanism named “covering subset”. Thus, this technique is inefficient.

The focus of this paper, the constellation management system is similar in essence to other systems such as Condor [10]. The Map Reduce implementation relies on data cluster management system that is responsible for distributing and running user tasks on a large collection of shared mechanisms.

After promoting the concept of cloud computing in 2006, it has developed rapidly and many companies have started their own cloud computing platform, such as Azure from Microsoft, IBM Blue Cloud Computing Platform from IBM, Google Compute Engine from Google, and Dynamo from Amazon [1]. Though the technology of cloud computing develops continuously, HADOOP platform is the most popular object which majority of scholars study. HADOOP computing platform is composed of Map Reduce computing framework and HDFS file storage system, and current research about HADOOP [2, 3] focuses on performance optimization but it’s processing efficiency is not high for small files [4, 5].

III. PROPOSED MAPREDUCE ARCHITECTURE

Map Reduce allows for distributed processing of the map and reduction operations. In mapping operation, all maps can be performed in parallel. Similarly, a set of 'reducers' can perform the reduction phase, which provided that all outputs of the map operation that share the same key are presented to the same reducer at the same time, or that the reduction function is associative.

In this model, a user specifies the computation by two functions Map and Reduce. In the mapping phase, Map Reduce tasks the input data and feeds each data element to the mapper. In the Reducing phase, the reducer processes all the outputs from the mapper and arrives at a final result.

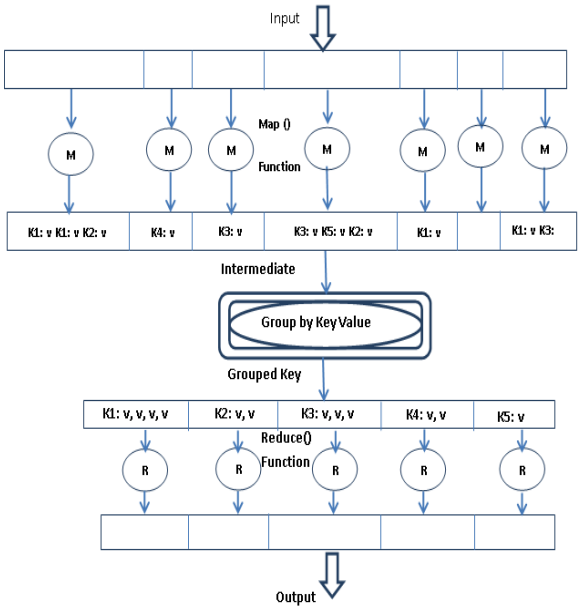

Figure 2. Proposed Map Reduce Architecture.

All the input data associated with the key valueis shown in figure 2. The data will be grouped for intermediate by using 'key value'. Reduce function can be used several key value. Finally we can show reducing output file. The Map function is applied in parallel to every pair (keyed by k1) in the input dataset. This produces a list of pairs (keyed by k2) for each call. After that, the Map Reduce framework collects all pairs with the same key (k2) from all lists and groups them together, creating one group for each key. Thus the Map Reduce framework transforms a list of (key, value) pairs into a list of values. A list of arbitrary values and returns one single value that combine all the values returned by Map.

The Map Reduce programming model divides a program into tasks, of which there are two types: map () and reduce (). Both the two functions are defined as per data structured in <key, value> pairs. The Map function is moved in parallel to take one pair of data with a type in the input dataset and then produces a list of intermediate pairs for calling Reduce.

$(k1, v1) \rightarrow list(k2, v2);$
$Red(k2, list(v2)) \rightarrow list(v3)$

Block Size is another important factor of the implementation of HADOOP, which also makes effect on data efficiency for Map Reduce. For Map Reduce, HADOOP divides the input to a Map Reduce job into fixed-size pieces (usually equals block size) called input splits. HADOOP makes one map task for each split.

IV. METHODS

A. Proposed Data Transmission Methodology

In the proposed method, the data blocks are divided to get maximum data transmission efficiency. In this method at first the target data are stored in a primary storage device. To transmit the data, the format of data container in Figure 3 will take 80-132 characters in size. Trailer and header are also connected to the data field. Both the trailer and header contain an 8 bits flag and control field. The data are divided into block (packet or frame) and transmit one block each time. There are block interval between two blocks. Every block data contain 2 bytes header information at the front and 2 bytes trailer information at the end.

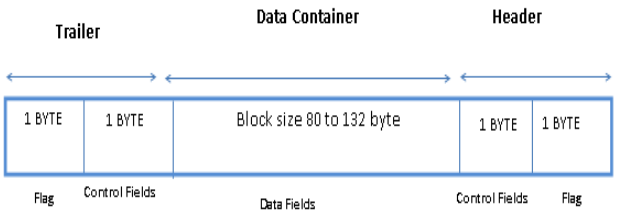

Figure 3. The format of data transmission block

B. Algorithm of Proposed Approach

Step1. The input reader reads data from stable storage (typically a distributed file system) and generates key/value pairs.

Step2. Split the input file (0……..N)

Step3. The Map () is used to the split file

(split 0 ……..split N)

Step4. Then the split files are shuffling according to their file category.

Step5. Apply Reduce () to optimize it. The Reduce can iterate through the values that are associated with that key and produce zero or more outputs.

Step6. Finally analyze the files efficiency by using the formula.

Data Transmission efficiency=$\frac{Real\ Data}{Total\ Data} \times 100\ \%$

Here, Total Data=Real Data + Overhead data

C. Mathematical Analysis

Now we assume the Throughput of the data file. Throughput is the ratio of the file size and time.

Throughput (N) = $\frac{\sum_{i=0}^{N} Filesize_i}{\sum_{i=0}^{N} Time_i}$ Mb/sec [File Size (MB)/Transfer Speed (kbps) = Time(s)]

Average IO Rate (N) = $\frac{\sum_{i=0}^{N} Rate_i}{N}$

To analyze the efficiency by using the formula

Data Efficiency (μ) =$\frac{R_d}{T_d} \times 100\%$ $T_{d=R_d+O_d}$

Overhead Data O_d= $\left| \frac{Real\ Data}{Block\ size} \right| \times 32$

Here, $T_d = Total\ Data,\ \ R_d = Real\ Data\ and\ \ O_d = Overhead\ Data$

Data Transfer Speed (kbps) = $\frac{\text{File Size (MB)}}{\text{Time (s).}}$

V. RESULTS AND DISCUSSION

In this research various size of data are used to process in existing and proposed methods to get a comparison between the systems. The outcomes of the proposed system showed a better efficiency than the existing methods.

The different size of data below 100 MB are processed in existing system and new system, which are tabulated in Table 1.

Table 1. Different stored file from 0mb to 100 MB and corresponding efficiency

Memory Storage Space(MB)	Data Transmission Efficiency (%)	
	Existing Method	Proposed Method
5	62.5	13.5
10	62.5	23.8
15	71.0	31.9
20	68.9	38.5
25	67.5	43.9
30	71.4	48.4
35	70.0	52.2
40	72.7	55.5
45	71.4	58.4
50	70.4	60.9
55	72.3	63.2
60	71.4	65.2
65	70.6	67.0
70	72.1	68.6
75	71.4	70.1
80	72.7	71.4
85	72.0	72.6
90	71.4	73.7
95	72.5	74.8
100	71.9	75.7

The data transmission efficiency listed in this table express that if the data size is less the efficiency of the existing system is higher than the proposed system. But this efficiency trend skipped the previous method when the amount of data passed over 80MB is shown in Figure 4.

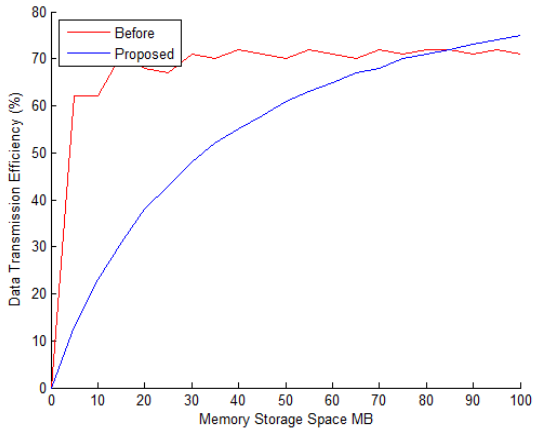

Figure 4. The efficiency trend found between 0 MB 100 MB

The proposed system showed a better efficiency when the size of data become more than 100 MB tabulated in Table 2. Here it is followed that the efficiency between both systems showed a big difference. The highest efficiency of the existing system was around 73% which is about 22% less than the proposed system with 95%. It is noticeable that the efficacy of the both methods became saturated after a certain amount of data size depicted in Figure 5.

Table 2. Different stored file over 100 MB and corresponding efficiency

Memory Storage Space (MB)	Data Efficiency (%)	
	Existing Method	Proposed Method
512	72.72	94.11
1022	72.68	94.10
1535	72.71	94.11
2372	72.69	94.88
2550	72.71	95.22
50555	72.725	95.237
61000	72.727	95.205
75013	72.726	95.207
99236	72.726	95.210
152345	72.7263	95.219
250123	72.7268	95.235

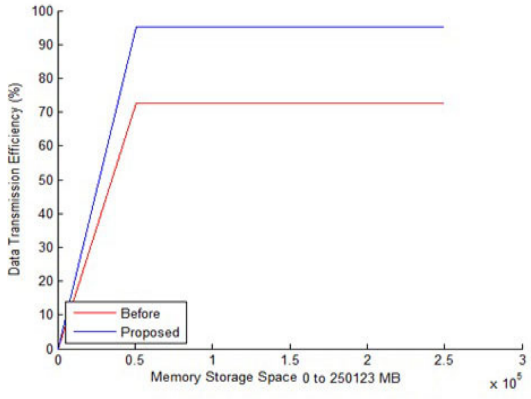

Figure 5. Comparison graph of the efficiency found from both systems

VI. CONCLUSION AND FUTURE WORK

This research focuses on to measure the better data efficiency of periodic tasks that run on HADOOP platforms. It also studied how to divide the data blocks efficiently to distribute and manage the data. Data block distribution according to the proposed architecture in HADOOP achieved around 22% more efficiency than the other existing systems. Next we plan to work on Bandwidth in HADOOP clustering. As the HADOOP contains different categories of file extension, so it would be possible to increase data transmission rate by grouping files before execution.

VII. REFERENCES

[1]. J. Dean, and S. Ghemawat, "Map Reduce: Simplified Data Processing on Large Clusters," In: OSDI 2004: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation, San Francisco, CA, pp. 137-149, 2004.

[2]. J. Pan, Y. L. Biannic, and F. Magoulès, "Parallelizing Multiple Group-by Query in Share-Nothing Environment: a Map Reduce Study Case," in Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, Chicago, Illinois, pp. 856-863, 2010.

[3]. S. Chen, and S. Schlosser, "S. W.:Map-Reduce Meets Wider Varieties of Applications," IRP-TR-08-05, Technical Report, Intel Research Pittsburgh, 2008.

[4]. S. Leo, and G. Zanetti, "Pydoop: a Python Map Reduce and HDFS API for HADOOP," in Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, Chicago, Illinois, pp. 819-825, 2010.

[5]. D. Huang, X. Shi, and S. Ibrahim et al., "MR-Scope: a Real-Time Tracing Tool for MapReduce," in Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, Chicago, Illinois, pp. 849-855, 2010.

[6]. P. Kumar, and V. S. Rathore, "Efficient Capabilities of Processing of Big Data using Hadoop Map Reduce," International Journal of Advanced Research in Computer and Communication Engineering, vol. 3, issue 6, pp. 7123-7126, 2014.

[7]. J. Liu, F. Liu, and N. Ansari, "Monitoring and analyzing big traffic data of a large-scale cellular network with Hadoop," Network, vol. 28, issue 4, pp. 32-39, 2014.

[8]. C. Chen, Z. Liu, W. Lin, S. Li, and K. Wang, "Distributed Modeling in a MapReduce Framework for Data-Driven Traffic Flow Forecasting," Intelligent Transportation Systems, vol. 14, issue 1, pp. 22–33, 2013.

[9]. J. Leverich and C. Kozyrakis, "On the energy (in) efficiency of hadoop clusters," ACM SIGOPS Operating Systems Rev., vol. 44, issue 1, pp. 61–65, 2010.

[10]. D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: The Condor experience," Concurrency and Computation: Practice and Experience, vol. 17, issue 2-4, pp. 323-356, 2005.