ISSN 2349-6193
Impact Factor (PIF): 2.243

IJESMR

International Journal OF Engineering Sciences & Management Research

IMPLEMENTATION OF COMBINER FOR EFFICIENT BIG DATA PROCESSING IN HADOOP MAPREDUCE FRAMEWORK

Mr. Harshawardhan S. Bhosale
Department of Computer Engineering, JSPM's Imperial College of Engineering & Research, Wagholi, Pune, India

Keywords: Big Data, Hadoop Framework, Online Aggregation, Combiners

ABSTRACT
Big Data is a data that cannot be processed or analyzed by using traditional systems such as relational databases and data warehouses. The Big Data can be structured, unstructured or semi-structured. An open source, large data processing framework called Hadoop is been used to process large amounts of data. However, processing jobs in Hadoop is time-consuming. In this paper I have proposed a Hadoop Online Aggregation Technique to decrease response time of Hadoop by executing the job partially. In Online Aggregation of Hadoop MapReduce an early result are made available to the user before job completion. In online aggregation, Hadoop system processes a query in an online fashion. The processing of Hadoop is also benefited by adding the combiners in MapReduce paradigm. The proposed technique can reduce the total computing time taken by the Hadoop MapReduce framework.

INTRODUCTION
Recent days the cost of data storage and processing have motivated all kinds of organizations to store and analyze huge amounts of data. Big data differs from traditional data in three characteristics, which are Volume, Variety and Velocity.

- ***Volume:*** The amount of data that is being collected today is incomparable to that of few years back. Data does not fit in memory. And, therefore there is a need of partitioning and distributing the data in several places.
- ***Variety:*** Data is being collected from different sources, such as web, user interactions, social media, sensors and review sites. The data collected from these sources is semi-structured or unstructured. The data being collected is present in variety of forms.
- ***Velocity:*** The rate at which data is being generated is fast. For example, it is reported that 100 hours video data are being uploaded to YouTube every minute.

Large-scale data management and analysis have emerged as one of the main research challenges in computing. Modern large-scale data processing systems can processes web data, transaction and content-delivery logs, and scientific and business data [4] [12]. Nowadays, parallelism is also used in modern data processing systems. However, in a highly parallel setting, analyzing big data takes a considerable time to respond. Many data analysis applications can tremendously benefit from using early approximate results, which can be available before job completion. Such applications gives some inaccuracy in the results. Early accurate approximation techniques are also implemented in relational databases. Wavelet transformations, histogram-based approximated results or sampling techniques are few approximated techniques used in modern large scale data processing systems.

For large scale data processing Hadoop is been used now days [10] [13]. Hadoop is an open source implementation of Google's MapReduce. It is the most popular and efficient big data processing framework. It has batch processing nature, but, it does not allow execution of job partially. The challenge with this systems is posed by the unstructured data format. In Hadoop the data is stored in a distributed file system i.e. in Hadoop Distributed File System. Once the job is submitted for execution the data from HDFS is transformed into key-value pairs.

This paper presents an early result estimation technique. I have proposed a simple online aggregation technique implementation, which will significantly improve the performance of Hadoop framework. Additional combiner is also added to MapReduce paradigm to reduce the processing time. The main objectives from the proposed system are: 1) to implement combiner with MapReduce paradigm in Hadoop; 2) implementation of online aggregation taking to provide the user with early results by taking the snapshots.

The rest of this paper is organized as follows. Section II and III discusses related work and necessary background for the paper respectively. In section IV, I have discussed motivation behind the work. In section V, proposed

IJESMR
International Journal OF Engineering Sciences & Management Research

system architecture along with mathematical model and multiplexer logic is discussed. In section VI experimental setup and results are given. Finally, the conclusions is given in section VII.

RELATED WORK

Chris Jermaine [10] [14], Proposes an Online Aggregation for Large-Scale Computing. Given the current interest on very large-scale and data-oriented computing. In this paper the author has discussed the problem of providing OLA. While they have concentrate on implementing OLA with a MapReduce engine.

The EARL library [5] is an extension to the Hadoop framework and focuses on accurate estimation of final results, while providing reliable error estimations. It uses the bootstrap technique, which is applied to a single pre-computed sample. Consequently, numerous subsamples are extracted and used to compute the estimate. The accuracy of estimation can be improved with an expansion of the initial sample size and increase in number of used subsamples. EARL allows estimations for arbitrary work-flows and requires only minimal changes to the MapReduce framework.

The BlinkDB [13] approximate query engine creates and uses pre-computed samples of various sizes to provide fast answers. It relies on two types of samples: large uniform random and smaller multi-dimensional stratified. Queries are evaluated on a number of selected samples and an initial estimate is produced. BlinkDB is based on the predictable query column sets (QCS) model, so it assumes that a constant set of data columns, used for group or filter predicates, exist in the stored datasets. As a result, the system can estimate results of standard aggregate queries easily. More complex queries, including arbitrary joins, are currently not supported. In case the accuracy or time constraints of a query are not met, larger or smaller samples can be selected. The accuracy of various queries estimation depends on the composition and sizes of stored samples.

Another important work builds Online Aggregation [7] for MapReduce jobs, using the Hyracks execution engine. The authors argue that Online Aggregation is newly relevant for the Cloud Computing cost model, as it can save computations and therefore money. In such a setup, where failures are also quite frequent, it is hard to guarantee the statistical properties of the partially processed input. They propose an operational model and a Bayesian framework for providing estimations and confidence bounds for the early returned results. However, in order to guarantee such properties, the system only supports applications conforming to a specialized interface and limited to the set of common aggregate functions.

Finally, Facebooks Peregrine [20] is a distributed low latency approximate query engine, built on top of Hive and HDFS. It supports a subset of operators and provides approximate implementations for some aggregate functions. A user has to explicitly use the approximate functions in their query and can get terminate it before the execution is complete. After termination, information is provided on the number of scanned records and possible failures that occurred during execution. In order to provide fast results, Peregrine uses one pass approximate algorithms and an in-memory serving tree framework for computing aggregations.

BACKGROUND

In this section, a briefly review of the MapReduce programming model, the MapReduce Online framework and the Online Aggregation technique is given.

A. Hadoop MapReduce Framework

MapReduce [1] [2] [9] is a programming model for largescale parallel data processing. There are two user-defined functions, map and reduce in the MapReduce programming model. One needs to specify the path to input data and for the output data to store the results. Data is organized in block format in HDFS. The data is read from the file system and shipped to map tasks. During map reduce processing time the data is parsed into key-value pairs. The output pairs are then grouped by key and are sent to parallel reduce tasks, which apply the reduce function on each group. The result of each reduce task produces one file which is then stored in the distributed file system. MapReduce has become popular, as it ensures efficient execution of tasks across large numbers of machines and successfully manages to hide the complex details of parallelization, data distribution, fault tolerance and load balancing from the user. Following are the phases in MapReduce paradigm [2]. Figure 1 describes the work flow of the Hadoop framework.

ISSN 2349-6193
Impact Factor (PIF): 2.243

IJESMR
International Journal OF Engineering Sciences & Management Research

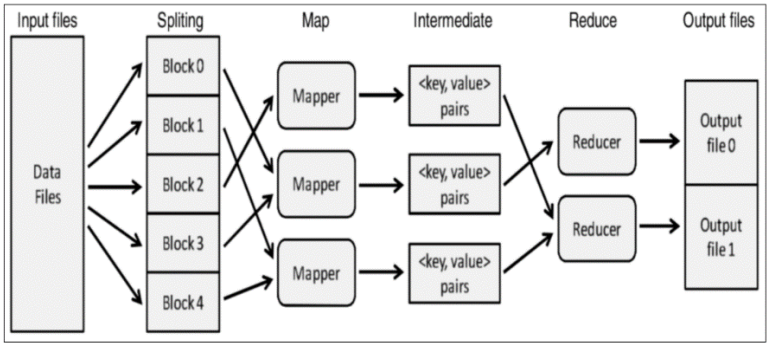

Figure 1: MapReduce Workflow

- Read: Reading the input split and creating the key-value pairs
- Map: Executing the user-provided map function
- Collect: Collecting the map output into a buffer and partitioning
- Spill: Sorting, using the combiner if any, performing compression if asked, and finally spilling to disk, creating file spills
- Merge: Merging the file spills into a single map output file. Merging might be performed in multiple rounds

The Reduce Task is also divided into four phases as given below:

- Shuffle: Copying the map output from the mapper nodes to a reducer's node and decompressing, if needed. Partial merging may also occur during this phase.
- Merge: Merging the sorted fragments from the different mappers to form the input to the reduce function.
- Reduce: Executing the user-provided reduce function.
- Write: Writing the (compressed) output to HDFS.

B. MapReduce Online

MapReduce Online [3] [8] is a modified version of Hadoop MapReduce. It supports Online Aggregation, which leads to reduced response time. Traditional MapReduce implementations materialize the intermediate results of mappers and do not allow pipelining between the map and the reduce phases. In this approach the reducers cannot start executing tasks before all mappers have finished. This limitation lowers resource utilization and leads to inefficient execution for many applications. The main motivation of MapReduce Online is to overcome these problems, by allowing pipelining between operators, while preserving fault-tolerance guarantees. C. Online Aggregation Online Aggregation [9] is a technique which enables interactive access to a running aggregation query. In general, when a query is submitted, no feedback is given during the query processing time. The accumulated results are returned only after the aggregation process is completed. The Online Aggregation technique enables partial query processing. There is no need of requiring prior knowledge of the query specifications. As a result, users are able to observe the progress of running queries and control their execution. The user can stop query processing in case early results are acceptable.

MOTIVATION

In Big Data processing, main problems is that good performance requires both good data locality and good resource utilization. A characteristic of Big Data processing is that the amount of data being processed is typically large with compared to the amount of computation done on it. The processing can benefit from data locality, which can be achieved by moving the computation close to the data using Hadoop. Still there are few challenges with Hadoop too which are discussed in introduction. MapReduce is supposed to be for batch processing and not for online transactions. The data from a MapReduce Job can be fed to a system for online processing. The current Hadoop MapReduce implementation does not provide such capabilities. As a result, for each MapReduce job, a customer has to assign a dedicated cluster to run that particular application, one at a time. Each cluster is dedicated to a single MapReduce application so if a user has multiple applications, the he has to run them in serial on that same resource or buy another cluster for the additional application. This gives the motivation to work towards improving the performance of Hadoop by implementing online aggregation technique in Hadoop MapReduce framework.

ISSN 2349-6193
Impact Factor (PIF): 2.243

IJESMR
International Journal OF Engineering Sciences & Management Research

PROPOSED SYSTEM

In this Section, I have described the aggregation technique in detail. In this section the design goals of system and the integration of the proposed technique with the Hadoop framework have been explained.

A. System Architecture

The section III gives Hadoop MapReduce work flow. Reduce tasks gather map outputs by issuing request to nodes on which execution of a map task is being done. The reduce task cannot request until the map task has completed and its final output has been committed to disk. This means that map task execution is completely decoupled from reduce task execution. To support pipelining of MapReduce job, I have proposed modified the map task paradigm. It will push data to reducers as it is produced from map functions. Imagine a job where the reduce key has few distinct values, and the reducer applies an aggregate function. Combiners allow map-side pre aggregation [6]. The combiner applies a function like reduce to each key at the mapper side. Because of this the network traffic can also be reduced. Figure 2 shows the proposed system architecture.

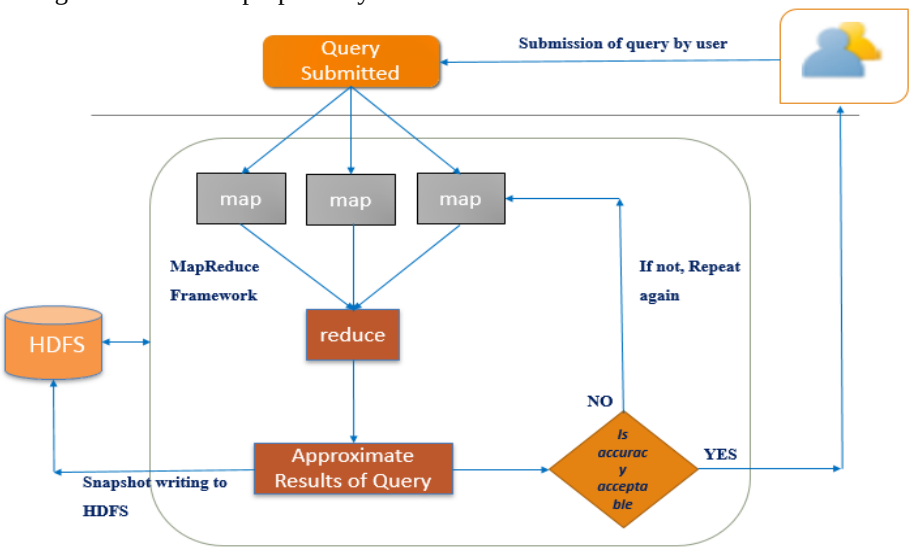

Figure 2: Architecture for Online Aggregation of MapReduce

1) Partitioner and Combiners:

The technique that can be used for aggregation is the combiner. Combiners provide a mechanism within the MapReduce framework [1] [5]. It reduces the amount of data generated by the mappers. The combiners can be considered as mini-reducers which process the output of mappers. If we consider the example of word count job, the combiners aggregate term counts on data processed by each map task. This results in a reduction in the number of intermediate key-value pairs that need to be shuffled across the network. An associative array (i.e., Map in Java) is introduced inside the mapper. This array tally up term counts within a single document. This version gives a key value pair for each unique term in the data instead of a key value pair for each term in the data being processed. For the same example we can consider some words appear frequently within a document. This method can save the number of intermediate key-value pairs emitted, especially for huge data.

2) Taking Snapshots:

In order to provide the user with intermediate approximated results, snapshots are needs to be taken. Duration of the snapshot has to be defined. As I discussed, in online aggregation on MapReduce, the output of the mapper will be sent to the reducers before the complete processing of the mapper is done. The reducer will process the data given by mapper as input and generates the results. This result will be considered as a snapshot and will be shown to user. If the result is acceptable then the user can stop further processing of map reduce. If the result is not acceptable then the mapper and reducer will continue with the processing.

B. Mathematical Model

ISSN 2349-6193
Impact Factor (PIF): 2.243

IJESMR
International Journal OF Engineering Sciences & Management Research

This section describes mathematical modeling of the proposed work. Turing machine is used to describe the processing of the system.

1) Multiplexer Logic:

In addition to mathematical model, the multiplexer logic also discussed here. As the figure 3 indicates, the data from the HDFS is given to mapper as input. The mapper processes the data and the result is then sent to the reducers. The number of mappers and the reducers will be depending on the programming logic used and the number blocks to be processed. As figure indicates, the result of reducers is written to the HDFS and same is then sent to the mapper for further processing. The whole process is executed continuously until the desired output is not achieved. The difference between the traditional Hadoop and the proposed system is that, the intermediate results of mapper are forcefully sent to the reducers to obtain the early results. Where as in traditional system the reducer cannot start until the mapper finishes it processing completely.

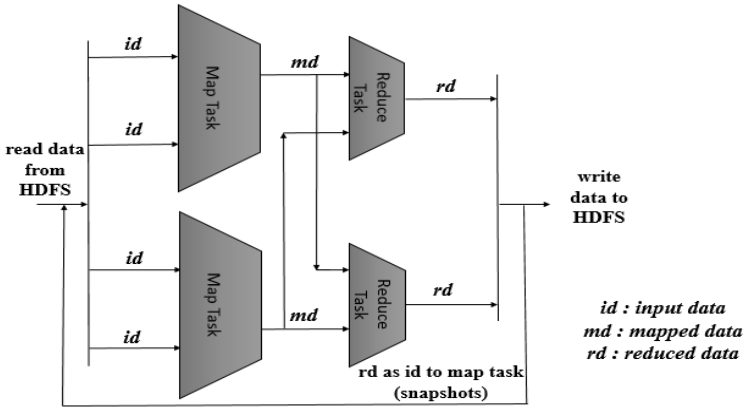

Figure 3: Multiplexer Logic for Aggregated MapReduce Framework

2) Performance Model:

The time taken for a Hadoop to execute a job on given data set can be described as

$$T = ov+(ps+pa)*h \quad (1)$$

where, ov = the fixed overhead in running a job
ps = processing time for Hadoop system
pa = processing time in map and reduce
h = time over which data is collected

Here, the dynamic part of the processing time is divided into two parts, one part related to Hadoop system and the other is with the application logic. There is a fixed overhead cost in running any Hadoop job, which is represented by ov. The overhead is mostly associated with the work involved in job. It is independent of the data size. The dynamic part of the time has two components. The first component represented by ps, contains all the time taken by Hadoop system for processing a job. This part is mostly disk and network IO bound. Bradly speaking it constitutes the following:

a) Input and output for map
b) Shuffle
c) Input and output for reduce

The second component represented by pa. It is the time taken by the application logic i.e. time spent in the map and reduce functions. It contains the following parts:

a) Execution of map
b) Execution of reduce

The overhead is fixed and not much can be done about it. The focus is on the dynamic part of the time. We can get the best performance when pa dominates over ov and ps.

EXPERIMENTAL RESULTS

The results obtained from the traditional Hadoop framework and implemented proposed system are taken on different sets of data. Both results are then compared to find out the conclusion. For both traditional and proposed system the same environment is used. The setup used for Hadoop and the results obtained are discussed below.

A. Environment

ISSN 2349-6193
Impact Factor (PIF): 2.243

IJESMR
International Journal OF Engineering Sciences & Management Research

The figure 4 describes the experimental setup of proposed system. The hadoop-1.2.1 is used for setup. Ubuntu 12.04 operating system is installed on all the nodes. Total 3 nodes are configured, one as master node and remaining two as slave nodes. The processors and memory allocated to the nodes is given in table 1.

	Description	Total Virtual Machines
Machine	Processor: Intel Corei5-4210U CPU @ 1.70 GHz 2.40 GHz RAM: 4.00 GB	1- Master 2- Slaves
Hadoop Version	Hadoop-1.2.1	
Virtual Machine	VMware Workstation Version 11.0.0	

Figure 4: Experimental Environment

Table No. 1: Specification for Each Node

Sr. No.	Machine	Memory	CPU	Disk
1	Master	4	2 GB	20 GB
2	Slave1	2	1 GB	20 GB
3	Slave1	2	1 GB	20 GB

B. Results

Simple word count job for 1 GB and 2 GB data is executed on the traditional Hadoop framework and the results were obtained. The time taken by the traditional Hadoop system is shown in figure 5. Then the same job is executed on modified Hadoop framework. The results are obtained for 1 GB and 2 GB data sets. The time taken by modified Hadoop framework is shown in figure 6. After comparing both the figures, it show that the time taken by the modified Hadoop paradigm is less with compared to traditional Hadoop paradigm.

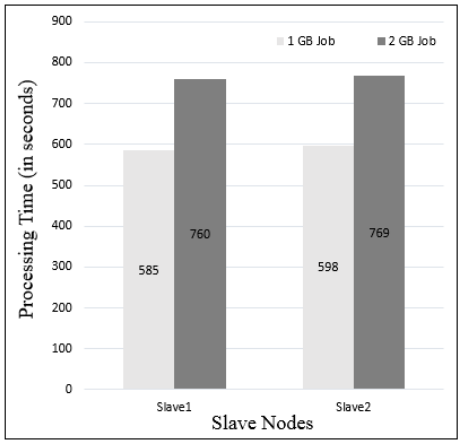

Figure 5: Execution Time of WordCount Job

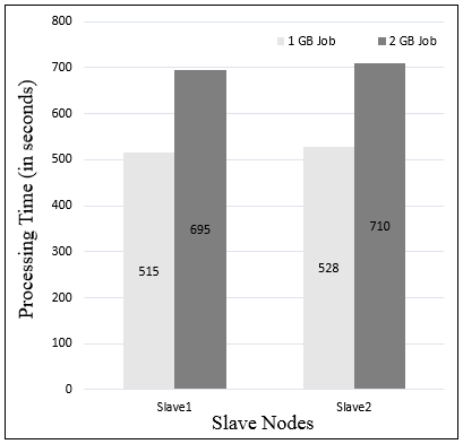

Figure 6: Impact of Combiners Strategy on Execution Time of WordCount Job

CONCLUSION

In this paper I have proposed an Online Aggregation approach for efficient big data processing. The aggregated results will be shown to the user and if user is agree with the obtained results then the further processing of MapReduce can be stopped. The early result will be shown to user by taking snapshots of the intermediate results obtained. As a local aggregate the combiners are implemented alongside with mapper and reducers for the aggregation. The combiners processes intermediate data produced by the mappers. And the result is then forwarded to reducers for further processing. The processing time of Hadoop is reduced at some instance.

ISSN 2349-6193
Impact Factor (PIF): 2.243

IJESMR
International Journal OF Engineering Sciences & Management Research

REFERENCES

1. S. Vikram Phaneendra and E. Madhusudhan Reddy, Big Data- solutions for RDBMS problems- A survey, In 12th IEEE/IFIP Network Operations and Management Symposium (NOMS 2010) (Osaka, Japan, Apr 19, 2013)
2. Kiran kumara Reddi & Dnvsl Indira, Different Technique to Transfer Big Data : survey, IEEE Transactions on 52(8) (Aug.2013) 2348 2355
3. Jimmy Lin MapReduce Is Good Enough?, The control project. IEEE Computer 32 (2013).
4. Jiawei Han and Micheline Kamber, Classification and Prediction in Data Mining: Concepts and Techniques, 2nd ed., San Francisco, CA The Morgan Kaufmann, 2006.
5. N. Laptev, K. Zeng, and C. Zaniolo, Early accurate results for advanced analytics on mapreduce, vol. 5, no. 10. VLDB Endowment, 2012, pp. 10281039.
6. Report from Pike research, http://www.pikeresearch.com/research/ smartgrid-data-analytics.
7. National Climate Data Center [Online]. Available:http://www.ncdc.noaa.gov/oa/ncdc.html
8. D. Borthakur, The Hadoop Distributed File System: Architecture and Design, 2007.
9. J. Hellerstein, P. Haas, and H. Wang,Online aggregation, In SIGMOD Conference, pages 171182, 1997.
10. C. Jermaine, S. Arumugam, A. Pol, and A. Dobra, Scalable approximate query processing with the dbo engine, In SIGMOD Conference, pages 725736, 2007.
11. The apache hadoop project page, http://hadoop.apache.org/, 2013, last visited on 1 May, 2013.
12. J. Dean and S. Ghemawat, Mapreduce: simplied data processing on large clusters, Communications of the ACM, vol. 51, no. 1, pp. 107 113, 2008.
13. S. Agarwal, A. Panda, B. Mozafari, S. Madden, and I. Stoica, Blinkdb: Queries with bounded errors and bounded response times on very large data, in ACM EuroSys 2013, 2013.
14. N. Pansare, V. R. Borkar, C. Jermaine, and T. Condie, Online aggregation for large mapreduce jobs, vol. 4, no. 11, 2011, pp. 11351145.