

An Efficient Approach for Processing Big Data with Incremental MapReduce

Solanke Poonam G¹ B. M. Patil²

^{1,2}Department of M.B.E.S

^{1,2}College of Engineering, Ambajogai

Abstract— Now a day, data is constantly evolving and becomes a big data. The data is being generated from different sources – transactions, social media, sensors, digital images, video, audio and clickstreams for domains together with healthcare, retail, energy and utilities. Big data with 3 V’s: volume, variety and velocity. For processing such big volume of data, variety of data and the data with high velocity and having high storage capacity, we introduced Hadoop which is evolved day by day. We used MapReduce at this point as a programming model. In this paper we worn Incremental MapReduce most extensively used framework for processing big data. To improve the time of processing big data and optimizing data content of big data we applied PageRank and k-means iteratively along with MapReduce. Therefore to process big data incremental MapReduce approach is used.

Key words: Big data, Hadoop, incremental MapReduce, iterative.

I. INTRODUCTION

Big data is the term for a collection of data sets so big and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications. Big data streams are characterized by having high volume data (data in petabytes and zetabytes), variety of data (data may be structured, semi structured or unstructured) and velocity of data (how fast data processes). To process such big data hadoop is evolved. Hadoop is an open source framework that allows to store and process big data in distributed environment across clusters of computers using simple programming models. Hadoop is a framework for running applications on large clusters. Modeled after Google’s MapReduce/GFS framework and implemented in Java. Amazon/A9, Facebook, Google, IBM, Intel Research uses hadoop.

A. Hadoop benefits

- Cost effective – Because it works on commodity hardware.
- Big cluster (1000 nodes on cluster) – Big storage and more processing power due to number of nodes on cluster.
- Parallel processing – In MapReduce framework thousands of nodes processes in parallel and generate results in timely manner.
- Big storage – Thousands of nodes with high capacity (100 GB). So storage capacity increases.
- Failover – Automatic failover. If node crashes, framework will identify automatically.
- Data distribution – Hadoop framework handles itself because it has thousands of nodes.
- MapReduce framework – Designed and will work on hadoop framework as map and reduce function.

- Moving code to data – Writes a MapReduce code, sends MapReduce code to cluster and get data.
 - Heterogeneous hardware – Can use hardware (CPU, memory etc.) of IBM, oracle, HP, Intel etc.
- Hadoop uses java framework for processing huge volume of data in cluster. Hadoop includes core technology
- 1) HDFS - a distributed filesystem for storage purpose and
 - 2) Map/Reduce - offline computing engine.

Hadoop File System was developed by using distributed file system design. It is run on commodity hardware. Contrasting other distributed systems, HDFS is highly fault-tolerant and designed using low-cost hardware. HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available to parallel processing.

MapReduce is a programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster. MapReduce program poised of a Map function and Reduce function. Map () procedure that performs filtering and sorting, reduce () method that performs a summary operation. MapReduce execution sequence is shown in fig.1.

map(K1,V1)→[<K2,V2>]
reduce(K2,{V2})→[<K3,V3>]

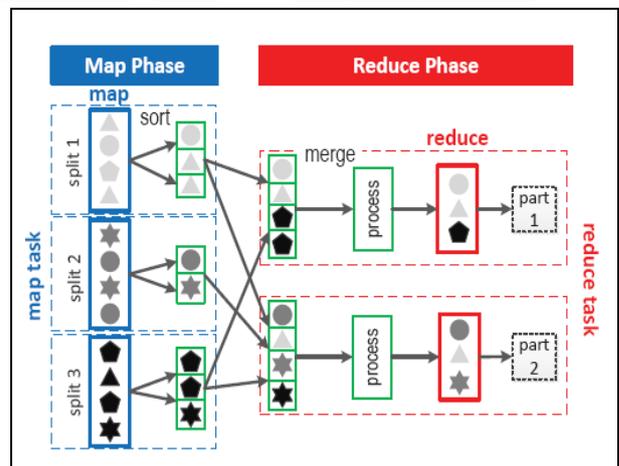


Fig. 1: MapReduce execution sequence

The MapReduce system orchestrates the processing by marshalling the distributed servers, running the different tasks in parallel, organization all communications and data relocate between the different parts of the system and provided that for redundancy and fault tolerance.

Incremental MapReduce is an addition to MapReduce used for processing big data efficiently and timely with the optimization in data content of big data and also required optimized time to process this big data.

II. LITERATURE SURVEY

We begin by reviewing the concept of MapReduce programming model, HaLoop, IterMR, Incoop and incMR

A. MapReduce

MapReduce is a programming model and an allied execution for processing and generating large data sets. Map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication but it takes a more time for processing [2].

B. HaLoop

Bu et al. used a new technique called as HaLoop which is modified version of Hadoop MapReduce Framework, as Map Reduce lacks built-in-support for iterative programs HaLoop allows iterative applications to be assembled from existing Hadoop programs without modification, and significantly improves their efficiency by providing inter-iteration caching mechanisms and a loop-aware scheduler to exploit these caches. HaLoop is built on top of Hadoop and extends it with a new programming model and several important optimizations that include (1) a loop-aware task scheduler, (2) loop-invariant data caching, and (3) caching for efficient fix point verification. HaLoop performs worse than plain MapReduce. This is because HaLoop employs an extra MapReduce job in each iteration to join the structure and state data [3].

A number of distributed frameworks have newly emerged for big data processing. HaLoop improves the efficiency of iterative computation by making the task scheduler loop-aware and by employing caching mechanisms. Twister employs a lightweight iterative MapReduce runtime system by sensibly constructing a Reduce-to-Map loop. IMapReduce supports iterative processing by directly passing the Reduce outputs to Map and by distinguishing variant state data from the static data [4].

C. Incoop

Incoop, which allows existing MapReduce programs, not designed for incremental processing, to execute transparently in an incremental manner. In Incoop, computations can respond automatically and efficiently to modifications to their input data by reusing intermediate results from previous computations, and incrementally updating the output according to the changes in the input. But Incoop Supports only task-level incremental processing and supports only one-step computation [5].

D. IncMR:

IncMR framework is for incrementally processing new data of a large data set, which takes state as implicit input and combines it with new data. Map tasks are created according to new splits instead of entire splits while reduce tasks fetch their inputs including the state and the intermediate results of new map tasks from designate nodes or local nodes. Data

locality is considered as one of the main optimization means for job scheduling. But optimization of data is not done so size and location of state data is not recognized [6].

III. IMPLEMENTATION DETAILS

Processing is done on a datasets which are the iterative with respect to its base file. Four iterative algorithms are used for the accomplishment. In section A, we design PageRank algorithm. Section B includes Kmeans algorithm. Section C and D includes the explanation of MapReduce and incremental MapReduce respectively. System architecture is shown in fig. 2.

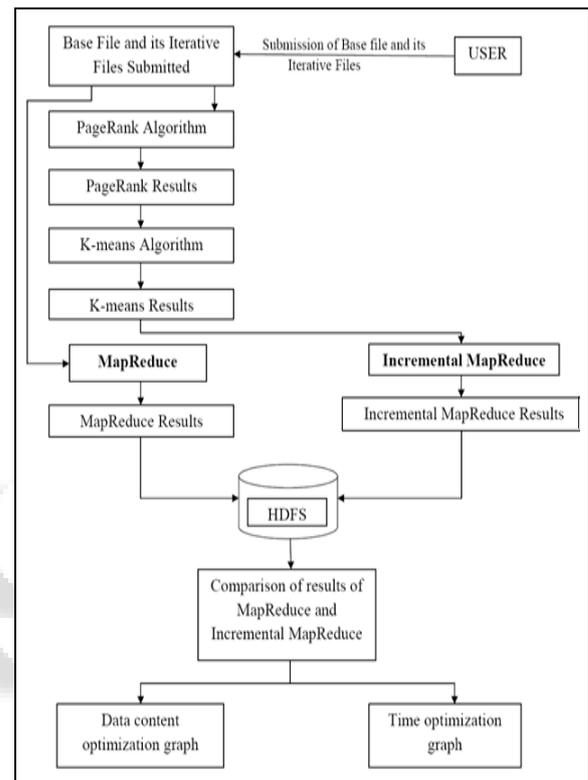


Fig. 2: System Architecture

A. PageRank

PageRank is a link investigation algorithm that assigns a numerical weighting to every element of a hyperlinked set of documents, with the point of measuring its comparative significance within the set. Votes throw by pages that are themselves "important" weigh up more deeply and facilitate to make further pages "important". PageRank is a well-known iterative graph algorithm for ranking web pages. It computes a ranking score for all vertex in a graph. After initializing all ranking scores, the working out performs a MapReduce job per iteration.

The PageRank algorithm is at the heart of the Google search engine. It is this algorithm that in spirit decides how significant a precise page is and thus how elevated it will demonstrate in a search result. In PageRank algorithm a page is important, if other important pages link to it. This scheme can be seen as a method of calculating the importance of pages by voting for them. Each link is viewed as a vote - a de facto suggestion for the importance of a page - whatever reasons the page has for connecting to a precise page. The PageRank-algorithm can, with this explanation, be seen as the contradict of an online survey, where pages

vote for the importance of others, and this result is then tallied by PageRank and is reflected in the search results.

B. K-means

Clustering is a classification of objects into dissimilar groups in such a way the partitioning of dataset into subsets so that data in each subset share some common feature according to some defined distance measure which is Euclidean distance measure.

Euclidean distance,

$$d = \sqrt{\sum_{i=1}^n (p_i - q_i)^2} \dots\dots\dots (3.1)$$

K-means clustering algorithm is a simple technique for estimating the mean of a set of k-group. K-means is a frequent and well-known clustering algorithm. It partitions a set of 'n' objects into 'k' clusters based on a similarity measurement of the objects in the dataset. The clusters have an prominent intra-cluster and a little inter-cluster similarity. As the number of objects in the cluster vary, the center of gravity of the cluster shifts.

Simply speaking k-means clustering is an algorithm to classify or to group the objects based on attributes/features into K number of group. K is positive integer number. The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid.

C. MapReduce

MapReduce is a programming model and an allied implementation for processing and generating big data sets. Users state a map function that processes a key/value pair to build a set of intermediary key/value pairs, and a reduce function that merges the total intermediary values allied with the same intermediate key. Implementation of MapReduce runs on a large cluster of product machines and is extremely scalable: an individual MapReduce calculation processes numerous terabytes of data on thousands of machines.

The solid part of the MapReduce output framework is a large distributed sort. The hot spots, which the application defines, are:

- Input reader – The input reader divides the input into proper size splits and the framework assigns on split to each map function. The input reader reads data from steady storage space and generates key/value pairs.
- Map function – The map function takes a series of key/value pairs, processes each, and generates zero or more output key/value pairs.
- Partition function – Each map function output is allocated to a particular reducer by the applications partition function for sharing purposes. The partition function is given the key and the number of reducers and returns the key of the desired reducer.
- Comparison function – the input for each reducer is pulled from the machine where the map ran and stores using the application's comparison function.
- Reduce function – The framework calls the applications reduce function one of each unique key in the stored order. The reduce function can iterate through the values that are associated with that key and produce zero or more outputs.
- Output writer – The output writer writes the output of the reduce function to the stable storage.

D. Incremental MapReduce

In order to hold up incremental and iterative processing, a few MapReduce APIs are changed or added. Incremental MapReduce is an enhancement in the MapReduce. Incremental MapReduce is a task-level coarse-grain incremental processing system. Incremental MapReduce exploits inspection to stay re-computation by preliminary since the previously converged state, and during the stage incremental updates on the changeable information. Incremental MapReduce improves the run time of re-computation on plain MapReduce by an eight fold speedup. Incremental MapReduce takes input file as the output of the k-means clustering algorithm. Therefore in incremental MapReduce there is optimization of data content. Incremental MapReduce requires computation time less than the computation time of MapReduce because of the optimization in the data content.

IV. IMPLEMENTATION RESULTS

In this section we present comparison results of MapReduce and the extension of MapReduce i.e. incremental MapReduce. In this result, comparison is done on the basis of data content optimization and time optimization. We have given input as a one base file and four iterative files. Then each iterative file is compared with the base file and gives the output in the reduced format. Output of MapReduce is a file of large size while the output of incremental MapReduce is a file of small size than the result of the MapReduce. Comparison results are shown in fig. 3.

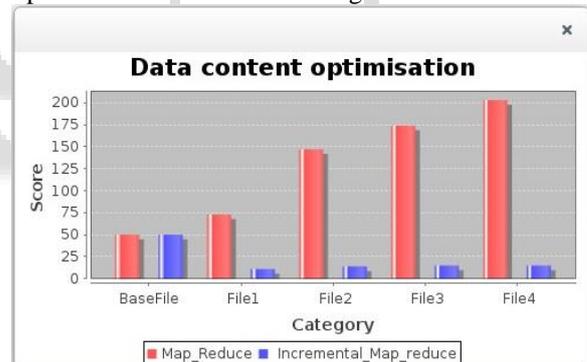


Fig. 3: Comparison graph of data content optimization in incremental MapReduce vs. plain MapReduce

Output time required for processing big data file in MapReduce is more while the output time required for processing big data file in incremental MapReduce is less than the result of the MapReduce. Comparison results are shown in fig. 4.

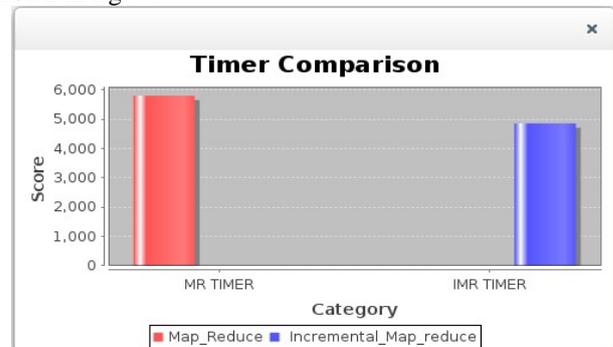


Fig. 4: Comparison graph of time optimization in incremental MapReduce vs. plain MapReduce

As shown in above fig. for the same input data sets there are different outputs for MapReduce and incremental MapReduce. Base file is of the same size in both MapReduce and incremental MapReduce. After the processing of iterative files with its base file, optimization of data content is done and processing time required in incremental MapReduce is also optimized.

From the above graph, we see that the more optimization of data content is occurred in the incremental MapReduce than the plain MapReduce. As more the iterative files taken, more optimized data in the incremental MapReduce. For this output, the processing of the algorithms is iterative processing. Which means the complete processing of algorithms should be iteratively in the order of their execution. Execution of PageRank algorithm, k-means clustering algorithm, MapReduce and Incremental MapReduce is done iteratively. For the execution of these algorithms there is no. of dependencies.

In this method the major merit is the optimization in the data content and processing time required. We are going to evaluate the performance of our projected scheme in terms of the computation overhead introduce by each operation. Base file and iterative files are taken as the computing parameter. When the number of iterative files increases, then there is more optimization on data content and time required for processing in incremental MapReduce.

V. CONCLUSION

We are processing big data here. For processing of big data hadoop is used. MapReduce is used as programming model for processing big data and incremental MapReduce is used for more efficient processing of big data. Incremental MapReduce Optimizes The Data Content And The Time Required For Processing Big Data.

ACKNOWLEDGMENT

I would like to sincerely express thanks to my guide Dr. B. M. Patil for his appreciable support, continuous encouragement and his invaluable suggestions. I am grateful for all the suggestions and hints provided by him. It was great moment to work with him.

REFERENCES

- [1] Yanfeng Zhang, Shimin Chen, Qiang Wang, and Ge Yu, Member, IEEE, "i²MapReduce: Incremental MapReduce for Mining Evolving Big Data" IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 27, NO. 7, JULY 2015.
- [2] Jeffrey Dean et al. MapReduce: Simplified data processing on large clusters. In Proceedings of the 6th USENIX OSDI, pages 137–150, 2004.
- [3] Yingyi Bu_ Bill Howe _ Magdalena Balazinska _ Michael D. Ernst "The HaLoop Approach to Large-Scale Iterative Data Analysis" VLDB 2010 paper "HaLoop: Efficient Iterative Data Processing on Large Clusters.
- [4] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox, "Twister: A runtime for iterative mapreduce," in Proc.19th ACM Symp. High Performance Distributed Comput., 2010, pp. 810–818.
- [5] P. Bhatotia, A. Wieder, R. Rodrigues, U. A. Acar, and R. Pasquin, "Incoop: Mapreduce for incremental computations," in Proc. 2nd ACM Symp. Cloud Comput., 2011, pp. 7:1–7:14.
- [6] C. Yan, X. Yang, Z. Yu, M. Li, and X. Li, "IncMR: Incremental data processing based on mapreduce," in Proc. IEEE 5th Int. Conf. Cloud Comput., 2012, pp.pp. 534–541.
- [7] Xindong Wu, Fellow, IEEE, Xingquan Zhu, Gong-Qing Wu, and Wei Ding" Data Mining with Big Data" IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 26, NO. 1, JANUARY 2014.
- [8] D. Gillick, A. Faria, and J. DeNero, MapReduce: Distributed Computing for Machine Learning, Berkley, Dec. 2006.
- [9] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis, "Evaluating MapReduce for Multi-Core and Multi-processor Systems," Proc. IEEE 13th Int'l Symp. High Performance Computer Architecture (HPCA '07),pp. 13-24, 2007.
- [10] Shadi Ibrahim_ Hai Jin_ Lu Lu "Handling Partitioning Skew in MapReduce using LEEN"ACM 51 (2008) 107–113.Ghoting and E. Pednault, "Hadoop-ML: An Infrastructure for the Rapid Implementation of Parallel Reusable Analytics," Proc. Large-Scale Machine Learning: Parallelism and Massive Data Sets Workshop (NIPS '09),2009.
- [11] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein "Online Aggregation and Continuous Query support in MapReduce" SIGMOD'10, June 6–11, 2010, Indianapolis, Indiana, USA. Copyright 2010 ACM 978-1-4503-0032-2/10/06.
- [12] Balaji Palanisamy, Member, IEEE, Aameek Singh, Member, IEEE Ling Liu, Senior Member, IEEE" Cost-effective Resource Provisioning for MapReduce in a Cloud"gartner report 2010, 25.
- [13] Niketan Pansare¹, Vinayak Borkar², Chris Jermaine¹, Tyson Condie "Online Aggregation for Large MapReduce Jobs" August 29September 3, 2011, Seattle, WA Copyright 2011 VLDB Endowment, ACM.
- [14] Kyong-Ha Lee and Hyunsik Choi "Parallel Data Processing with MapReduce: A Survey" SIGMOD Record, December 2011 (Vol. 40, No. 4).
- [15] Albert Bifet "Mining Big Data In Real Time" Informatica 37 (2013) 15–20 DEC 2012.
- [16] Y. Zhang, Q. Gao, L. Gao, and C. Wang, "imapreduce: A distributed computing framework for iterative computation,"J. Grid Comput., vol. 10, no. 1, pp. 47–68, 2012.
- [17] Jonathan Stuart Ward and Adam Barker "Undefined By Data: A Survey of Big Data Definitions" Stamford, CT: Gartner, 2012.
- [18] Jimmy Lin "MapReduce Is Good Enough?" The control project. IEEE Computer 32 (2013).
- [19] D. G. Murray, F. McSherry, R. Isaacs, M. Isard, P. Barham, and M. Abadi, "Naiad: A timely dataflow system," inProc. 24th ACM Symp. Oper. Syst. Principles, 2013, pp. 439–455.

- [20] Kiran kumara Reddi & Dnvsl Indira “Different Technique to Transfer Big Data : survey” IEEE Transactions on 52(8) (Aug.2013) 2348 { 2355}.
- [21] Kenn Slagter Ching-Hsien Hsu “An improved partitioning mechanism for optimizing massive data analysis using MapReduce” Published online: 11 April 2013 © Springer Science+Business Media New York 2013.
- [22] Ahmed Eldawy, Mohamed F. Mokbel “A Demonstration of SpatialHadoop:An Efficient MapReduce Framework for Spatial Data”Proceedings of the VLDB Endowment, Vol. 6, No. 12 Copyright 2013 VLDB Endowment 21508097/13/10.
- [23] Sameer Agarwal†, Barzan MozafariX, Aurojit Panda†, Henry Milner†, Samuel MaddenX, Ion Stoica “BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data” Copyright © 2013i ACM 978-1-4503-1994 2/13/04.
- [24] Bernice Purcell “The emergence of “big data” technology and analytics” Journal of Technology Research 2013.
- [25] Umasri.M.L, Shyamalagowri.D, Suresh Kumar.S “Mining Big Data:- Current status and forecast to the future” Volume 4, Issue 1, January 2014 ISSN: 2277 128X.
- [26] Jonathan Paul Olmsted “Scaling at Scale: Ideal Point Estimation with ‘Big-Data” Princeton Institute for Computational Science and Engineering 2014.
- [27] Mrigank Mridul, Akashdeep Khajuria, Snehasish Dutta, Kumar N “ Analysis of Bidgata using Apache Hadoop and Map Reduce” Volume 4, Issue 5, May 2014” 27.

