ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The Cost-Effective Iterative MapReduce in Big Data Environment

K.Chandra Sekhar
Department of Information Technology
MVSR Engineering College,
Hyderabad, Telangana - 501510, India.

P.Karthik
Department of Information Technology
MVSR Engineering College,
Hyderabad, Telangana - 501510, India.

Abstract
Data is expanding step by step with the Development of Information Technology. We could separate more important Information from the tremendous scale data. Presently a day's practically every online clients hunt down items, administrations, points of interest and so forth to figure PageRank utilizing the MapReduce approach to parallelization. This gives us a method for registering PageRank that can on a fundamental level be consequently parallelized, thus conceivably scaled up to huge connection charts, i.e., to expansive accumulations of webpages. Depict a solitary machine usage which effectively handles a million or so pages. We utilize a group to scale out much further –be intriguing to perceive how far we can get. About PageRank and MapReduce finally survey the essential actualities. How about we begin with PageRank. Hadoop Map-Reduce is a product system for effectively composing applications which process unfathomable measures of data in parallel on expansive bunches of item equipment in a solid, shortcoming tolerant way.

Keywords: ***Analysis, Big Data, Hadoop, Map Reduce, Report, Security.***

Introduction

In Big data the data originates from different, heterogeneous, self-ruling sources with complex relationship and persistently developingupto 2.5 quintillion bytes of data are made every day and 90 percent data on the planet today were delivered inside recent years .for instance Flicker, an open picture sharing site, where in a normal 1.8 million photographs for every day are get from February to walk 2012.this demonstrates that it is extremely troublesome for big data applications to oversee, prepare and recover data from substantial volume of data utilizing existing programming devices. It's ended up test to extricate proficient data for future use .There are diverse difficulties of Data mining with Big Data. We neglect it in next segment. At present Big Data preparing relies on parallel programming models like MapReduce, and in addition giving registering stage of Big Data administrations. Data mining calculations need to look over the preparation data for getting the insights for unraveling or advancing model parameter. Because of the substantial size of data it is getting to be costly to examination data shape. The Map-Reduce based methodology is utilized for data block emergence and mining over huge datasets utilizing all-encompassing measures like most regular questions. Our paper is sorted out as takes after: first we will see key difficulties of Big Data Mining then we neglect a few techniques like, MapReduce and Page Rank algorithm [1-3]. Map-Reduce are a disseminated parallel programming model acquainted by Google with backing huge data preparing. To start with form of the Map Reduce library was composed in February 2003. The programming model is motivated by the guide and lessens primitives found in Lisp and other useful dialects.

MapReduce for Big Data ApplicationsRecently MapReduce has emerged as one of the most popular computing frameworks for Big Data processing because of the simple programming model and also includes the automatic management of the parallel execution. The MapReduce framework and the open source implementation is widely adopted the major and leading companies like Yahoo!, Facebook and Google. The computation in MapReduce framework has been divided into two main phases, that is map and reduce phases, that in turn are carried out by different map tasks and reduce

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

tasks, respectively. During the map phase, map tasks are launched in parallel in order to convert the input splits to intermediate data to form the key/value pairs. The local machine stores these key/value pairs then they are organized into multiple data partitions, one per reduce task. During the reduce phase, each of the reduce task fetches its part of data partitions from all map tasks to generate the final result. In between the map phase and the reduce phase, there is a shuffle step. During this step, the data produced by the map phase are ordered, partitioned and transferred to the appropriate machines executing the reduce phase. During this time, it causes a huge volume of traffic from the network traffic pattern carried out from all map tasks to all reduce tasks. The network traffic imposes a serious constraint on the efficiency of data analytic applications. To explain with example, when we have a tens of thousands of machines, the data shuffling could account for 58.6% of the cross-pod traffic and it amounts to over 200 petabytes in total in the analysis of SCOPE jobs [7]. There will be considerable performance overhead in case of the shuffle- heavy MapReduce tasks, which could be up to 30-40 %. A hash function shuffles the intermediate data by default in Hadoop, which then leads to a high network traffic as it ignores network topology and data size associated with each key.

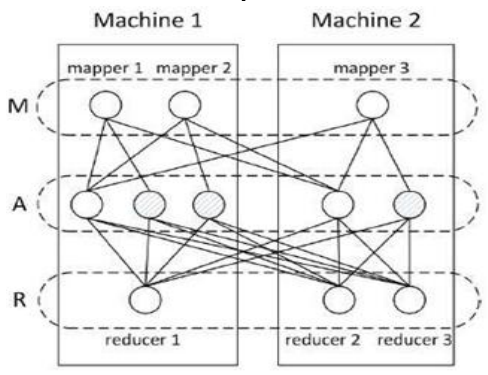


Figure: 1. Three layer model for Traffic Aware optimization

To tackle the problem of high network usage, incurred by the traffic-oblivious partition scheme, we take into account of both task locations and data size associated with each key in the project. By assigning keys with larger data size to reduce tasks closer to map tasks, network traffic can be significantly reduced. To further reduce network traffic within a MapReduce job, we consider to aggregate data with the same keys before sending them to remote reduce tasks. By aggregating the data of same keys before sending them reducers we reduce network traffic as shown in the Fig. 1.

Literature Review

Yanfeng Zhang et al [1] "i2MapReduce: Incrementing the MapReduce for Mining Evolving of the Big Data", VOL. 27, NO. 7, JULY 2015. Zaharia et al [2] proposed framework on the strong appropriated datasets of guide diminish. A shortcoming tolerant reflection for the in-memory of group processing, a web administration is encountering mistakes and an administrator needs to hunt terabytes data of the guide decrease of logs in the Hadoop record framework (HDFS) to discover the cause in big data mining. Utilizing Spark, the administrator can stack only the blunder messages from the logs into Random Access Memory over a set or the fields of hubs and question them intelligently in the data sets. J. Li et al [3] proposed framework is utilized as a part of building quick, conveyed programs with parceled tables, with the expanded accessibility of data focuses and cloud stages, software engineers from various issue areas confront the errand of composing parallel applications that keep running crosswise over numerous hubs. These application range from machine learning issues (k-implies bunching, neural networks preparing), chart calculations (PageRank), experimental calculation and so on. A hefty portion of these applications broadly get to and change shared transitional state put away in memory.

Mihaylov et.al [5] framework helpful in recursive, delta based data driven calculation, Web and interpersonal organization situations, inquiry workloads incorporate impromptu and OLAP inquiries, and in addition iterative calculations that investigate data connections (e.g., join examination, bunching, learning). Advanced DBMSs bolster impromptu and OLAP inquiries, however most are not sufficiently vigorous to scale to vast bunches. Then again, cloud stages like MapReduce execute chains of bunch errands crosswise over groups in a shortcoming tolerant manner, however have a lot of overhead to bolster specially appointed questions. Ewen et al [7] created framework that considers Spinning quick iterative data streams, a technique to coordinate incremental cycles, a type of work set emphasess, with parallel data streams.In the wake of demonstrating to coordinate mass emphasess into a dataflow framework and its streamlining agent, displaying an expansion to the programming model for incremental cycles. The augmentation lightens for the absence of changeable state in dataflow and takes into consideration misusing the scanty computational conditions inborn in numerous iterative calculations. The assessment of a prototypical execution demonstrates that those viewpoints lead to up to two requests of extent speedup in calculation runtime, when misused.

Howe et. al [6]proposed framework called as Hadoop -Efficient iterative data handling on extensive groups, the developing interest for largescale data mining and data investigation applications has driven both industry and the scholarly world to design new sorts of exceptionally versatile data-escalated registering stages. MapReduce and Dryad are two prevalent stages in which the dataflow appears as a coordinated non-cyclic chart of administrators. These stages need worked in backing for iterative projects, which emerge actually in numerous applications including data mining, web positioning, diagram investigation, model fitting, etc. Hadoop, a changed variant of the Hadoop MapReduce structure that is intended to serve these applications. Hadoop not just develops MapReduce with programming support for iterative applications, it likewise drastically enhances their productivity by making the errand scheduler circle mindful and by including different storing systems. We assessed Hadoop on genuine questions and genuine datasets. Contrasted and Hadoop, overall, Hadoop diminishes question runtimes by 1.85, and rearranges just 4 percent of the data amongst mappers and reducers.

Y. Bu, B. et.al [8] Map Reduce and Dryad are two popular platforms in which the dataflow takes the form of a directed acyclic graph of operators. These platforms lack built-in support for iterative programs, which arise naturally in many applications including data mining, web ranking, graph analysis, model fitting, and so on. Map Reduce with programming support for iterative applications, it also dramatically improves their efficiency by making the task scheduler loop-aware and by adding various caching mechanisms Ekanayake et al [9] proposed framework known as Twister: A runtime for iterative mapreduce, MapReduce programming model has streamlined the usage of numerous data parallel applications. The straightforwardness of the programming model and the nature of administrations gave by numerous usage of MapReduce draw in a ton of energy among dispersed registering groups. From the years of involvement in applying MapReduce to different investigative applications we distinguished an arrangement of augmentations to the programming model and changes to its design that will extend the appropriateness of MapReduce to more classes of uses. D. Logothetis et.al [13] the need for stateful dataflow programs that can rapidly sift through huge, evolving data sets. These data-intensive applications perform complex multi-step computations over successive generations of data inflows, such as weekly web crawls,daily image/video uploads, log files, and growing social networks.While programmers may simply re-run the entire dataflow when new data arrives, this grossly inefficient, increasing result latency and squandering hardware resources and energy.

For example, incrementally computing PageRank using CBP can reduce data movement by 46% and cut running time in half.
P.Bhatotia et.al [15]Many online data sets grow incrementally over time as new entries are slowly added and existing entries are deleted or modified. Taking advantage of this incrementality, systems for incremental bulk data processing, such as Google's Percolator, can achieve efficient updates. This efficiency, however, comes at the price of losing compatibility with the simple programming models offered by non-incremental systems, e.g., Map Reduce,

and more importantly, requires the programmer to implement application-specific dynamic/ incremental algorithms, ultimately increasing algorithm and code complexity. J. Cho and H. Garcia-Molina[16] crawler selectively and incrementally updates its index and/or local collection of web pages, instead of periodically refreshing the collection in batch mode. The incremental crawler can improve the \freshness" of the collection signi_cantly and bring in new pages in a more timely manner.

S. Kang et.al[19] Programs are expressed as a sequence of iterations, in each of which a vertex canreceive messages sent in the previous iteration, send messages of other vertices, and modify its own state and that of its outgoing edges or mutate graph topology. This vertex centric approach is exible enough to express a broad set of algorithms. The model has been designed for efficient, scalable and fault-tolerant implementation on clusters of thousands of commodity computers, and its implied synchronicity makes reasoning about programs easier. Y. Zhang, et.al[21]propose a distributed computing framework, PrIter, which enables fast iterative computation by providing the support of prioritized iteration.

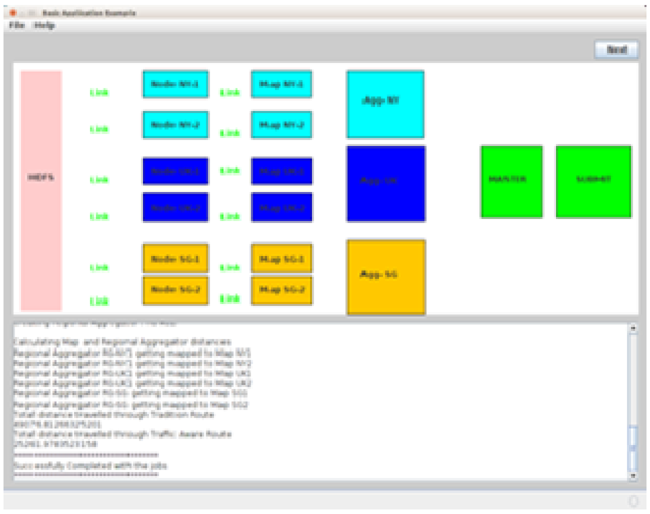
Proposed Method

The processing of large-scale data is simplified using the MapReduce programming model which works very well on the commodity cluster which exploits the processing by processing of map tasks and reduce tasks in parallel.

There have been many efforts that have been made towards in order to enhance performance of jobs of MapReduce; in shuffle phase network traffic that is generated is ignored many a times. It plays a key role in improving the performance. Historically, partition of intermediate data is done using a hash function in reduce tasks. However, data size and network topology for each key are not considered. Hence this is not efficient from traffic perspective [6-9].

Following are the two issues this project aims to resolve:

1. Network traffic: Cost of the network traffic has to be decreased using a novel intermediate data partition method.
2. Aggregator placement problem: An algorithm called the distribution algorithm which is designed to solve the issue of optimize the big-scale decomposition of the big data applications.



```
Calculating Map and Regional Aggregator distances
Regional Aggregator RG-NY1 getting mapped to Map NY1
Regional Aggregator RG-NY1 getting mapped to Map NY2
Regional Aggregator RG-UK1 getting mapped to Map UK1
Regional Aggregator RG-UK1 getting mapped to Map UK2
Regional Aggregator RG-SG getting mapped to Map SG1
Regional Aggregator RG-SG getting mapped to Map SG2
Total distance travelled through Tradition Route
490776.81268325201
Total distance travelled through Traffic Aware Route
252861.9783521158
==================================
Successfully Completed with the jobs
==================================
```

Figure: 2. Proposed MapReduce Model with Aggregators

In this section, we develop a distributed algorithm to solve the problem on multiple machines in a parallel manner. Our basic idea is to decompose the original large-scale problem into several distributively solvable subproblems that are coordinated by a high-level master problem. The model is as shown in the Figure 3. The figure shows the MapReduce with the aggregators placed for processing for traffic aware scheme [4-5].

Algorithm 1 Distributed Algorithm

```
1: set $t = 1$, and $\nu_j^p (j \in A, p \in P)$ to arbitrary
   nonnegative values;
2: for $t < T$ do
3:     distributively solve the subproblem SUB_DP and
       SUB_AP on multiple machines in a parallel man-
       ner;
4:     update the values of $\nu_j^p$ with the gradient method
       (15), and send the results to all subproblems;
5:     set $t = t + 1$;
6: end for
```

Figure: 3. Distributed Algorithm

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

In this section, we verify that our distributed algorithm can be applied in practice using real trace in a cluster consisting of 5 virtual machines with 1GB memory and 2GHz CPU. Our network topology is based on three tier architectures: an access tier, an aggregation tier and a core tier (Fig. 4). The access tier is made up of cost effective Ethernet switches connecting rack VMs [2].

The access switches are connected via Ethernet to a set of aggregation switches which in turn are connected to a layer of core switches. An inter-rack link is the most contentious resource as all the VMs hosted on a rack transfer data across the link to the VMs on other racks. Our VMs are distributed in three different racks, and the map-reduce tasks are scheduled as in Fig. 6. For example, rack 1 consists of node 1 and 2; mapper 1 and 2 are scheduled on node 1 and reducer 1 is scheduled on node 2. The intermediate data forwarding between mappers and reducers should be transferred across the network. The hop distances between mappers and reducers are shown in Fig. 4, e.g., mapper 1 and reducer 2 has a hop distance 6.data forwarding between mappers and reducers should be transferred across the network.

The hop distances between mappers and reducers are shown in Fig. 4, e.g., mapper 1 and reducer 2 has a hop distance 6.

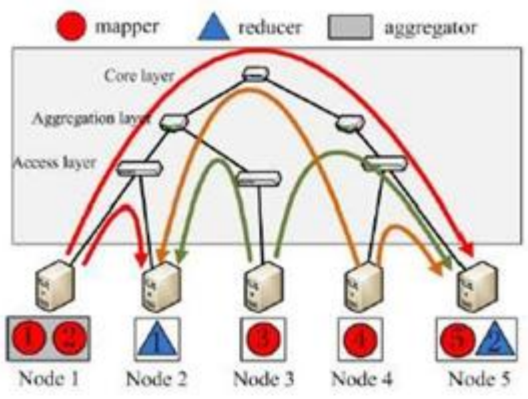


Figure 4. A small example

The intermediate data from all mappers is transferred according to the traffic-aware partition scheme. We can get the total network 2690:48 in the real Hadoop environment while the simulated network cost is 2673:49. They turn out to be very close to each other, which indicates that our distributed algorithm can be applied in practice.

CONCLUSION

In this system, we look into so that we can reduce network traffic cost for a MapReduce job by designing a novel intermediate data partition scheme. Furthermore, we jointly consider the aggregator placement problem, where each aggregator can reduce merged traffic from multiple map tasks. A decomposition-based distributed algorithm is proposed to deal with the large-scale optimization problem for big data application and an online algorithm is also designed to adjust data partition and aggregation in a dynamic manner. The partition and aggregators help to add to distance aware routing for processing the data for the big data applications. Placing the aggregators as close to the nodes and the client would also add to the network traffic reduction and in turn helps to reduce the cost of the data processing.

References

[1] Huan Ke, Student Member, IEEE, Peng Li, Member, IEEE, Song Guo, Senior Member, IEEE, and Minyi Guo, Senior Member, IEEE "On Traffic-Aware Partition and Aggregation in MapReduce for Big Data Applications " IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 27, NO. 3, MARCH 2016

[2] Y. Wang, W. Wang, C. Ma, and D. Meng, “Zput: A speedy data uploading approach for the hadoop distributed file system,” in Cluster Computing (CLUSTER), 2013 IEEE International Conference on. IEEE, 2013, pp. 1–5.

[3] T. White, Hadoop: the definitive guide: the definitive guide. ” O’Reilly Media, Inc.”, 2009.

[4] S. Chen and S. W. Schlosser, “Map-reduce meets wider varieties of applications,” Intel Research Pittsburgh, Tech. Rep. IRP-TR-08-05, 2008.

[5] J. Rosen, N. Polyzotis, V. Borkar, Y. Bu, M. J. Carey, M. Weimer, T. Condie, and R. Ramakrishnan, “Iterative mapreduce for large scale machine learning,” arXiv preprint arXiv:1303.3517, 2013.

[6] S. Venkataraman, E. Bodzsar, I. Roy, A. AuYoung, and R. S. Schreiber, “Presto: distributed machine learning and graph processing with sparse matrices,” in Proceedings of the 8th ACM European Conference on Computer Systems. ACM, 2013, pp. 197– 210.

[7] A. Matsunaga, M. Tsugawa, and J. Fortes, “Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications,” in eScience, 2008. eScience’08. IEEE Fourth International Conference on. IEEE, 2008, pp. 222–229.

[8] J. Wang, D. Crawl, I. Altintas, K. Tzoumas, and V. Markl, “Comparison of distributed data-parallelization patterns for big data analysis: A bioinformatics case study,” in Proceedings of the Fourth International Workshop on Data Intensive Computing in the Clouds (DataCloud), 2013.

[9] R. Liao, Y. Zhang, J. Guan, and S. Zhou, “Cloudnmf: A mapreduce implementation of nonnegative matrix factorization for largescale biological datasets,” Genomics, proteomics & bioinformatics, vol. 12, no. 1, pp. 48–51, 2014.