International Journal For Technological Research In Engineering
Volume 4, Issue 5, January-2017 *ISSN (Online): 2347 - 4718*

MAP REDUCE: NETWORK TRAFFIC COST REDUCTION IN BIG DATA APPLICATIONS

G. Pranusha[1], A. Obulesu[2], Dr. G. Vishnu Murthy[3]
[1]M.Tech Student, [2]Assistant Professor, [3]Professor & head,
Department of CSE, CVSR College of Engineering, Village Venkatapur, Mandal Ghatkesar,
Dist Ranga reddy, Telangana, India.

ABSTRACT:* Map reduce was projected to create simpler the parallel distributing using a distributed computing platform that gives only 2 interfaces. Map reduce has commence as a major model for processing information in large data centers. Map Reduce could be a 3 part rule consisting of Map, Shuffle and reduce phases. As a result of its intensive deployment, there are varied recent papers exactness sensible schemes to induce higher the performance of Map reduce systems. Of these exhausting work focuses on one in all the 3 phases to get performance improvement. To reduce network traffic at intervals a Map Reduce employment, we tend to regard to combination information to send them to distant reduce tasks with same keys. In existing system, a decomposition-based distributed algorithm and on-line rule is employed to manage the large-scale improvement drawback and aggregation of data in a very dynamic manner severally. This paper mainly focuses thoroughly on the system method of implementing Partitioning cluster.

I. INTRODUCTION

Big data could be a term that refers to information sets or mixtures of data sets whose size (volume), quality (variability), and rate of growth (velocity) build them troublesome to be captured, managed, processed or analyzed by standard technologies and tools, like relative databases. Hadoop Map Reduce programming model is getting used for process Big Data that consists of information process functions: Map and Reduce. Parallel Map tasks are run on computer file that is partitioned into fastened sized blocks and turn out intermediate output as a group of &lt;key, value&gt; pairs. These pairs are shuffled across totally different reduce tasks supported &lt; key, value&gt; pairs. Every reduce task accepts just one key at a time and process information for that key and outputs the results as &lt; key, value&gt; pairs. The Hadoop Map Reduce design consists of one Job Tracker (Master) and lots of Task Trackers (Workers). The Map Reduce on-line could be a changed version of Hadoop Map Reduce that supports on-line Aggregation and reduces latent period. Ancient Map cut back implementations pass the intermediate results of mapper and don't enable pipelining between the map and therefore the reduce phases. This approach has the advantage of easy recovery within the case of failures, however, reducers cannot start execution tasks before all mapper have finished. This limitation lowers resource utilization and ends up in inefficient execution for several applications. The most motivation of Map cut back on-line is to beat these issues, by allowing pipelining between operators, whereas protective Fault tolerance guarantees. Redis is an ASCII text file, networked, in-memory, key-value information store with optional durability. It’s written in ANSI C. The name Redis suggests that Remote wordbook Server. In its outer layer, the Redis information model may be a wordbook that maps keys to values. one among the most variations between Redis and alternative structured storage systems is that Redis supports not only strings, however additionally abstract information sorts like lists of strings, sets of strings (collections of non-repeating unsorted elements), sorted sets of strings (collections of non-repeating elements ordered by a number referred to as score), hashes wherever keys and values are strings. The kind of a worth determines what operations (called commands) are on the market for the worth itself. Redis supports high-level, atomic, server side operations like intersection, union, and distinction between sets and sorting of lists, sets and sorted sets; The main goal of the project work is to implement on-line Map Reduce and Redis on the highest of the Hadoop, which will improve the performance of Hadoop for economical massive information processing.

II. RELATED WORK

Different aspects of information center networks are studied in the literature. Kandula et al. study one massive information center running map-reduce applications, gathering data from the network traffic. They conclude that 86 of the aggregation and core switches conferred congestion of quite ten seconds. Exploitation common strategies to infer an information center traffic matrix from link level, SNMP information don't turn out satisfactory results. During this situation, comparison to the package level information, there is a mean error of hour on the traffic estimation. Benson et al. show that information centers supported three-tier tree topologies concentrate the traffic in top-of-rack switches. The study additionally reveals that patterns of virtualization and consolidation don't seem to be nonetheless detected. They conclude that operators choose the placement wherever services are going to be deployed; therefore the positioning isn't done willy-nilly. On those information centers, core switches have high utilization, with a high variation in a very day, and aggregation and edges switches have less traffic, with low variation. Their work shows that the location of VMs is important and should differ consistent with the sort of information center however additionally that the traffic distribution is dynamic. Benson et al. analyze nineteen information centers with totally different topologies and applications, exploitation packet traces and SNMP statistics.

The topology could be a 2-tier or a 3-tier on all of them but the applications vary among them, creating the amount of flows and total traffic on the network have a special behavior for each information center. They observe some traffic characteristics, just like the ON/OFF pattern of traffic arrival rate, with a lognormal distribution. Other proposals recommend the improvement of virtual machine placement within the information center. Sandpiper implements machine-controlled detection and migration of virtual machines. Sandpiper tries to spot and solve mainframe and memory hotspots in a very data center, observation the usage of every server and also the wants of each virtual machine, and migrating virtual machines as needed. Totally different from our work, sandpiper doesn't take the network traffic into consideration, thus we are able to not compare the data center network performance to VMP. Meng et al. discuss virtual machine placement within the data center. They formulate a reduction drawback based mostly on fastened prices of needed information measure of every virtual machine and the value of communication between the virtual machines, using min-cut algorithms to seek out divisions. Mainframe and memory resources don't seem to be employed in the rule since they assume that each virtual machine has constant size and every server supports a hard and fast range of VMs. They show that the virtual machine placement as associate improvement drawback is NP-hard. To reduce the complexness of the projected rule, they analyze the data employing a situation with constant traffic or grouping specific VMs, reducing the amount of analyzed parts. Wang et al. propose a model of VM consolidation with network information measure constraint imposed by network devices, where VMs with best-known information measure demands should be consolidated into variety of servers with identical capability limit. Their work isn't involved with traffic engineering of the core network; it only models the VM consolidation into every server of the network. The model doesn't use mainframe, memory and core utilization within the calculations. Biran et al, Describe a reduction drawback to see the location of VMs in a very information center based mostly within the network bandwidth, CPU, and memory resources. Their formulation is advanced and doesn't scale to the dimensions of knowledge centers, thus they additionally produce two heuristics supported the reduction algorithm. The rule must be dead in a very tree topology which can be achieved by transformation, forward that any topology will be reworked. The calculation is created off-line, and at each modification it has to be dead. They assume that each user features a specific range of VMs which will only ask each other, thus all the VMs are already clustered in their approach.

III. FRAME WORK

The Map reduce on-line could be a changed version of Hadoop Map Reduce, a preferred ASCII text file implementation of the Map reduce programming model. It supports on-line Aggregation and stream process, whereas additionally up utilization and reducing latent period. Ancient Map Reduce implementations take place the intermediate results of mapper and don't enable pipelining between the map and also the reduce phases. This approach has the advantage of easy recovery within the case of failures, however, reducers cannot begin execution tasks before all mapper have finished. This limitation lowers resource utilization and results in inefficient execution for several applications.

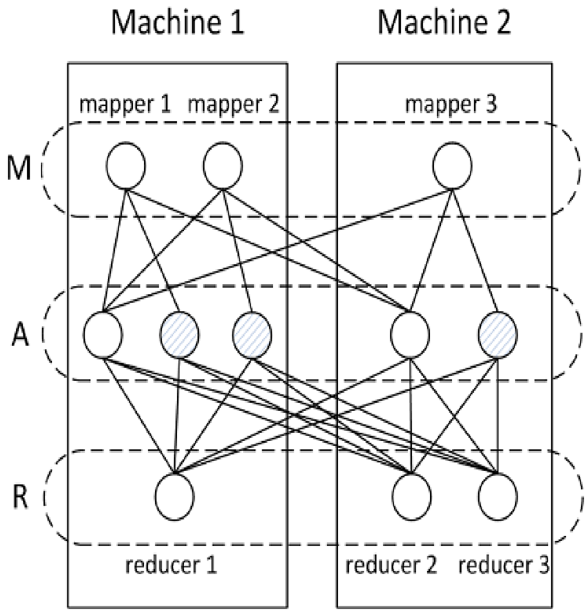
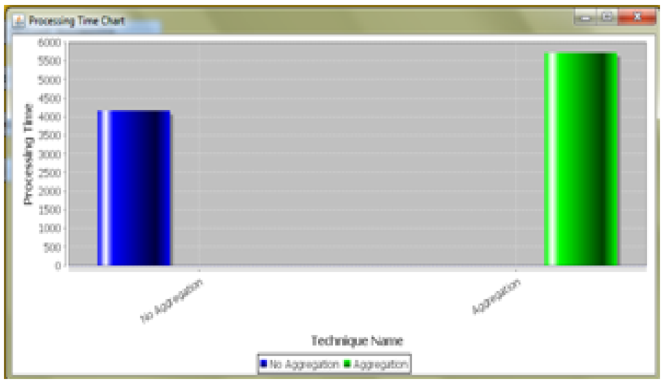


Fig. Three-layer model for the network traffic minimization problem.

The main motivation of Map reduce on-line is to overcome these issues, by permitting pipelining between operators, whereas protective fault-tolerance guarantees. Although Map reduce was originally designed as a batch oriented system, it's usually used for interactive information analysis: a user submits a job to extract data from a data set, then waits to look at the results before proceeding with successive step within the information analysis method. This trend has accelerated with the event of high level query languages that are dead as Map reduces jobs, like Hive, Pig. traditional Map reduce implementations offer a poor interface for interactive data analysis, as a result of they are doing not emit any output till the job has been dead to completion In several cases, an interactive user would like a fast and dirty approximation over an accurate answer that takes a lot of longer to reason. within the information literature, on-line aggregation has been planned to deal with this drawback, but the batch-oriented nature of ancient Map reduce implementations makes these techniques tough to use

IV. EXPERIMENTAL RESULTS

Define Reducer location and add 2 reducer locations. Defining different reducer at certain location (For location we are taking the latitude and longitude values). After upload documents and upload the input data to our application. Start Map Reduce Aggregation: In the above, the request has been processed by Reducer 1, because the reducer 1 is nearer to the mapper location. Network traffic cost graph:



V. CONCLUSION

In this system, we look into so we will reduce network traffic price for a Map Reduce job by planning a completely unique intermediate information partition theme. Moreover, we jointly consider the individual placement drawback, wherever every individual will scale back incorporated traffic from multiple map tasks. A decomposition-based distributed formula is projected to deal with the large-scale improvement drawback for large information application and an internet algorithm is additionally designed to regulate information partition and aggregation during a dynamic manner. The partition and aggregators facilitate to feature to distance aware routing for process the info for the big knowledge applications. Inserting the aggregators as near the nodes and therefore the consumer would also add to the network traffic reduction and successively helps to reduce the value of the info processing.

REFRENCES

[1]. J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.

[2]. W. Wang, K. Zhu, L. Ying, J. Tan, and L. Zhang, “Map task scheduling in mapreduce with data locality: Throughput and heavy-traffic optimality,” in INFOCOM, 2013 Proceedings IEEE. IEEE, 2013, pp. 1609–1617.

[3]. F. Chen, M. Kodialam, and T. Lakshman, “Joint scheduling of processing and shuffle phases in mapreduce systems,” in INFOCOM, 2012 Proceedings IEEE. IEEE, 2012, pp. 1143–1151.

[4]. 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.

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

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

[7]. 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.

[8]. 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.

[9]. 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.

[10]. 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.