ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 6, Special Issue 12, July 2017

Network Traffic Reduction using Map Reduce

Raju .R.K[1], Shreyas .B[2]

6th sem Students, Department of Computer Science Engineering, City Engineering College, Bangalore, India[1 2]

ABSTRACT: TheMapReduce programming rearranges substantial scale information preparing on thing group by abusing parallel map tasks and reduce tasks. Numerous endeavors have been made to enhance the execution of MapReduce employments, they disregard the network traffic generated in the shuffle phase in the rearrange stage, which assumes a basic part in execution improvement. Customarily, a hash capacity is used to segment the data among the reduce task, which, be that as it may, is not activity proficient in light of the fact that system topology and information estimate connected with every key are not contemplated. In this paper, we study to lessen network traffic cost for a MapReduce work by designing a data partition. Moreover, we together consider the aggregator arrangement issue, where every aggregator can lessen blended movement from numerous guideerrands.

KEYWORDS: Map task, reduce task, data partition , aggregation.

I. INTRODUCTION

MapReduce has risen as the most main stream registering system for huge information handling because of its straightforward programming model and programmed administration of parallel execution. MapReduce and its open source usage Hadoop have been embraced by driving organizations, for example, Hurray!, Google and Facebook, for different huge information applications, and machinelearning bioinformatics cybersecurity. MapReduce isolates a calculation into two primary stages, map task and reduce task. In the guide phase map assignments are dispatched in parallel to change over the first data parts into halfway information in a type of key/worthcombines. These key/esteem matches are put away on neighborhood machine and sorted out into numerous information segments, one for every reduce task. In the reduce stage, each reduce task undertaking gets its own particular share of information segments from all guide undertakings to create the last result. There is a rearrange venture in the middle of guide and lessen stage. In this stride, the information delivered by the map task are requested, delivered and exchanged to the suitable machines executing the reduce stage. The subsequent system movement design from all map tasks to all reduce tasks can bring about an awesome volume of system movement, forcing a genuine limitation on the productivity of information expository application. By default, intermediate data are shuffled according to a hash function in Hadoop, which would lead to large network traffic because it ignores network topology and data size associated with eachkey.

As shown in Fig.1, we consider a toy example with two map tasks and two reduce tasks, where intermediate data of three keys K1, K2, and K3 are denoted by rectangle bars under each machine. If the hash function assigns data of K1 and K3 to reducer 1, and K2 to reducer 2, a large amount of traffic will go through the to

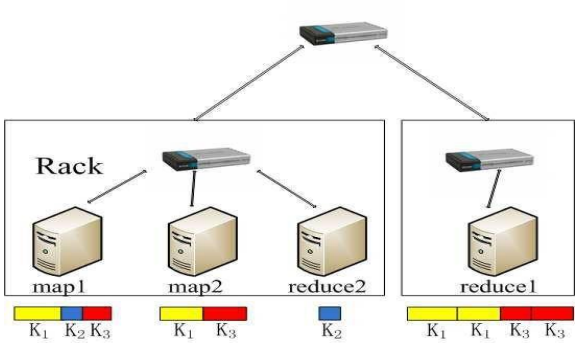
ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

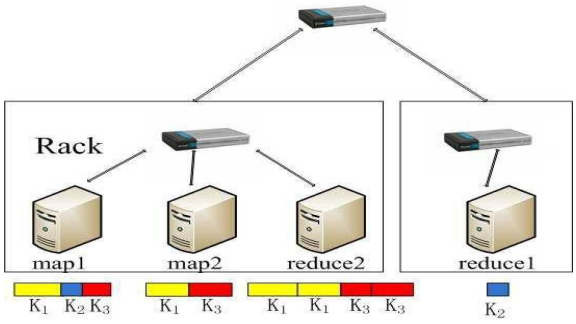
(An ISO 3297: 2007 Certified Organization)

Vol. 6, Special Issue 12, July 2017

switch. To tackle this problem incurred by the traffic-oblivious partition scheme, we take into account of both task locations and data size associated with each key in this paper. By assigning keys with larger data size to reduce tasks closer to map tasks, network traffic can be significantly reduced. In the same example above, if we assign K1 and K3 to reducer 2, and K2 to reducer 1, as shown in Fig. 1(b), the data transferred through the top switch will be significantly reduced.



(a) Hash Partition



(b) Traffic aware partition

In this paper, we mutually consider data partition and aggregation for a Map Reduce work with a target that is to minimize the aggregate system movement. Specifically, we propose a conveyed calculation for huge information applications by declining the first vast scale issue into a few sub problems that can be illuminated in parallel. Also, an online calculation is intended to manage the information segment and accumulation in a dynamic way.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 6, Special Issue 12, July 2017

II. EXISTING SYSTEM

Traditionally, a hash function is used to partition intermediate data among reduce tasks, which, however, is not traffic-efficient because network topology and data size associated with each key are not taken into consideration. However, the data retrieval and the security is not considered.

III. LITERATURE SURVEY

In the reference of paper[1], Map Reduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes akey/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. Many real world tasks are expressible in this model, as shown in the paper. 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. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system. Our implementation of Map Reduce runs on a large cluster of commodity machines and is highly scalable: a typical Map Reduce computation processes many terabytes of data on thousands of machines. Programmers find the system easy to use: hundreds of Map Reduceprograms have been implemented and upwards of one thousand Map Reduce jobs are executed on Google's clusters everyday.In the reference of paper[2], It presents Purlieus, a Map Reduce resource allocation system aimed at enhancing the performanceof Map Reduce jobs in the cloud. Arrangements virtual Map Reduce bunches in a region mindful way empowering Map Reduce virtual machines (VMs) access to info information and critically, middle of the road information from nearby or close-by physical machines. We show how this territory mindfulness amid both guide and diminish periods of the occupation enhances runtime execution of individual employments as well as has an extra preferred standpoint of lessening system movement created in the cloud server farm. This is proficient utilizing a novel coupling of, generally free, information and VM position steps. We direct a nitty gritty assessment of Purlieus and show noteworthy funds in system movement and right around half decrease in employment execution times for an assortment of workloads. In the reference of paper[3], Map-Reduce is a programming model that enables easy development of scalable parallel applications to process vast amounts of data on large clusters of commodity machines. Through a simple interface with two functions, map and reduce, this model facilitates parallel implementation of many real-world tasks such as data processing for search engines and machine learning. In any case, this model does not straightforwardly bolster preparing different related heterogeneous datasets. While preparing social information is a typical need, this confinement causes troubles or potentially wastefulness when Guide Lessen is connected on social operations like joins. We improve Map-Reduce into a new model called Map- Reduce-Merge. It adds to Map-Reduce a Merge phase that can efficiently merge data already partitioned and sorted (or hashed) by map and reduce modules. We likewise exhibit this new model can express social variable based math administrators and in addition execute a few join calculations.In the reference of paper[4], Map Reduce is a popular framework for data-intensive distributed computing of batch jobs. To simplify fault tolerance, many implementations of Map Reduce materialize the entire output of each map and reduce task before it can be consumed. In this paper, we propose a changed Map Reduce design that enables information to be pipelined between operators. This augments the Map Reduce programming model past bunch preparing, and can lessen finish times and enhance framework usage for group occupations too. We present a modified version of the Hadoop Map Reduce framework that supports online aggregation, which allows users to see "early returns" from a job as it is being computed. Our Hadoop Online Model (Bounce) likewise underpins nonstop questions, which empower MapReduce projects to be composed for applications, for example, occasion observing and stream handling. Jump holds the adaptation to non-critical failure properties of Hadoop and can run unmodified client characterized MapReduce programs.In the reference of paper[5], Large datasets ("Big Data") are becoming ubiquitous because the potential value in deriving insights from data, across a wide range of business and scientific applications, is increasingly recognized. The data growth has been accompanied by rapid adoption of large, elastic, multi-tenanted computing clusters ("compute clouds"), leading to a virtuous cycle: the scalability of cloud computing makes it possible to analyze ever larger datasets, and the proliferation of Big Data leads to further adoption of cloud computing. In particular, machine

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

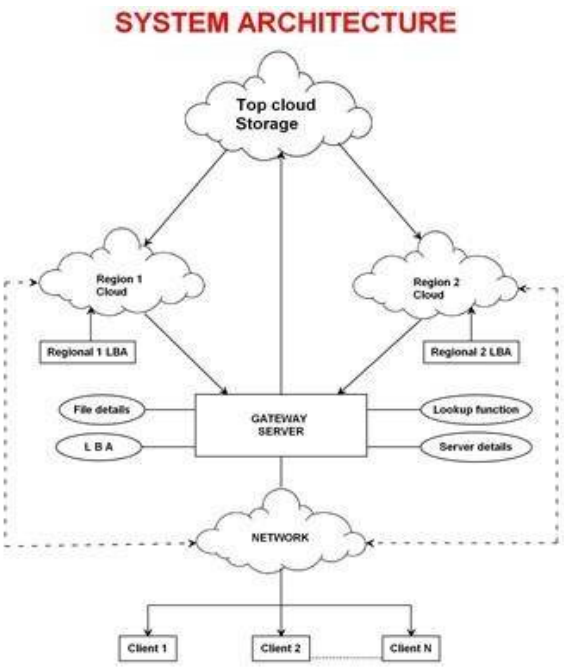
Vol. 6, Special Issue 12, July 2017

learning- one of the foundational disciplines for data analysis, summarization and inference- on Big Data has become routine at most organizations that operate large cloudsusually based on systems such as Hadoop that support the MapReduce programming paradigm. It is now widely recognized that while MapReduce is highly scalable, it suffers from a critical weakness for machine learning: it does not support iteration. Consequently, one has to program around this limitation, leading to fragile, inefficient code. Further, reliance on the programmer is inherently awed in a multi-tenanted cloud environment, since the programmer does not have visibility into the state of the system when his or her program executes. Prior work has sought to address this problem by either developing specialized systems aimed at stylized applications, or by augmenting MapReduce with ad hoc support for saving state across iterations (driven by an external loop). In this paper, we advocate support for looping as a first-class construct, and propose an extension of the MapReduce programming paradigm called Iterative MapReduce.

IV. PROPOSED SYSTEM

In this paper it jointly considers data partition and aggregation for a MapReduce job with an objective that is to minimize the total network traffic. In particular we propose a distributed algorithm for big data applications by decomposing the original large-scale problem into several sub problems that can be solved in parallel. Moreover, an online algorithm is designed to deal with the data partition and aggregation in a dynamic manner. It Reduces network traffic cost in both offline and online cases. Fault tolerance is increased Clients register to the cloud. Clients login to the cloud using the username and password. Client uploads the file .The file gets uploaded.
CloudloginUsername:-admin Password:-admin

Hash code is generated to the uploaded file. The client will login with the username and password.
Username:-user Password:-user
The client can download the file from regional cloud and the transactions gets updated.



(c) System Architecture

The system involves the top cloud, regional cloud 1, regional cloud 2 and clients. The client produces the files. The client uploads the file to the cloud. It provides download services if the client requires. User has to select the file from

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 6, Special Issue 12, July 2017

system and click the upload option. Files will get divided into small blocks (500 bytes each block). Generate hash tag for all block. Compare generated hash block with existing hash tag from database if hash tag matched in that case we will not upload that block into cloud, we will increase number of instance of that block in database table. If hash tag not matched in that case we will add that block hash details in database and upload that block in cloud. LBA - LogicalBlockAddressing technique is used to identify what are the blocks are present in a file. User has to select a file to download. Request Send to the Server, Server has to fetch the IP address of the client system and using Look-Up function identify the IP address Region, let it be Selected Region (SR). Each Region has one Storage Space and One Map Area. In Storage space all the downloaded blocks data are available and in Map area downloaded block details are available. Using LBA server has to find block numbers which are in selected file. Server has to check in Map area of SR whether all the blocks required for the file is available, if all the blocks are available in SR storage space download and merge the blocks and give it to the user. If few blocks are available and few are missing then get the missing blocks from Top Region and place it in SR storage space and update the SR Map details. User can see all its transactiondetails.

V. CONCLUSION

In this paper, we study the joint optimization of intermediate data partition and aggregation in MapReduce to minimize network traffic cost for big data applications. We propose two tasks i.e., map task and reduce task. To deal with the large- scale formulation due to big data, we design a data partition to solve the problem on multiple machines. Furthermore, we extend to handle the MapReduce job in aggregation of same data and merge to produce the final results. We conduct extensive results under the online cases. The results demonstrate that our proposals can effectively reduce network traffic cost under various network settings.

REFERENCES

[1] "Mapreduce: simplified data processing on large clusters"- J. Dean and S.Ghemawat.
[2] "Purlieus: locality aware resource allocation for mapreducein a cloud"- B. Palanisamy, A. Singh, L. Liu, and B.Jain.
[3] "Mapreduce-merge: simplified relational data processing on large clusters-H.-c. Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker.
[4] "Iterative mapreduce for large scale machine learning"- J. Rosen, N. Polyzotis, V. Borkar, Y. Bu, M. J. Carey, M. Weimer,T. Condie, andR. Ramakrishnan.
[5] "Mapreduce online."- T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R.Sears.