International Conference on Information Engineering, Management and Security
2015 [ICIEMS 2015]

ISBN	978-81-929742-7-9
Website	www.iciems.in
Received	10 - July - 2015
Article ID	ICIEMS054

VOL	01
eMail	iciems@asdf.res.in
Accepted	31- July - 2015
eAID	ICIEMS.2015.054

ENHANCING MAP-REDUCE JOB EXECUTION ON GEODISTRIBUTED DATA ACROSS DATACENTERS

JEBILLA P[1], Dr P JAYASHREE[2]
[1]M.E Computer Science & Engineering
[2]Associate Professor, Dept of Computer Science and Technology
[1,2] Anna University-MIT Campus.

Abstract: *In Recent era the size of the data set we need to handle grows rapidly. Efficiently analyzing Big data has always been an issue in our current era. Cloud computing along with the implementations of MapReduce framework provides a parallel processing model and associated implementation to process huge amount of data. In cloud, in many scenarios the input data set are geographically distributed across datacenters. This paper deals with enhancing the MapReduce job execution on the geo distributed data. Possible execution paths are analyzed. A Data Transformation Graph is used to determine the schedules for the job sequences which are optimized using the Shortest Path Algorithm. The proposed model deals with the extending of the Existing Dijkstra's Algorithm to consider the node weight in addition to the edge weight. Ozzie workflow and mapper side joins are used to reduce the execution time and cost. As shown by the comparisons the execution time of the MapReduce job execution has been enhanced.*

Keywords: cloud computing, data center, map reduce, geodistribution

I. INTRODUCTION

Analyzing Big data is so complex that the traditional data processing applications are inadequate. We face many challenges when we need to analyze, capture, search, storage, sharing, transfer the large data sets. The data sets inherently arise due to the applications generating and retaining more information to improve operation and monitoring. Many applications such as social networking sites generates increasing amount of data. MapReduce framework[2] adopted by Apache Hadoop [3]has become a part of the standard toolkit for processing the large set of data sets using the cloud services provided by various cloud vendors.

A. Geodistribution

In today's hyper connected world more and more data is being created every day. Users have started to look beyond the illusion that the resources are omnipresent. More and more applications relying on cloud platforms are geodistributed. Data is stored near the respective sources which can be distributed for frequent access. Data can also be replicated across data centers for availability to limit the cost of costly updates. But how to analyze these data sets more efficiently is a major problem. Map Reduce is used to handle these datasets. These subdata sets can handled by gathering them into a single datacentre or by executing instance of MapReduce job separately on each subdata set in respective datacenters and then aggregate the results or to perform the MapReduce jobs as a single geodistributed where mappers and reducers may be deployed in different data centers. MapReduce operations perform poorly as they were not designed to work on multiple data centers [4][5]. MapReduce executing on geodistributed dataset may follow any combinations of the above options. There can be many execution paths for performing a MapReduce job on a geodistributed dataset and performance can differ.

II. BACKGROUND AND RELATED WORKS

A. MAPREDUCE FRAMEWORK

MapReduce framework such as Apache Hadoop have become a part of the standard toolkit for processing large dataset using cloud resources, and are provided by most cloud vendors. MapReduce works by dividing input data into chunks and process these in a series of parallelizable steps. Sequence of Mapreduce jobs are executed on a given input by applying first job on the given data and then the second job on the output of the fir st job. In many scenarios distinct MapReduce jobs are executed in sequence rather than to perform the job iteratively. When performing a sequence of MapReduce jobs the number of possible execution path increases.

A simple MapReduce program consists of two functions as Map and Reduce such as,

Map <key1; val1> -> list(<key2; val2>)

Reduce <key2; list(val2)> -> ist(val3)

The map function takes the input data and it outputs a set of <key, value> pair. The reducer accepts the set as input and emits set of values .Execution one of this function is a Phase. A MapReduce job includes map and reduce function, input data. The execution of a single job can include any number of mappers and reducers. The input files divided into chunks are inputted to the mapper. These are known as the splits. A partitioning function is used to assign the mapper phase output to the reducer. The Mapreduce framework parallelizes the execution of all functions and ensures fault tolerance.

B. APACHE HADOOP

Apache Hadoop is an open source software framework for distributed storage and distributed processing of very large datasets. The core part of Hadoop consists of a storage called Hadoop Distributed File System (HDFS) and a processing part using MapReduce. Hadoop splits the files into large blocks and distributes these block among the nodes in the cluster and the MapReduce transfers the code to the node to process the data. Hadoop is bundled with the HDFS which is used to store the input of the map phase and also the output of the reduce phase. However, it does not store the intermediate results which are the output of map phase. They are stored on the individual local file systems of the nodes. The Hadoop follows a master-slave model where the master is responsible for accepting jobs dividing those into tasks that includes mappers and reducers and to assign those tasks to slave worker nodes. The hadoop cluster has a single namenode plus a cluster of datanodes. The namenode is the center piece of the HDFS file system which keeps the directory tree of all files in the filesystem and also tracks where across the cluster the data is present. Datanodes are used to store the data and there are number of datanodes. The data is replicated among the datanodes. Above this filesystem comes the MapReduce engine which consists of Job Tracker, to which the client submit the mapreduce jobs. The job tracker service within the hadoop that farms out the mapreduce tasks to specific nodes in the cluster, ideally to the nodes that have the same data. Each worker node has a Task Tracker that processes the assigned tasks. A heartbeat from Task tracker is sent to Job Tracker periodically to update its status.

C. HDFS

The Hadoop Distributed File System is a distributed file system which has been designed to run on commodity hardware. It is designed to be highly fault tolerance and can be deployed on low cost hardware and is also designed to store huge amount of data and it provides high throughput access to the data. HDFS provides fast and scalable access to the information loaded on the clusters and it stores data reliably as it has replication of data across clusters.

D. PREVIOUSLY PROPOSED EXTENSIONS

Numerous efforts have been proposed to improve the efficiency of the execution of the MapReduce jobs. Yang et al.[8] introduced an extra MapReduce phase known as the merge which works after the map and the reduce phase and also extends the MapReduce model for heterogeneous data. Chang et al. [9] Introduced approximation algorithm that was used to order the MapReduce jobs to minimize the overall job completion time. Zaharia et al. [6] worked on to improve the performance of the Hadoop by making it aware of the heterogeneity of the network. MapReduce Online [7] made some modification in MapReduce which allows the MapReduce components to start executing before the data is fully materialized. It allows the reducers to start its process before the complete output of the mapper is available. Amrith Dhananjayan et al. [16] presented an application of the Lyapuno stability theory for load balancing Data Center Networks modeled as discrete event systems. Gaochao Xu et al. [17] introduces a load balance model for the public cloud based on the cloud partitioning concept with a switch mechanism to choose different strategies for different situations. Jehn-Ruey Jiang et al.[11] proposed to extend the well known Dijkstra's shortest path algorithm to consider not only the edge weight but also the node weight in the SDN networks. David B. Wilson et.al [12] described a new forward-backward variant of Dijkstra's and Single source shortest path which allows some of the edges to be scanned backward .

II. MODEL

Sequences of operations are performed on the geodistributed data center. The MapReduce jobs J1; J2; . . . ; Jm give rise to 2 x m operations as each MapReduce job consists of two phase map and reduce. The state of the data before a phase is identified as a stage which starts with 0. The input data is in stage 0 and the final output after the m MapReduce jobs are done it is stage 2m. To move a data from stage s to s + 1, a MapReduce phase is applied to the data partitions and the same amount of data partitions are produced as output.

A. G-MR Overview

G-MR is a hadoop based framework system which is designed to efficiently process the geodistributed data sets and can efficiently perform a sequence of MapReduce jobs on the geodistributed data across data centers. The G-MR employs a Data Transformation Graph (DTG) that is used to determine the optimized execution path for performing sequence of MapReduce jobs. The DTG is used to optimize the execution time of the MapReduce jobs. The optimized execution path obtained may be different from the optimum execution path.

There are generally three main execution paths for performing the MapReduce job on the geodistributed data which are identified as COPY, GEO and MULTI. The COPY execution path is identified as copying all the input data to a single data center prior to the MapReduce job execution. When individual MapReduce jobs are executed on each data center on the corresponding input and the output is aggregated to produce the final result is the MULTI execution path. The other option is to perform the MapReduce job as a single geodistributed operation with mappers and reducers which are distributed across the datacenters. Determining an optimized execution path for a given scenario is not a straightforward process and it becomes even more difficult when sequence of MapReduce job is involved.

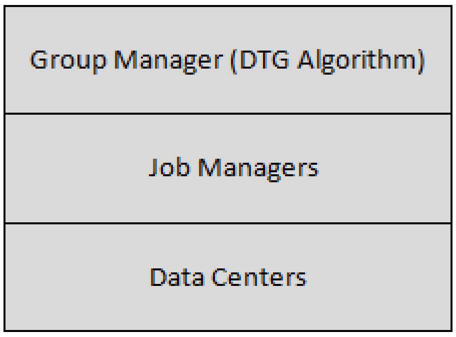


Fig. 1. Architecture of G-MR.

B. ARCHITECTURE AND EXECUTION

G-MR consists of a single component named Group Manger and many components named as Job Manager which is deployed on each participating datacenters. The Group manager determines the optimized execution path while the job Manager manages the MapReduce job that has to be performed within the corresponding datacenter. Fig 1 overviews the architecture of G-MR. The Group Manager executes the DTG algorithm to determine the optimized execution path. The Group Manager informs the corresponding Job Managers about the MapReduce jobs it has to perform in the corresponding datacenters and also regarding the subset of the data on which the job has to be executed.

C. DTG Algorithm

The DTG Algorithm involves constructing a Data Transformation Graph which represents the possible execution paths for performing the MapReduce jobs on the given input dataset. In the existing G-MR system the DTG graph constructed has the edge weight that denotes either the execution time or the cost. The execution time is taken into account. A given node in the graph denotes number of MapReduce phases that has been applied on the input data and its location. The DTG algorithm was used to determine the optimized solution in terms of either the execution time or the cost which involved both the maintaining the node and for transferring the data. Edge between nodes denotes the data copy operations. A node in the graph is described as NsΔ, where s is the number of MapReduce operations in the sequence it is applied and Δ describes the current distribution of data. dk denotes that Psk is located in data center DCdk .Fig 2 shows a simple DTG for a sequence of two mapreduce jobs. Each MapReduce job is represented by three stages in a DTG numbered from s=0 to 2 where s=0 denotes data prior to map phase, s=1 denotes data after map phase and prior to reduce phase and s=2 denotes data after reduce phase. The Execution path is defined as the path from the starting node to the ending node in the DTG. The Data Transformation graph is constructed first based on which the optimized way for executing the MapReduce jobs are determined. This was done using the well know Dijkstra's algorithm [10]. We have used the extended Dijkstra's Algorithm to find the optimized execution path. The extended Dijkstra's algorithm uses the node weight in addition to the edge weight. The node weight denotes the activeness of the node. It could either denote the time or the cost as the edge weight. Based on the activeness of the node, each node is assigned a node weight. The node weight is included only for the outgoing edges in the intermediate nodes and not on the starting and the ending node. This helps in finding the most optimized execution path to perform sequences of mapreduce jobs.

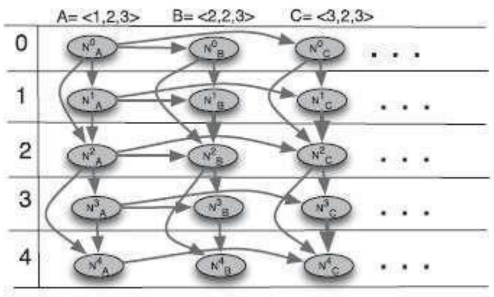


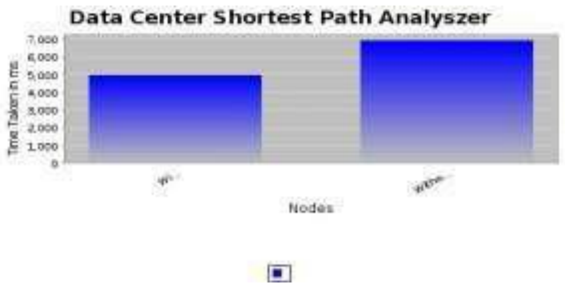
Fig. 2. DTG for a sequence of two MapReduce jobs.

D. Optimizing MapReduce Operation

We have also used the oozie workflow scheduler to manage the MapReduce jobs. It is a server based workflow engine specialized in running the workflow jobs. This Oozie workflow jobs are collection of actions grouped in the form of Directed Acyclic Graph (DAG). The workflow actions start the job in remote systems, which upon completion will call back oozie to notify the action completion and to proceed with the next action. Oozie workflow contains control flow nodes (start, end, decision, fork, join, kill) and action nodes (map-reduce, pig, etc). The control flow nodes define the start and the end of the workflow and Action nodes provides a mechanism to trigger the execution of the processing task. Oozie can identify the completion of the processing task using polling and callback functions. Each task is provided with a callback URL which has to be invoked by the task at the time of the completion else Oozie polls the processing task for completion if failure happens. MapReduce paradigm needs to analyse massive amount of data. Applications of such type need to process large number of data sets which in turn needs to perform several join operations. Query evaluation is a part of the analysis of the large datasets. Joins is most important form of query. Algorithms have been broken into two categories- Two Ways Join and Multi- way joins [13]. A two-way join is performed on the given two datasets P and Q which is defined as a combination of tuples p and q such that p.a = q.b where a and b are values in column P and Q. The multi-way join is defined as a combination of tuplesof n number of datasets. The Reduce side joins looks like a natural way to join the datasets which uses the built-in framework to sort the intermediate keys before they reach the reducer. But this is very time consuming. Hadoop offers another way to join datasets before they reach the mapper. We have used the map-side join [14] to join the datasets to perform the MapReduce jobs. Here the sorting order and the number of partitions must be identical in all the datasets to be joined.

IV. RESULT AND COMPARITIVE ANALYSIS

Big data analysis which is a major issue is overcome by using the Hadoop framework and the MapReduce framework. The Data Transformation graph was introduced to find the all possible execution paths from the source to the destination. The existing Dijkstra's algorithm was used to find the optimized execution path previously. Here we have considered a single datacenter for our implementation. We have extended the Dijkstra's algorithm to include the node weight in addition to the edge weight. The edge weight corresponds to the execution time and the node weight corresponds to the time based on the activeness of the node through which the data is obtained. We also have implemented the map join algorithm to combine the datasets obtained from various datacenters before the mapper phase which helps in reducing the execution time. The execution time of the mapreduce jobs has been enhanced.



V. CONCLUSION

We have modeled a system which enhances the execution of the map reduce jobs which are executed on the geo distributes data across the datacenters. A G-MR system, a MapReduce framework which can efficiently execute a sequence of mapreduce jobs on the geo distributed data relies on the DTG to perform the sequence of job minimizing the execution time. The extended Dijkstra's algorithm is used to enhance the performance of the DTG in the execution of the sequence of the mapreduce jobs.

REFERENCES

[1] Chamikara Jayalath, Julian Stephen, and Patrick Eugster "From the cloud to the Atmosphere: Running MapReduce across data centers" IEEE TRANSACTIONS ON COMPUTERS, VOL. 63, NO. 1, JANUARY 2014.
[2] J.Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Proc. Sixth Conf. Symp. Operating Systems Design and Implementation (OSDI), 2004.
[3] "Hadoop," Apache Software Foundation, 13.
[4] "Hadoop Across Data-Centers,
" http://mailarchives.apache.org/mod_mbox/hadoopgeneral/2010 01.mbox//%3C4B4B6AC7.6020801@lifeless.net%3 E, 2013.
[5] Jairam Chandar 'Join Algorithms using Map/Reduce'
[6] M. Zaharia, A. Konwinski, A.D. Joseph, R.H. Katz, and I. Stoica, "Improving MapReduce Performance in Heterogeneous Environments," Proc. Eighth USENIX Conf. Operating Systems Design and Implementation (OSDI), 2008.
[7] T. Condie, N. Conway, P. Alvaro, J. Hellerstein, K. Elmeleegy, and R. Sears, "MapReduce Online," Proc. Seventh USENIX Conf. Networked Systems Design and Implementation (NSDI), 2010.
[8] H.-C. Yang, A. Dasdan, R.-L. Hsiao, and D.S. Parker, "Map-Reduce-Merge: Simplified Relational Data Processing on Large Clusters," Proc. ACM SIGMOD Int'l Conf. Management of Data,2007.

[9] H. Chang, M. Kodialam, R. Kompella, T. Lakshman, M. Lee, and S. Mukherjee, "Scheduling in MapReduce-like Systems for Fast Completion Time," Proc. IEEE INFOCOM, 2011.
[10] E.W. Dijkstra, "A Note on Two Problems in Connection with Graphs," Numerische Math., vol. 11, pp. 1-269, 1959.
[11] Jehn-Ruey Jiang, Hsin-Wen Huang, Ji-Hau Liao, and Szu-Yuan Chen "Extending Dijkstra's Shortest Path Algorithm for Software Defined Networking
[12] David B. Wilson, Uri Zwick "A forwardbackward single-source shortest paths algorithm" 2013 IEEE 54th Annual Symposium on Foundations of Computer Science
[13] Jairam Chandar "Join Algorithms using Map/Reduce"
[14] A. Pigul "Comparative Study Parallel Join Algorithms for MapReduce environment" Saint Petersburg State University.
[15] Chung-Cheng Li, Kuochen Wang, 2014 'An SLA aware Load Balancing Scheme for Cloud Data centers' Information Networking (ICOIN), International Conference.pp 58-63
[16] Amrith Dhananjayan1, Kiam Tian Seow1 and Chuan Heng Foh2 , 2013 'Lyapunov Stability Analysis of Load Balancing in Datacenter Networks' Globecom Workshop - The 5th IEEE International Workshop on Management of Emerging Networks
and Services.pp 912-916.
[17] Gaochao Xu, Junjie Pang, and Xiaodong Fu, February 2013 'A Load Balancing Model Based on Cloud Partitioning for the Public Cloud' TSINGHUA SCIENCE AND TECHNOLOGY ISSN 1 007 - 02 14 04/12 Volume 18, Number 1. pp 34-39.