International Journal of Engineering Research
Volume No.3, Issue No.12, pp : 745-748

ISSN:2319-6890)(online),2347-5013(print)
01 Dec. 2014

Implementation of Parallelization Contract Mechanism Extension of Map Reduce Framework for the Efficient Execution Time over Geo-Distributed Dataset

Ms. Kirtimalini N., Prof.T.A.Chavan
kirtikakade20@gmail.com, tachavan@rediffmail.com

Abstract-The world is surrounded by technology and Internet with extreme dynamic changes day by day in it.Due to that quintillion bytes of data is created. Source of this data is in the form of petabytes and zettabytes,which is known as Big data.Examples of such data are climate information, trajectory information, transaction records, web site usage data etc .As this data is in the abundant form so that not easy to process and require more time to execute.Hadoop is only scalable that is it can reliably store and process petabytes. Hadoop plays an important role in processing and handling big data It includes MapReduce – offline computing engine, HDFS – Hadoop Distributed file system, HBase – online data access.Map Reduce functions as dividing input files into chunks and processing these in a series of parallelizable steps., mapping and reducing constitute the essential phases for a Map Reduce job. As this freamework provides solution for large data nodes by providing distributed environment. Moving all input data to a single datacenter before processing the data is expensive. Hence we concentrate on geographical distribution of geo-distributed data for sequential execution of map reduce jobs to optimize the execution time. But it is observed from various results that mapping and reducing function is not sufficient for all type of data processing. The fixed execution strategy of map reduce program is not optimal for many task as it does not know about the behavior of the functions. Thus, to overcome these issues, we are enhancing our proposed work with parallelization contracts. These contracts help to capture a reasonable amount of semantics for executing any type of task with reduced time consumption. The parallelization contracts include input and output contract which includes the constraints and functions of data execution

The main aim of this paper is to discuss various known Map reduce technology techniques available for geodistributed data sets by using different techniques. Further, the paper also discloses the implementation of these techniques, and comparision results of this method with the existing systems. Future trends including use of query optimizing techniques to improve the results of the query as well as reduce the cost for the computation. To achieve this we use the indexing mechanism to the cache system to preserve the query search results.

*Keywords:***MaReduce,PACT,big data,Geodistributed data sets,hadoop**

1. Introduction

Hadoop is a free, Java-based programming framework that supports the processing of large data sets in a distributed computing environment. It is part of the Apache project sponsored by the Apache Software Foundation. Hadoop makes it possible to run applications on systems with thousands of nodes involving thousands of terabytes. Its distributed file system facilitates rapid data transfer rates among nodes and allows the system to continue operating uninterrupted in case of a node failure. This approach lowers the risk of catastrophic system failure, even if a significant number of nodes become inoperative. Hadoop was inspired by Google's MapReduce, a software framework in which an application is broken down into numerous small parts. Any of these parts (also called fragments or blocks) can be run on any node in the cluster. Doug Cutting, Hadoop's creator, named the framework after his child's stuffed toy elephant. The current Apache Hadoop ecosystem consists of the Hadoop kernel, MapReduce, the Hadoop distributed file system (HDFS). The Hadoop framework is used by major players including Google, Yahoo and IBM, largely for applications involving search engines and advertising. The preferred operating systems are Windows and Linux but Hadoop can also work with BSD and OS X.

In an internet era million of users today uses various computer applications and Internet services. The sheer volume of data that these services work with has led to interest in parallel processing on commodity clusters. For this the best example is Google, which uses its MapReduce framework to process 20 petabytes of data per day. These services generate clickstream data from millions of users every day, which is a potential gold mine for understanding access patterns and increasing ad revenue Other Internet services, such as e-commerce websites and social networks,also cope with enormous volumes of data.. Furthermore,for each user action, a web application generates one or two orders of magnitude more data in system logs, which are the main resource that developers and operators have for diagnosing problems in production.

2.Hadoop Distributed File System (HDFS)

Hadoop Distributed File System (HDFS) serves as the large scale data storage system. Similar to other common file systems, the HDFS supports hierarchical file organization. The NameNode splits large files into fixed sized data blocks which are scattered across the cluster. Typically the data block size for the HDFS is configured as 128MB, but it can be configured by file system clients as per usage requirements. The data storage

International Journal of Engineering Research ISSN:2319-6890)(online),2347-5013(print)
Volume No.3, Issue No.12, pp : 745-748 01 Dec. 2014

is of type write once/read many (WORM) and once written, the files can only be appended and cannot be modified to maintain data coherency. Since HDFS is built on commodity hardware, the machine failure rate is high. In order to make the system fault tolerant, data blocks are replicated across multiple DataNodes. HDFS provides replication, fault detection and automatic data block recovery to maintain seamless storage access. By default replication takes place on three nodes across the cluster . When a client tries to access the failed DataNode, the NameNode maps the block replica and returns it to the client. For achieving the high throughput, the file system nodes are connected by high bandwidth network.

3.**NameNode**

The NameNode maintains the file system metadata as the HDFS directory tree and operates as a centralized service in the cluster. It controls the mapping between file name, data block locations and the DataNodes on which data blocks are stored. It also writes the transaction logs to record modifications in the file system. Clients communicate with the NameNode for common file system operations such as open, close, rename and delete.

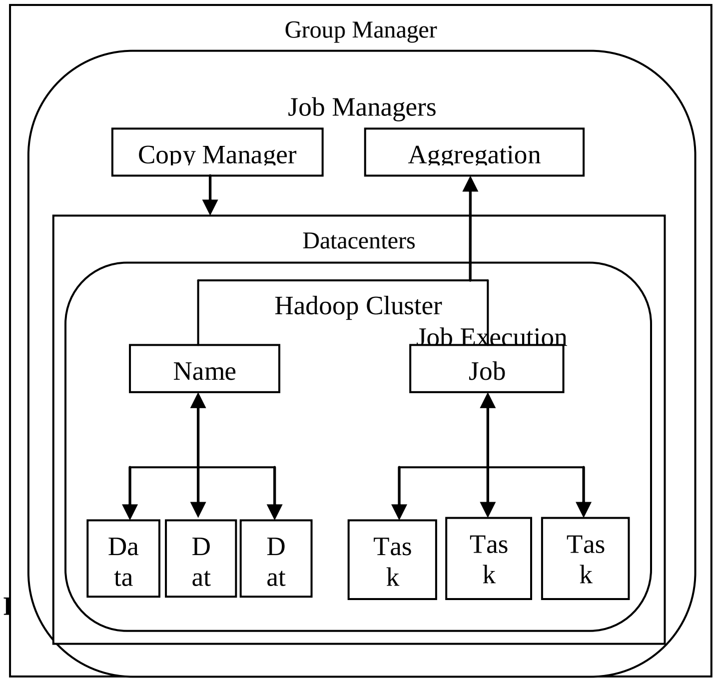
4.Namespace

The namespace is a live record of the HDFS located on the centralized NameNode server. It is a directory tree structure of the file system which documents various aspects of the 8 HDFS such as block locations, replication factor, load balancing, client access rights and file information. The namespace serves as a mapping for data location and helps HDFS clients to perform file system operations. The metadata is stored as a file system image (fsimage) [7] file which is a persistent checkpoint of the file system. The edit log records the write operations submitted by the file system clients. When the edit log size exceeds a predefined threshold, the NameNode moves the transactions into live memory and apply each operation to the fsimage. A backup of the namespace is periodically stored on the local disk of the NameNode and synchronized with a secondary master node as a provision against NameNode failure. When the NameNode reboots, it collects the file system namespace from the local copy.

5.DataNode

A DataNode is a storage server that accepts read/write requests from the NameNode. DataNodes store data blocks for local or remote clients of HDFS. Each data block is saved as a separate file in the local file system of the DataNode. The DataNode also performs block creation, deletion and replication as a part of file system operations. For keeping the records up-to-date, the DataNode periodically reports all of its data block information to the NameNode. 9 DataNode instances can talk to each other for data replication. To maintain its live status in the cluster, it periodically sends heartbeat signals to the NameNode. When the NameNode fails to receive heartbeat signals from the DataNode, it is marked as a dead node in the cluster

System Architecture



A Geo- distributed data called census data is taken as input for processing. The U.S. Census Bureau produces and publishes estimates of the population for each state and county, as well as the nation as a whole. We utilize administrative data from a number of sources to estimate 1) the change in population since the most recent decennial census, and 2) the population for each year since the most recent decennial census. With each annual release of population estimates, the entire time series of estimates beginning on April 1, 2010 is revised and updated. This census dataset is used further. The dataset is analyzed and preprocessed to remove noise and unwanted fields from the dataset before applying into hadoop for execution. Noise data like invalid data, null values, etc are identified and removed from the dataset. Our target is fixed for job execution and based on the target(output) needed, fields will be selected for processing.

- Noise Removal

Noise removal should be done to avoid redundancy.Hadoop Distributed File System (HDFS) serves as the large scale data storage system. Similar to other common file systems, the HDFS supports hierarchical file organization. The NameNode splits large files into fixed sized data blocks which are scattered across the cluster. Typically the data block size for the HDFS is configured as 128MB, but it can be configured by file system clients as per usage requirements. Hadoop Distributed File System (DFS) will be configured for uploading the preprocessed geo data into hadoop. The configuration includes setting VM (hadoop platform) IP and port for connection. FSDataInputStream and FSDataOutputStreamis used to upload and download data from hadoop. Different datacenters are analysed for data execution across different datacenters.

- Job Execution

International Journal of Engineering Research
Volume No.3, Issue No.12, pp : 745-748
ISSN:2319-6890)(online),2347-5013(print)
01 Dec. 2014

The GroupManager executes the DTG algorithm (Novel Algorithm). This group manager is executed for Determining path for performing the MapReduce job. After this the GroupManager starts to execute the job in determined optimized Multi-execution path based on the parallelization contract(PACT). Input contracts is defined based on the first order function which is the target that we need to achieve using the census dataset and mapreduce framework. The input contract includes the properties that we need to implement the second order function. The second order function is the function that we need to define based on the output data. Executing individual MapReduce jobs in each datacenter on corresponding inputs and then aggregating results is defined as MULTI execution path. This path and PACT programming is used to execute the jobs effectively.

- Executing Jobs

JobManagers perform the jobs accordingly using the HadoopMapReduce clusters deployed in corresponding datacenters. The Group-Manager may also instruct a JobManager to copy data to a remote datacenter or aggregate multiple sub-datasets copied from two or more remote datacenters. The effective job execution can be achieved usi 1. the G-MR and PACT Model. The job tracker forwards the ta to task tracker for execution and the results from the task track is received and combined to get aggregative result.

6.Related Work And Comparative Analysis

Volley, a system that addresses these challenges. Cloud services make use of Volley by submitting logs of datacenter requests. Volley analyzes the logs using an iterative optimization algorithm based on data access patterns and client locations, and outputs migration recommendations back to the cloud service. To utilize Volley, applications have to log information on the requests they process. These logs must enable correlating requests into “call trees” or “runtime paths” that capture the logical flow of control across components, as in Pinpoint or X-Trace . Volley takes approximately 14 hours to run through one month’s worth of log files. We first map each client to a set of geographic coordinates using the commercial geo-location database mentioned earlier. Iteratively Move Data to Reduce Latency. Volley iteratively moves data items closer to both clients and to the other data items that they communicate with. This iterative update step incorporates two earlier ideas: a weighted spring model as in Vivaldi and spherical coordinates as in Htrae . After computing a nearly ideal placement of the data items on the surface of the earth, we have to modify this placement so that the data items are located in datacenters, and the set of items in each datacenter satisfies its capacity constraints. Like Phase 2, this is done iteratively: initially, every data item is mapped to its closest datacenter. For datacenters that are over their capacity, Volley identifies the items that experience the fewest accesses, and moves all of them to the next closest datacenter. Because this may still exceed the total capacity of some datacenter due to new additions, Volley repeats the process until no datacenter is over capacity.

HOG: Distributed HadoopMapReduce on the GridMapReduce is a framework pioneered by Google for processing large amounts of data in a distributed environment. Hadoop is the open source implementation of the MapReduce framework. Due to the simplicity of its programming model and the run-time tolerance for node failures. The architecture of HOG is comprised of three components. The first is the grid submission and execution component. In this part, the Hadoop worker nodes requests are sent out to the grid and their execution is managed. The second major component is the Hadoop distributed file system (HDFS) that runs across the grid. And the third component is the MapReduce framework that executes the MapReduce applications across the grid.

7.Implimentation of PACT

The PACT (parallelization contract) programming model is a generalization of the well-known map/reduce programming model, extending it with further second-order functions, as well as with Output Contracts that give guarantees about the behavior of a function. PACTs enable several types of optimizations on the data flow during query transformation and even during query execution. PACTs are designed to be as generic as (and compatible to) map/reduce, while overcoming several of map/reduce’s major weaknesses:

The functions map and reduce alone are not sufficient to express many data processing tasks both naturally and efficiently. Map/reduce ties a program to a single fixed execution strategy, which is robust but highly suboptimal for many complex tasks.

Map/reduce makes no assumptions about the behavior of the functions. Hence, it offers only very limited optimization opportunities.

There are four different components on the architectural, refer to the following sections:

1.**Cross**operates on multiple inputs of key/value pairs and builds a Cartesian product
over its input sets. Each element of the Cartesian product becomes an independent
subset.

2.**CoGroup**groups each of its multiple inputs along the key. Independent subsets are
built by combining the groups with equal keys of all inputs. Hence, the key/value
pairs of all inputs with the same key are assigned to the same subset.Please refer Fig 2 of data flow diagram

.**Match**operates on multiple inputs. It matches key/value pairs from its input datasets with the same key. Each possible two key/value pairs with equal key form an independent subset. Hence, two pairs of key/value pairs with the same key are processed independently by possibly different instances of the user function, while the CoGroup contract assigns them to the same subset and guarantees to process them together.

8.Comparative Results of existing system and Proposed system

If we use hadoop as a server to analyse big data ,here we have used Ubuntu OS to run virtually through VMWareWorkstation,thenComparitive results are as follows.For testing purpose we have used census information as an input.Refer Fig 3, Fig 4 and Fig 5

International Journal of Engineering Research
Volume No.3, Issue No.12, pp : 745-748

ISSN:2319-6890)(online),2347-5013(print)
01 Dec. 2014

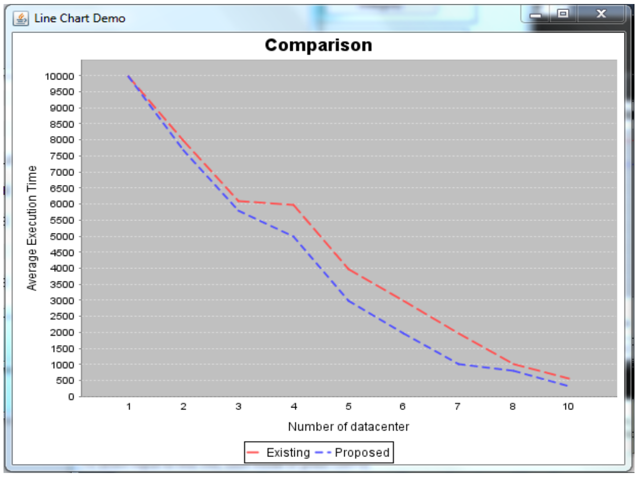


Fig.3 Comparative Results of average execution time across number of data centers

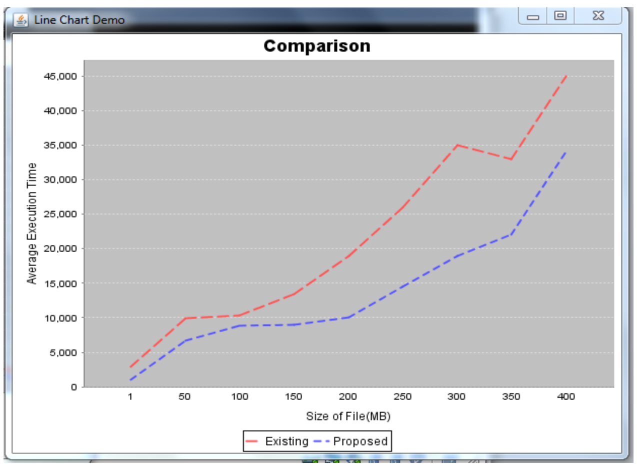


Fig.4 Comparative Results of average execution time across size of files.

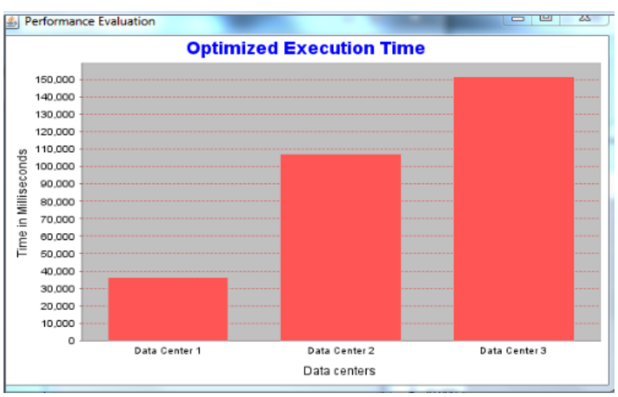


Fig.5 Optimized execution time of data centers

Conclusion

As per the results of implementation of MapReduce and PACT ,it is observed that PACT programming model is more efficient over big data with geographically distributed datasets. The PACT programming model encourages a more modular programming style. Although often more user functions need to be implemented.Data analysis tasks can be expressed as straight-forward data flows. That becomes in particular obvious, if multiple inputs are required. In this way we can say that over the extension of the Map reduce ,PACT programming model is more efficient as it require less execution time for noisy data which is geographically distribted.

References

i. *Alexander Alexandrov, Stephan Ewen, Max Heimel, FabianHueske,Odej Kao, Volker Markl, Erik Nijkamp, Daniel Warneke (2010), MapReduce and PACT - Comparing Data Parallel Programming Model, Technische University at Berlin,Germany.*

ii. *ChamikaraJayalath, Julian Stephen, and Patrick Eugster(2013),From the Cloud to the Atmosphere: Running MapReduce across Data Centers, IEEE transactions on computers, Vol. 63 no. 1.*

iii. *C. Olston, B. Reed, U. Srivastava, R. Kumar, and A.Tomkins(2013), Pig Latin: A Not-so-Foreign Language forData Processing, Proc. ACM SIGMOD Int'l Conf. Management of Data.HadoopAcross Data-Centers.Hadoop: The Definitive Guide, http://oreilly.com/catalog/9780596521981, 2013.*

iv. *H. Chang, M. Kodialam, R. Kompella, T. Lakshman, M. Lee,and S. Mukherjee(2011), Scheduling in MapReduce-like Systems for Fast Completion Time, Proc. IEEE Infocom.M. Zaharia, A. Konwinski, A.D. Joseph, R.H. Katz, and I.Stoica(2008), Improving MapReduce Performance in*

v. *Heterogeneous Environments, Proc. Eighth USENIX Conf.Operating Systems Design and Implementation (OSDI).M. Zaharia, A. Konwinski, A.D. Joseph, R.H. Katz, and I.Stoica(2008), Improving MapReduce Performance inHeterogeneous Environments, Proc. Eighth USENIX Conf.Operating Systems Design and Implementation (OSDI).M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly(2007),Dryad:Distributed Data-Parallel Programs from Sequential Building Blocks, Proc. ACM Second SIGOPS/EuroSys European Conf.Computer Systems (Eurosys)S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H6. Bhogan(2010), Volley: Automated Data Placement for Geo-Distributed Cloud Services, Proc. Seventh USENIX Conf.Networked Systems Design and Implementation (NSDI).T. Condie, N. Conway, P. Alvaro, J.Hellerstein, K. Elmeleegy,and R. Sears (2010), MapReduce Online, Proc. Seventh USENIX Conf. Networked Systems Design and Implementation(NSDI).*

vi. *T. Condie, N. Conway, P. Alvaro, J.Hellerstein, K. Elmeleegy,and R. Sears(2010), MapReduce Online, Proc. Seventh USENIX Conf. Networked Systems Design and Implementation(NSDI).*

vii. *V. Ramasubramanian, T. Rodeheffer, D. Terry, M. Walraed-Sullivan, T. Wobber, C. Marshall, and A. Vahdat(2009),Cimbiosys: A Platform for Content-Based Partial Replication,Proc. Sixth USENIXSymp. Networked Systems Design and Implementation (NSDI).*

viii. *W. Lloyd, M.J. Freedman, M. Kaminsky, and D.G. Andersen(2011), Don't Settle for Eventual: Scalable Causal Consistencyfor Wide- Area Storage with COPS, Proc. ACM 23rd Symp.Operating Systems Principles (SOSP).*