International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-2 Issue-1, March 2013

A Comparative analysis of Data Replication Strategies and Consistency Maintenance in Distributed File Systems

Priya Deshpande, Aniket Bhaise, Prasanna Joeg

***Abstract:** The data plays vital role in data intensive applications or applications which relay on large data files, In the era of information technology the size of data is increasing drastically and this type of data is usually referred to as "Big Data". Which is usually in the unstructured or may be in structured format in data grids or in cluster, and manipulation of such type of data like retrieving, storing and updating it is very tedious job in data intensive application. Data grids are type of data cluster technique which deals with such big data. which may be heterogeneous or homogeneous in nature depending on their property but in the era of fast growing technology the term heterogeneous data grids now replacing by cloud computing to serve as one of the service of cloud computing. In network of cloud computing, data replication and consistency maintenance plays key role to share data between nodes (data intensive applications) to achieve high performance, data availability, consistency and partial tolerance. In this paper we discuss the various data replication strategies with Hadoop Distributed File System which provides MapReduce Framework for data replication and consistency maintenance in cloud computing, to achieve high performance, consistency, availability and partial tolerance and discuss the performance evaluation of these various techniques and frameworks like cloud MapReduce, Integrated data replication and consistency maintenance and also MapReduce with Adaptive Load balancing for Heterogeneous and Load imbalanced cluster (MARLA).*

***Keywords**— Distributed System, Data Intensive Applications, Data Grids, Data Replicas, Job Scheduling, Cloud Computing.*

I. INTRODUCTION

Over the last decade the use of internet has been grown rapidly, where the need of file sharing in peer to peer systems increases. The majority of peer to peer application share file on internet as per their requirement and these files may be audio, video, text file or transactional data.

A study reveals that the utilization of such files are repetitive and manipulated by most of the nodes in the internet such repetitive access or sharing of file may overhead the server or producer node, also delayed the response. Such popular files are called as hot files in web.

Paper submitted on: March 05, 2013 at 03:30 PM

To reduce the overhead on producer node and for high response, file replication is optimal solution to share files on internet or on cluster but if such large number of crowd is accessing these files from various replication sources then majority of difficulty we face is consistency of file and availability, this properties of file replication is called as CAP property [5][6] that is nothing but the file Consistency, Availability and Partial Tolerance. File replication should provide fidelity of file consistency and unnecessary creation of replicas in various nodes, file replication should work in dynamic form that is it also allows peers to update, delete and provide consistent file in minimum partial tolerance. For such situation Author Konstantin Shvachko et al. proposed Hadoop Distributed File System and implemented at Yahoo! With 25 petabytes of data, Hadoop provides the framework called as MapReduce for analysis and transformation of very large data sets in large clusters. Hadoop is an Apache project which comes under open source license, whose actual base is UNIX file system, Hadoop clusters has capability to scale its computation, storage as well as IO capacity by just increasing its horizontal capacity. Here horizontal capacity means increasing the actual number of resources these resources may be nodes, storage devices or commodity servers, there are lots of other vendor who are practicing Hadoop and developed their own file system as some of pig, ZooKeeper, Chukwa of Yahoo! Corporation, hive of Facebook, HBase of Microsoft and MapReduce of Apache group.

As data is generating day by day in very large quantity in the internet, the applications such as data-Intensive scientific applications or data application which relay on large files needs to develop new techniques for data replication for file sharing in data cluster system. This large data is known as "Big Data" which is may in the form of Structured or unstructured data we can also called such data as SQL based data and Non SQL based data respectively. To manipulate and access SQL base data there are some RDBMS data warehouse technology to analyse and access such big data but unstructured data which can't be access by traditional query language and these files and they are large in size, to manipulate such large unstructured data in distributed environment apache Hadoop is the framework which includes many technologies like MapReduce which interact with such big data very effectively and efficiently.

As such big data is unstructured they are also called as "key-value" stores also called as schema-free databases, these schema-free can't maintain complete consistency in across distributed systems or in distributed nodes by the CAP theorem [6][7], out of consistency, availability & partitioning tolerance only two can be achieved, most of the these database are more scalable architecture that relax the consistency requirement which called as eventual consistency.

Revised Manuscript Received on 30 March 2013.

* Correspondence Author
Priya Deshpande, Assistant Professor -MITCOE Pune, India.
Aniket Bhaise, ME-Student, MITCOE, Pune, India.
Prasanna Joeg, Professor MITCOE, Pune, India.

This property of unstructured data allow more scalability in distributed storage system without utilizing any kind of system lock or commit protocols

HBase and HDFS(Hadoop Distributed File system) are well known implementations of such unstructured big data which are successors of GFS(Google file system) which have adopted the MapReduce framework of Apache Hadoop package to manage data replication & consistency[3], Hadoop stores metadata on separate node called as NameNode and application data on other nodes called as DataNodes these all nodes are connected via Data grid protocol based network where files are replicated on DataNodes which ultimately provides dynamism and security of data. Now in cloud environment which is made up of various independent nodes, the major challenge is to provide these data as one service of cloud in distributed system instead of managing single node's resource. cloud OS with MapReduce is use to integrate data replication and consistency maintenance, The major advantage of implementing cloud OS in distributed environment is, it can manage much bigger infrastructure and cloud OS can support thousands of nodes in distributed environment. Most of companies like Amazon, Google, Microsoft and yahoo provide cloud services in which they have embarrassed the eventual consistency at some extend and by using MapReduce in cloud it parallelize the data replication and consistency. Where work is spread out to as many nodes as possible by Map function and then reducer will achieve high performance and scalability of data in distributed environment, the main advantage of cloud is it supports incremental scalability, decentralize in nature and heterogeneous environment of computing nodes. To achieve high performance of data replication and consistency maintenance in heterogeneous data clusters author Zacharia Fadika et al. introduces a framework MARLA [4] (MapReduce with adaptive Load balancing for heterogeneous and Load imbalanced Clusters), as cloud environment is made up of such heterogeneous data clusters in which peers are distinguishes from each other by their processing and storage capacity, This MARALA [4] framework provides dynamic load-balancing MapReduce implementation, integration and advantages of MapReduce model in heterogeneous clusters. In this paper we will study and compare the performance of these three frameworks that is IRM (integrated file replication and consistency maintenance) [2], cloud MapReduce and MARLA (MapReduce with adaptive load balancing for heterogeneous and Load imbalanced clusters) [4].

II. RELATED WORK

Data Replication in distributed system is basically use to release the load in hot spots and decrease the latency, the PAST [8] and CFS [9] are framework replicate the data near to requested site and file owner. In unstructured files system the data replication have average latency but can cause load imbalance because of unnecessary data replication of popular files. Some of nodes which contain replicated data may leave the network where there path from owner and subscriber recorded for update notifications so that availability of replicated data will not be reachable so to maintain availability and consistency of replicated data massage spreading based techniques are used SCOPE [10] and Freenet [11] are implementations of this techniques but still this techniques are centralize in nature. author Haiying shen et al. proposed and implemented centralized integrated file replication and consistency maintenance which were unable to keep track of utilization of replicas and their updates which will may achieve the replication of data but consistency may get jeopardize. In clouds MapReduce environment most of the companies like Amazon, Google, Microsoft and Yahoo provides services of Storage services such as Amazon S3 and Microsoft's Azure blob storage also provide Simple Queue Service for communication in distributed systems [3], Amazon EC2 and Microsoft Azure are services of cloud OS to implement data replication frameworks as cloud have advantage to Scheduling job and data intensive applications over wide-area networks that is it manages large infrastructure as well as service maximum number of users. Cloud also employs horizontal scaling in which some of the node may leave or join network frequently which contain replicated data, in which nodes can act independently and MapReduce will map the operation and then Reducer will optimize the data replication and consistency. MapReduce will work on the top of cloud OS. The Map will distribute the replicas and Reduce will optimise the work in parallel which allows us to achieve data replication and consistency integration in distributed environment [3]. Apache Hadoop and MARINE are the implementation of Cloud MapReduce but they face some challenges in heterogeneous clusters [3]. As it requires more space to share data for more robust fault tolerance, to overcome this problems author Zacharia Fadika et al. proposes MARLA which supports general parallel file system, grid and cloud computing. MARALA relay on dynamic scheduling mechanism for each node's task in which the task parameters and space required will not be predefined. Up till now majority of research work is targeted on job scheduling in data grids as well as in cloud environment but to schedule job in cloud environment data is important aspect which may be on the node where job is scheduled or on some other node, in such case job tracker have to copy the data from remote node to node where the job is scheduled which dominates the optimality of execution of job and increase the latency so the need of data replication is arise. But major problem was how to implement this data replication where the data is in various forms and structures and how to maintain the consistency of data in cloud environment, in cloud system various services are provided such as platform as service (PaaS) infrastructure as service (IaaS) and software as service (SaaS) which deals with various services of cloud and data, but any one type of data is manipulated such as structures or unstructured that is transactional data or sequential files, image file etc. respectively. Data replication is successful in homogeneous data grids in which large scientific data is replicated and manipulated as per requirement and need [12]. But this data replication technique does not ensure the consistency of data rather it appends the updates as the structure of data is uniform in nature in data grid.

III.DATA REPLICATION STRATEGIES

There are basically two types of data replication strategies and also further there are some categories in these two major data replication strategies which are previously explained and implemented by different authors, they are as follows:

A. Replication strategy for Read-only requests

Major assumption in Data Grids is that the data is read-only. There are some data replication strategies for read-only requests.

a) Replica Selection based on Replica location/ User preference

The replicas are selected based on users' preference and replica location. Vazhkudai et al. proposes a strategy that uses Condor's ClassAds (Classified Advertisements) [13] to rank the sites suitability in storage context [14]. The application requiring access to a file presents its requirement to the broker in form of ClassAds. The broker then does the *search*, *match* and *access* of the file that matches the requirements published in the ClassAds. Dynamic replica creation strategies discussed in dynamic replication strategies for high speed data grids [15] are as follows:

(i) Best Client: Each node maintains the record of access history for each replica, i.e. which data item is being accessed by which site. If the access frequency of a replica exceeds a threshold, a replica is created at the requester site.

(ii) Cascading replication: This strategy can be used in the tired architecture discussed above. Instead of replicating the data at the 'best client', the replica is created at the next level on the path of the best client. This strategy evenly distributes the storage space as well as other lower level sites have close proximity to the replica.

(iii) Fast Spread: Fast spread replicates the file in each node along the path of the best client. This is similar to path replication in P2P systems.

Since the storage space is limited, there must be an efficient method to delete the files from the sites. The replacement strategy proposed by Ranganathan et al. deletes the most unpopular files, once the storage space of the node is exhausted. The age of the file at the node is also considered to decide the unpopularity of file.

b) Economy-based replication policies

The basic principle behind economy-based polices are to use the socio-economic concepts of emergent marketplace behaviour, where local optimisation leads to global optimisation. This could be thought as an auction, where each site tries to *buy* a data item to create the replica at its own node and can generate revenue in future by *selling* them to other interested nodes. Various economy-based protocols have been proposed [16][17], which dynamically replicates and deletes the files based on the future return on the investment. Bell et al. uses a reverse auction protocol to determine where the replica should be created.

As the storage space of the site is limited, it must make a choice before replicating a file that whether it is worth to delete an existing file. Thus, the investment decision between purchasing a new file and keeping an old file depends on the change in profit between the two strategies.

c) Cost Estimation based

Cost estimation model [18] is very similar to the economic model. The cost estimation model is driven by the estimation of the data access gains and the maintenance cost of the replica. While the investment measured in Economic models [16][17] are only based on data access, it is more elaborate in the cost estimation model. The cost calculations are based on network latency, bandwidth, replica size, runtime accumulated read/write statistics [18] etc.

B. Replication Strategy for Update request

There is also need in large data grids to update the data replica for such update request there are two types of data replication strategies

a) Synchronous

In synchronous model a replica is modified locally. The replica propagation protocol then synchronizes all other replicas. However, it is possible that other nodes may work on their local replicas. If such a conflict occurs, the job must be redone with the latest replica. This is very similar to the synchronous approach discussed in the distributed DBMS section.

b) Asynchronous

Various consistency levels are proposed for asynchronous replication. Asynchronous replication approaches are discussed as follows [19]:

(i) Possible inconsistent copy (Consistency level: -1): The content of the file is not consistent to two different users. E.g. one user is updating the file while the other is copying it, a typical case of "dirty read problem".

(ii) Consistent file copy (Consistency level: 0): At this consistency level, the data within a given file corresponds to a snapshot of the original file at some point in time.

IV. ARCHITECTURE & METHODOLOGIES

A. IRM: Integrated File Replication & Consistency Maintenance

Author proposes integrate the file replication and consistency maintenance by letting each node work autonomously[2]. Where node will determine the need for file replication and updates the file on the basis of popularity of file and update rate, Fig 1 shows peer to peer file sharing network in which replicas file are shared in various nodes and dashed lines shows update polling.

IRM implement adaptive poling which performs timely update operation on file replicas depending on update frequency of master file and popularity of file by adapting this technique IRM also avoids unnecessary replication of file and updation of underutilized file replicas.

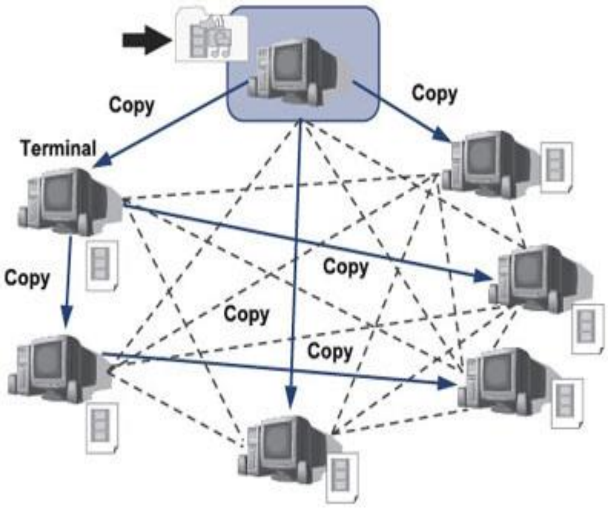

Fig. 1. IRM: File Replication & Consistency Maintenance [2].

When node receives the query for file it copy updated file from master node or from node which contain updated copy of that requested file and poll it for update after some time interval and depending on query rate of that file which ultimately ensures the consistency of file.

a) File Replication

To determine where to replicate file so that it can be fully utilized effectively and efficiently in peer to peer system, there is need to identify the nodes where file will be replicated, some of the nodes may carry more request to replicas while other nodes may not be frequent requesters so depending upon query frequency. And the junction nodes will be the right node where file can be replicated, highly utilized and make them consistent for other peers, for that IRM uses threshold value for file which defines minimum query request frequency, if request query initiation crossed this threshold value then replica is created on requester node same as that of at junction nodes if query passing frequency for file is greater than threshold value then junction node will also copy that file replica.

IRM also checks the file popularity periodically by updating their threshold values if these file replicas are underutilized or their request query initiation rate is low then such replicas are removed from nodes this technique is called it as Replica Adaptation. An Adaptation of request query initiation ensures that all file replicas are worthwhile and there is no overhead of unnecessary consistency maintenance of file.

b) File Consistency Maintenance

Node which contain file replica becomes nodes replica where nodes join or leave network rapidly which makes peer to peer network dynamic in nature in this dynamism it is very difficult to maintain file replica in consistent state, rather than depending on structured or message spreading technique IRM employs Adaptive polling of file replicas updates for consistency maintenance but the major issues were *How to determine when to probe file owner for replica update for periodic polling of file updates ?* And other issue was *How to optimise polling operation to save cost, resources and provide guarantee of consistency of file replica?*

IRM employs *linear increase multiplicative decrease algorithm* [2]. In which every file have some time interval after that it updates that is the frequency of updation of that file and nodes will also ensure that up to this time interval the file will not update and can fix the threshold value for polling that threshold is called as Time-To-Refresh, this threshold value is get updated time to time as per file polling result which ultimately makes it dynamic file consistency maintenance, if owner of file will get failed or leave the network then nodes which contain file replicas will probe to new owner of file or other junction node which contain updated file or it will remove the replica of that file from node. Now the major issue was that how to optimise polling work that is as replicas are created dynamically depending of that file's file request query initiation rate and also maintaining consistency only such files which are requested frequently and files which are underutilize will not be probe to its owner which increase load and unnecessary updation of file such files should be removed from node for that IRM proposes & implemented Adaptive File Consistency Maintenance Algorithm[2], In which both the threshold value that is for request of replica and polling of that replica compared and if result is more than threshold value then further file creation as well as updation takes place, algorithm mentioned below shows that these for polling of file it check for both threshold value after file replica creation which reduce polling request to file replica owner and unnecessary updation of replicas this will ultimately save cost, resources and give fidelity of consistency of file replica.

B. Architecture of Hadoop File System

HDFS file system Architecture as name suggest is a distributed system it contains various nodes for their operation where namespace of HDFS is hierarchy of files and directories which are represented on *NameNode* by *inodes* which records all data about data that is metadata, also *NameNode* also maintain *namespace* tree and mapping of block to *DataNodes*, whenever HDFS client request for read or write the file first it contact to *NameNode* from where it will get the location of data block then reads the data block from *DataNodes* nearer to client, each *DataNodes* can execute multiple requests to file or data block as HDFS client practises pipeline technique for their request of read and write

DataNodes stores the actual data as well as metadata like checksum and generation stamp, that is each data block is represented by two files in *DataNode* whose location is stored in *NameNode*, by default the file replication factor is three where file is replicated by *MapReduce* Framework API where these files are replicated at junction node which is near to switch called as racks in HDFS, and every operation to files are performed in pipeline fashioned as shown in figure 2.

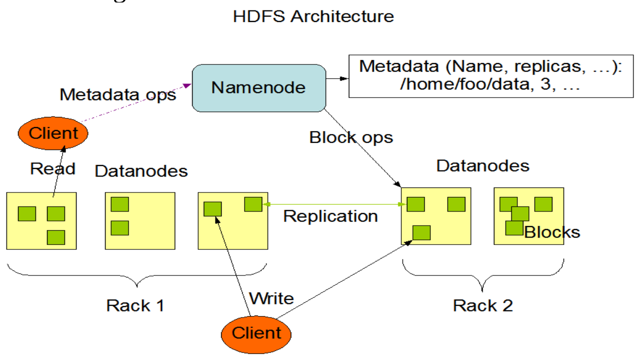

Fig. 2. Architecture Of HDFS [1].

Other than these nodes there are also *CheckpointNode* and *BackupNode*, this *CheckpointNode* is nothing but backup of *NameNode* which periodically copies the data of *NameNode* and protect the file system metadata, *BackupNode* is same as that of *CheckpointNode* but *BackupNode* is exact image of *NameNode* which is always synchronized with *NameNode*, as HDFS is single-writer, multiple-reader model, at a time only one HDFS client can write the file in pipeline fashion but multiple HDFS client can read file simultaneously, for this operation HDFS implements Lease on file for writing into the file as well as reading to the file, HDFS maintain the dynamism by providing replication at *DataNodes* at different switch or we should say racks, *DataNodes* may leave the network, fail or data file may get corrupted at such situation HDFS client get data file from other *DataNode* which is nearer to rack (switch) or HDFS client, it also have metadata file to check the integrity of data file that is checksum which recognizes that is file corrupted or consistent.

HDFS also provides tool named as DistCp for working with large data sets for inter/intra cluster parallel copying, this tool is work on MapReduce Framework that is it is MapReduce job in which MapReduce itself handle the parallelism of task scheduling, error detection and recovery.

C. Architecture of Dynamic Adaptive MapReduce Platform (MARALA).

MARALA is implemented on top of Hadoop that is MapReduce which performs data replication on heterogeneous cluster above figure shows the architecture of MARALA the name of technology which author Zacharia Fadika et al. proposes and implemented. MARALA architecture is distributed as per functionality of MapReduce model; those modules are *Splitter*, *TaskController* and *FaultTracker*.

Splitter which distribute the work among nodes equally which create replicas at each node this is mapping operation of MapReduce which deals with Shared Files system this mapping is set by user as per their need and requirement optimize their work by Reduce functionality of MapReduce this dynamic mapping & Reducing functionality will ensure that if any node get failed or leave the cluster it can resume its work after re-joining or overcome from failure state which also isolate data fidelity from nodes, *TaskController* is responsible for tracking of jobs of request of replication of file that is mapping and optimizing work that is reducing task, *FaultTracker* which is responsible for resubmitting jobs from task bag which are failed that is in case of node get failed or leave cluster which uses threshold value to identify failure of task and increase task tolerance these all functionality is distributed in nature and implemented on top of MapReduce which makes MARALA dynamic and consistent.

D. Architecture of Cloud MapReduce

In the cloud environment the scenario is totally different from heterogeneous cluster, here all nodes are autonomous and utilize the services of cloud where data replication is may be of structured or unstructured data. Author Huan Liu et al. proposes implementation of MapReduce on cloud in which author uses Amazon cloud service and uses word count application to illustrate the implementation where distribute data store is S3 and uses simple queue service called as SQS for computation where jobs that is for word count is scheduled on of cloud service EC2.

Figure 4 shows the architecture and working of MapReduce on top of cloud. Map function will distribute the work through various SQS which have key-value paired data then these mapping the output is transfer to master reduce queue which is polled by reduce worker to generate the output and then again put this output in SQS output queue.

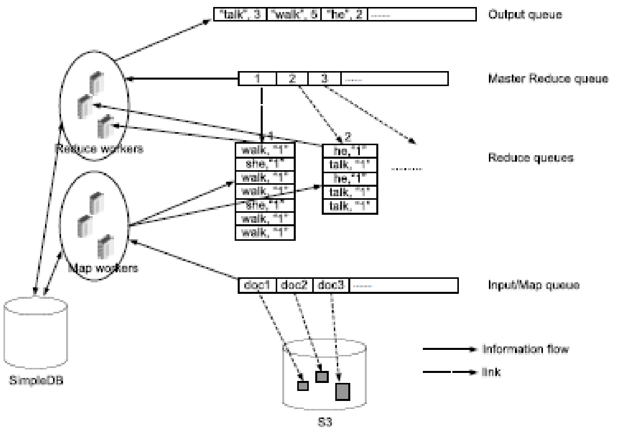

Fig. 3. Cloud MapReduce Architecture [3]

V. COMPARATIVE STUDY

In comparative study we are going to compare the experimental results of cloud MapReduce Framework, MARLA framework and Hadoop file system; here we are not going to discuss the performance evaluation of integrated data replication and consistency maintenance because as it works in peer to peer network, it will limit the scope of data replication of "Big Data" in heterogeneous network but in our future scope and further implementation we are going to use the concept of integrated data replication and consistency maintenance.

Now, here author Konstantin Shvachko et al. implements large HDFS cluster at Yahoo! With 3500 nodes and shows the outstanding results also author Zacharia Fadika et al. implements MARLA framework at NERSC (National Energy Research Computing Center) and grid & cloud research lab cluster Binghamton university which work on top of Hadoop file system and showed the performance of MARLA is best and author Huan Liu et al. implements the framework of MapReduce in cloud which shows optimal operation on HDFS, here in all these experiments is that all architecture adopts the MapReduce framework for their operation, in HDFS cluster at Yahoo! Author Konstantin Shvachko et al. executes sorting procedure, author Zacharia Fadika et al. executes word count procedure in MARALA architecture and author Huan Liu et al. executes word count, matrix multiplication, string matching and reverse index procedures these all evaluation were done in MapReduce Framework. Here all operation is performed on nodes which have configuration 1.2 GHz to 2.5 GHz processor, 16GB of RAM and Linux Operating System and having 1Gbps of bandwidth network.

Table 2. Comparative Results of Architectures [1][3][4].

Architecture	Nodes	Data Bytes	Map	Time	Efficiency
Hadoop	1460	1 TB	8000	62 s	Good
MARLA	75	50-100 GB	3200	1320 s	Better
Cloud MapReduce	100	100GB	200	264 s	Best

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved

Table 2. shows that cloud environment have higher efficiency as it provides the storage and computation isolated but you might think that Hadoop has good results but it also utilized the more resources and after all HDFS is our base for data replication here in this table map column shows the number of threads or task that are distributed in various nodes in pipelined fashion and time column represents time require to execute these task but as we see the number of nodes used in evaluation we will discover that cloud MapReduce has good performance than Hadoop and MARLA with minimum resources.

VI. CONCLUSION

Cloud MapReduce architecture gives outstanding results for Cloud based data replication and their operations, as cloud makes network heterogeneous in nature allow to scale and increase the capability in horizontal as well as vertical.

But in cloud MapReduce architecture implements the readymade cloud of amazon in which EC2 server for computation and S3 store for data storage, and MARLA [4] works on data cluster which at some extends makes it dominant in nature while in Hadoop the node NameNode makes is centralize in nature and jeopardize the reliability which also makes our future scope limited.

FUTURE SCOPE

With MapReduce framework the drawback of centralize operation of NameNode and homogeneous cluster can be overcome. In which each node will maintain their own data files with their metadata in HDFS that is *NameNode* and *DataNode* will be reside at same node. Each node will also maintain data replica or if needs, it will fetch that data file from nearer node. Where data replication and consistency will be maintain by the concept of IRM [2] and each node will work autonomously, that means each node will map the data replication and consistency operation using MapReduce framework and ultimately makes this “Big Data” replication distributed and dynamic. This cloud based data replication of “Big Data” become Software as a Service (SaaS) in cloud system.

REFERENCES

1. Konstantin Shvachko, Hairong Kuang, Sanjay Radia and Robert Chansler “The Distributed File System” IEEE 2010.
2. Haiying (Helen) Shen "IRM: Integrated File Replication and Consistency Maintenance in P2P System" IEEE Transactions On Parallel And Distributed Systems, Vol. 21, No. 1, January 2010.
3. Huan Liu and Dan Orban “Cloud MapReduce: a MapReduce Implementation on top of a Cloud Operating System” 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2011.
4. Zacharia Fadika, Elif Dede, Jessica Hartog and Madhusudhan Govindaraju “MARLA: MapReduce for Heterogeneous Clusters” 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2012.
5. Kapil Bakshi “Considerations for Big Data: Architecture and Approach” Proc. IEEE Symp. Mass Storage Systems and Technologies, 2012.
6. D. J. Abadi “Data Management in Cloud: Limitations and Opportunities” IEEE Data Eng. Bull., Vol. 32, no. 1, 2009.
7. E. Brewer, “Towards robust Distributed systems” in PODC, 2000.
8. A. Rowstron and P. Druschel “Storage Management and Caching in PAST, a Large Scale, Persistent Peer to Peer Storage Utility”, Proc ACM Symp. Operating System Principles, 2001.
9. F. Dabek, M. F. Kaashoek, D. Karger, R. Morris and I. Stocia, “Wide Area Cooperative Storage with CFS”, Proc ACM Symp. Operating System Principles, 2001.
10. X. Chen, S. Ren, H. Wang and X. Zhang “SCOPE: Scalable Consistency Maintenance in Structured P2P System” Proc. IEEE INFOCOM, 2005.
11. I. Clarke, O. Sandberg, B. Wiley and T. W. Hong “Freenet: A Distributed Anonymous Information Storage and Retrieval System ”, Proc Int’l Workshop Design Issue in Anonymity and Unobservability, pp. 46-66, 2001.
12. Dharma Teja Nukarapu, Liqiang Wang “Data Replication in Data Intensive Scientific Application with Performance Guarantee” IEEE Transaction on Parallel and Distributed System, vol. 22, No. 8, August 2011.
13. Vazhkudai S., Tuecke S., Foster I. (2001). Replica Selection in the Globus Data Grid, Proceedings of the International Workshop on Data Models and Databases on Clusters and the Grid (DataGrid 2001), (pp: 106-113), May 2001, IEEE Computer Society Press.
14. Raman P., Livny M.and Solomon M. (1998). Matchmaking: Distributed Resource Management for High Throughput Computing, In Proceedings of 7th IEEE Symposium on High Performance Distributed Computing (HPDC) (pp: 140-146), July 1998, IEEE Computer Society Press.
15. Ranganathan K. and Foster I. (2001). Identifying dynamic Replication Strategies for a High-Performance Data Grid, Proceedings of International Workshop on Grid Computing (pp: 75-86), Denver, USA, November 2001.
16. Carman M., Zini F., Serafini L. and Stockinger K. (2002). Towards an Economy-Based Optimisation of File Access and Replication on a Data Grid. Proceedings of the 1st IEEE/ACM International Conference on Cluster Computing and the Grid (CCGrid), (pp: 340-345), Berlin, Germany, May 2002, IEEE Computer Society Press.
17. Bell W. H., Cameron D. G., Carvajal-Schiaffino R., Millar A. P., Stockinger K., Zini F. (2003). Evaluation of an Economy-Based File Replication Strategy for a Data Grid, Proccedings of 3rd IEEE International Symposium on Cluster Computing and the Grid (CCGrid), Tokyo Japan, May 2003, IEEE Computer Society Press.
18. Lamehamedi H., Shentu Z., Szymanski B.and Deelman E. (2003). Simulation of Dynamic Data Replication Strategies in Data Grids, Proceedings of the 17th International Parallel and Distributed Processing Symposium (PDPS), Nice, France, IEEE Computer Society.
19. Dullmann D., Hosckek W., Jaen-Martinez J., Segal B., Samar A., Stockinger H. and Stockinger K.(2001). Models for Replica Synchronisation and Consistency in a Data Grid, Proceeding of 10th IEEE International Symposium on High Performance and Distributed Computing (HPDC), (pp: 67-75), San Francisco, August 2001.
20. Venugopal S., Buyya R., and Ramamohanarao K. (2005). A Taxonomy of Data Grids for Distributed Data Sharing, Management and Processing (Technical Report, GRIDS-TR-2005-3), Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia.