ISSN:0975-9646

On Replication Strategies for Data Intensive Cloud Applications

Rashmi Ranjana T P, Jayalakshmi D S, Dr. Srinivasan R
Department of Computer Science and Engineering
M S Ramaiah Institute of Technology
Bangalore, India

Abstract— **Data Replication is an important aspect to be considered under the cloud and particularly so in the area of data intensive applications. Various replication strategies used in Amazon, Google and Microsoft cloud offerings are discussed followed by comparison of current research scenario and associated algorithms through a suitable tabulation. The merits and gaps existing in these algorithms and suitable suggestions for improvement are presented in this paper.**

Keywords- Cloud Computing, Data Replication Static and Dynamic, Data Intensive Application

I. INTRODUCTION

Over recent years there is drastic increase in the size of data that needs to be managed effectively, where this data can be structured or unstructured. Data Intensive computing refers to computing of this large scale data [1]. Computing in such applications is not capacity driven instead I/O driven. Such data of size in petabytes cannot be stored on a single table and hence data must be portioned, distributed and requires specialized datacenters. The traditional Relational Database Management Systems [RDMS] do not fit to consistency and scalability.

At present there are many custom software packages that manage structured data and metadata in RDBMS (Google, Yahoo!, Facebook etc., also use RDBMS for OLTP). Data intensive clouds provide high computation power as an abstraction to users. A Data Intensive cloud can be deployed in two ways based on its usage as: private cloud or public cloud. There are complex computations that must be performed which require processing and management of large scale data, achieving high performance and high throughput, and storing it efficiently for future use. There are many research issues, in terms of capturing and accessing data effectively and fast.

Data intensive cloud applications are deployed on multiple data centers. In such clouds, faults are normal which lead to failure and crashes that occur any time. When an application or service needs data which is not available on local database, remote access to data on other data center has to be made. Distributed file systems such as GFS, HDFS, and so on, provide solutions to such applications. Data Replication can be defined as technique in which each logical data item of a database has several physical copies, each of them located on a different machine, also referred to as site or node [2]. Among the Distributed File Systems there are different levels of replication that is maintained in each of them. The data replication level in popular cloud data management systems are: Amazon S3 & Dynamo – item level, GFS – chunk level, HDFS – block level.

This approach can be used to reduce the latency of remote data access by storing data close to application and service that use it. Although there is improvement in performance by replicating data in each data center and accessing local databases as when required there are challenges with respect to synchronization of data and storage of large number of replicas. To cite a few, we are giving below some typical case studies.

II. DATA REPLICATION STRATEGIES - EXISTING SCENARIO

In recent years, the task of storing data persistently is done by simple storage system which can maintain a large scale data and also achieve availability and reliability. Service providers of cloud environments are responsible for providing such large scale data management systems.

A. Amazon Web Services – RDS

Amazon Relational Database Service (RDS) is a web service which makes it easy to set up, operate, and scale a relational database in the cloud. Amazon RDS DB Instances can be provisioned with General Purpose (SSD) Storage, Provisioned IOPS (SSD) Storage, or Magnetic Storage [3]. For production workloads, Amazon RDS provides replication to enhance database availability, scale beyond the capacity constraints of a single DB Instance for read-heavy database workloads, and disaster recovery.

There are two types of complementary replication features provided by Amazon RDS: (1) Multi-AZ Deployments – This is an option during deployments which increases the database availability and protects database's latest updates which are lost due to unplanned outages. When a DB Instance is created as multi-availability zone (AZ) deployment, RDS keeps a replica of data in standby mode in another availability zone which is physically at different location. The updates to database are made concurrently on both primary node and standby replica to prevent inconsistency in data. In cases of failure Amazon RDS will automatically failover to a stand-by which is up-to-date and there is no interruption in operations performed on database.

(2) Read Replicas – Although there is standby replica, which can be used to serve heavy read traffic, it is not possible to directly access it before failover. Thus read replicas feature can be used to elastically scale out capacity of single DB instance. When there is heavy read, multiple replicas of given source DB can be created in the AWS regions and thereby increase the overall throughput. The changes made to source DB instance are updated and then associated read replica is propagated with this update using MySQL's native replication (i.e., asynchronous replication).

Thus, these two techniques can be combined where Multi-AZ deployments are given as source DB instance to read replicas for getting advantages of availability, durability and scaling.

B. Amazon Web Services – DynamoDB

DynamoDB is fully managed NoSQL database service that has very high availability and scalability due to key/value based data store. The data items that need to be accessed for read and write operations are identified by a unique key. None of the operations span more than one item in such systems. Consistent hashing principle is used to store each data item and their replicas on hosts. DynamoDB automatically spreads the data and traffic for a particular table among sufficient number of servers to handle request, while maintaining consistency and performance.

C. Windows Azure Storage

The design goal of Windows Azure storage is to provide consistency, availability and partition tolerance (CAP Property) and load balancing. The layered architecture consists of three layers: 1) Front End Layer – takes the incoming requests, 2) Partition Layer – manages the partitioning of all of the data objects in the system, and 3) Distributed and replicated File System Layer (DFS) - actually stores the bits on disk and is in charge of distributing and replicating the data across many servers to keep it durable. In DFS layer the data is stored as "extents". For availability, each layer has its own form of automatic load balancing and dealing with failures and recovery in order to provide high availability when accessing your data. For durability, this is provided by the DFS layer keeping multiple replicas of your data and using data spreading to keep a low MTTR when failures occur. For consistency, the partition layer provides strong consistency and optimistic concurrency by making sure a single partition server is always ordering and serving up access to each of your data partitions [4].

To maintain high availability for service and overcome node failures, there are different methods followed at each layer. A typical case is shown by a flow chart below in Fig 1:

D. Google File System

Google File System is a distributed system that runs on clusters. The architecture followed is Master/Slave pattern, where Master is responsible for managing and monitoring clusters and data is stored on slaves called as chunkservers. In order to provide data safety and availability, data is replicated and stored on multiple chunkservers. By default minimum number of replicas in this system is three. Files are divided into chunks (referred to as blocks) of fixed size i.e., 64MB. Master has metadata which manages mapping between files and chunks, as client always refer to file as a whole.

Re-replication is performed for those chunks whose replica number has fallen below minimum replication count due to failures such as chunkserver crashes, disk failures or failed integrity checks. Another reason for re-replication can be to improve data access by increasing replica count of chunks, which belong to files that are accessed frequently. When the number of replicas increases and the file is not accessed more often, they are not deleted immediately but marked as deleted. Master periodically runs garbage collector to remove all chunks that have become orphaned or marked as deleted for at least three days.

Above mentioned are most popular commercial service providers for managing data intensive applications. The challenges organizations usually face when moving to the cloud are incompatibility issue, security issues, reliability issue and network issue.

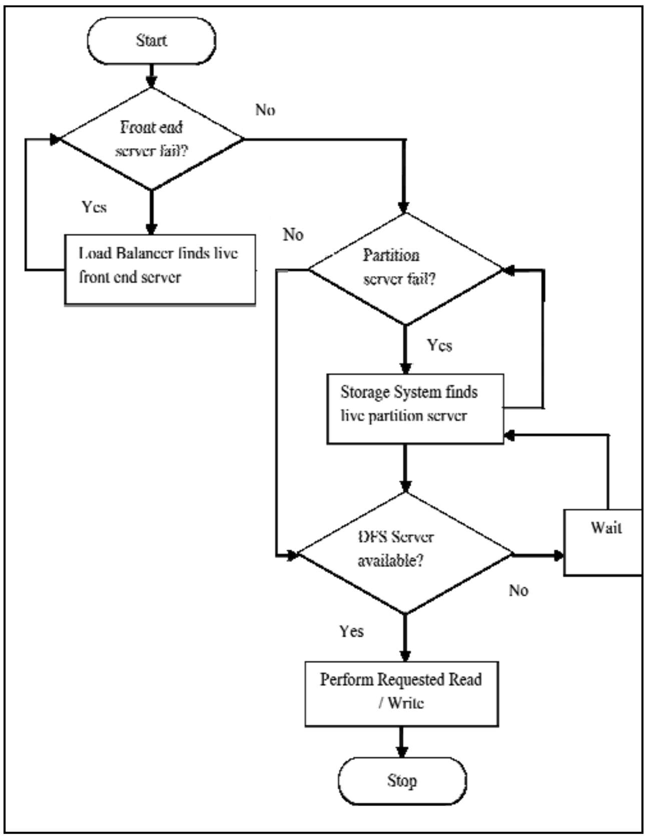
Start
Front end server fail?
No
Yes
Load Balancer finds live front end server
No
Partition server fail?
Yes
Storage System finds live partition server
DFS Server available?
Wait
No
Yes
Perform Requested Read / Write
Stop

Figure 1. Flowchart showing the methods used to overcome node failures

Going forward, there are many related works on cloud storage systems and cloud data replication which aim at bringing out most of the positive features of data replication and propose new strategies, models, algorithms to overcome issues without compromising on QoS.

E. Replication strategies in research literature

Data Replication algorithms can be categorized into two types:

(1) Static Replication - Replication strategy is predefined

(2) Dynamic Replication - Algorithm creates and deletes the replicas based on the access patterns of each replica.

The static replication algorithm proposed in [5] gives detailed description on the architecture of GFS, in which single master makes decision on data chunk replications based on factors like: average disk space utilization, number of recent replica creation, distribute new replicas. A new replica is created when number of replicas in the system for a particular data chunk falls below a limit specified by user. In [6], a *p*-median static centralized data replication algorithm is proposed along with a model for placing these new replicas so that the total distance between requesting sites and replica sites is minimized.

As there is uniform replication with fixed number of replica in Windows Azure and Amazon S3 there is inefficient resource usage for user. Each user has different requirements in terms of availability, durability, reliability, and hence static replication strategy cannot be used. To satisfy this, in [7] four differentiated replication strategies are proposed for datacenters which are combination of uniform (same number of replicas) or non-uniform replication with or without DHT lookup algorithm. The algorithms which take into account both user requirements and system behavior gives best results. In this work only OPERA (**OPE**n **R**eputation **A**rchitecture) monitor server is used to give reputation scores for all datacenters based on their availability. For every user request based on requirement one of the four strategies are used. Objective of the algorithm is to improve availability by increasing number of replicas and placing them on set of host machines randomly.

Static replication strategies are inadequate to those applications where computation patterns are non-uniform. Thus an adaptive replication strategy is proposed in [8] which adapts to change in workload. FIFO (First in First Out) scheduler and Fair scheduler are used to achieve better data locality with low system overhead. Instead of performing remote access to data every time it is required for computation, subset of a popular file can be stored on local node as a replica. There are two approaches: Greedy Approach – Any remote access data is replicated and Probabilistic Approach – Data not replicated as soon as they are read, instead they are replicated (or evicted) with a probability p. The number of replicas created for a file is maintained by Aging and Replica Eviction Policy which use LRU (Least Recently Used) policy in Greedy approach or use probabilistic approach. The threshold value Budget is mainly responsible for decision making in creation of new replica which can be a bottleneck.

In the previous work, importance is given in maintaining number of replicas for improving data locality. The location of this replica is of concern only when data center storage is limited. In scientific cloud workflows data must be stored effectively on data centers reducing data movement during workflow execution. In [9], matrix based k-means clustering strategy is proposed for such systems where data placement is of greater importance. In this strategy the datasets which are required by many tasks, together are said to be dependent data sets. Such kind of dependent data sets are kept in one data center so that there is no need of data movement. There are two stages in this strategy: Build-Time Stage – Algorithm goal is to set up k-initial partitions for k-means algorithm and Runtime Stage – Algorithm goal is to cluster the newly generated datasets to one of the k-data centers based on their dependencies, which will be calculated dynamically [9]. While placing datasets to a data center in runtime stage, it must be checked if there is space available for storage and it can balance overall workload of system.

Continuous change in dependencies between the tasks and data centers is an existing challenge which requires change in data placement strategy accordingly. Since the idea to reduce large volumes of data movement, replication can be used along with the two stages which are proposed in [9]. An extension of this work is given in [10] which proposes an additional step - Replication Stage. During Replication phase, tasks are scheduled to those data centers where most of the datasets required by them are available locally. This algorithm is developed to improve response time of the system.

The systems where in communication resources are bottleneck, data replication strategy is used for storing data closer to the locations where computing applications are executed. In [11], an energy efficient data replication is proposed to optimize the energy consumption and bandwidth along with improving other QoS. Data access statistics is used to identify data items which are most suitable for replication and replication sites where they can be placed. Cloud Manager at central database periodically performs analysis on statistics of data availability and updates for data, which is used to find out bandwidth and energy that is consumed. A model is presented for data transmissions in data centers such as: uplink and down link transmissions. The simulation results also show speeding up of workflow execution by minimizing communication delays.

In [12], a model is designed to get the relationship between availability and replica number, to find minimum number of replicas that a data item can have to realize availability leas required, to decide on where these replicas need to be placed on data nodes taking into account capacity and blocking probability of each node. When the number of sessions has reached its upper bound, connection requests from application servers will be blocked or rejected [12]. If there is an efficient replica placement strategy used, then inter-request and intra-request parallelism are improved, which also result in improvement of performance and load balance of HDFS (Hadoop Distributed File System) cluster. In this system, block independent distribution policy is considered. For varying workloads a Dynamic Replica Control Strategy is used to run on name node. There are two threshold values, namely threshold for migration and threshold for deletion based on which this dynamic replication control works.

Another adaptive strategy is proposed in [13], to improve reliability of system and other QoS. Replication process is managed by a scheduling broker, which contains all the information about the number of replicas and their locations at

TABLE I. COMPARISION OF EXISTING REPLICATION STRATEGIES

Sl. No.	Title of Paper	Techniques Used	What is Optimized	Experiment Setup	Metrics for Evaluation
1	Energy-Efficient Data Replication in Cloud Computing Datacenters [11]	•Dynamic Data Replication Strategy •3 tier topology •Model for data transmissions (Uplink and Downlink transmissions)	•Datacenter Energy consumption •Residual Bandwidth	GreenCloud Simulator	•Data center energy consumption •Available network bandwidth •Communication delay.
2	A Light-weight Data Replication for Cloud Data Centers Environment [13]	•Adaptive Data Replication Strategy •Lightweight time series prediction algorithm - HELS (Holt's Linear and Exponential Smoothing)	•Non-Functional QoS •Overall Reliability	CloudSim Simulator	Response time
3	Differentiated Replication Strategy in Data Centers [7]	• Differentiated Replication (DiR)(Uniform / Non-Uniform Replication) •DHT lookup algorithm.	•Resource Utilization •Availability	Chord/DHash (C++ on Linux)	•Execution time •Availability
4	SWORD: workload-aware data placement and replica selection for cloud data management systems [14]	•Workload-Aware Data Placement & Replication approach •Incremental Repartitioning Technique •Hypergraph Partition Algorithm (HPA) - to model workload	•Query Span •Reduction in total resource consumption •Transaction latency •Overall Throughput	In Application Domains like •Distributed Analytical •Distributed OLTP Data Stores	•Number of Partitions •QuerySize •Number of Queries •Graph Density
5	DARE: Adaptive Data Replication for Efficient Cluster Scheduling [8]	•Adaptive Data Replication Scheme •Greedy Algorithm •Competative Aging Alogrithm (i.e LRU, LRF) •Probability Algorithm - ElephantTrap	•System Overhead •Data Locality	In Hadoop Framework	•DataLocality •Geometric Mean of the Turnaround Time (GMTT) •Slowdown of Job
6	A Data Placement Strategy in Scientific Cloud Workflows [9]	•Matrix based k-means Clustering Strategy •Build-Time Stage - to set up k-initial partitions •Runtime Stage - cluster the newly generated datasets to one of the k-data centers	Data Movement between Data Centers	•SwinDeW-C Simulation Environment •Hadoop File Systems on Each Data Center • Vmware for physical servers	Number of Datasets that are actually moved during the Workflow Execution
7	Optimization of Tasks Scheduling by an Efficacy Data Placement and Replication in Cloud Computing [10]	same-as-above – 11 & Replication Stage - tasks are scheduled to those data centers where most the datasets are available locally	•Data Movement between Data Centers •Response Time	Simulator in Java	•Number of Displacements •Response Time
8	CDRM: A Cost-effective Dynamic Replication Management Scheme for Cloud Storage Cluster [12]	•Model is designed to get the relationship between Availability and Replica Number •Block Independent Distribution Policy •Dynamic Replica Control Strategy (Run on Namenode)	•Number of Replicas •Inter-request and Intra-request parallelism •Performance •Load Balance	Hadoop Framework	•Availability •Performance •Load Balance

different data centers. Recent pattern of data files request is used to predict next data file that will be requested. The linear series technique - HELS (Holt's Linear and Exponential Smoothing) is used to predict the future access frequency of the data since it has low computation time. If such popular file has replication factor less than threshold, replication is triggered. If a file is no more popular and there are no data access requests for it, then replicas of such files must be deleted.

There are read-only analytical workloads that need to process large volumes of data in an efficient manner, as well as transactional OLTP(On-Line Transaction Processing)-style workloads that need to support high throughputs with low latencies [14]. With the goal of having hold on different workloads such that overall system resource is utilized efficiently, work [14] proposes a workload – aware approach. In this paper an abstract metric called query span i.e., average number of machines used for execution of workload is proposed which is used instead of resource consumption. Along with employing data replication techniques to reduce average query span, an incremental repartitioning technique is proposed which is based on identifying candidate set of data items which can be migrated and yet their query span can be reduced efficiently. Hypergraph Partition Algorithm (HPA) is used, as there is large scale data used in execution.

III. Observations and Suggestions on above algorithms

By analyzing the table created above in Table 1, weakness in some of these strategies are pointed out along with some suggestions for improvement:

- In [7], the choice of a replication strategy is done in the beginning based on user request. Thus it falls back into the category of static replication prone to issues of static strategies. To search for replica BFS (Breadth First Search) is used which is not that efficient when the number of replicas is large. In this OPERA monitor server used is implemented to improve only availability but other non-functional requirements are not implemented. Replica placement must be considered seriously in this approach instead of placing randomly.
- In [8], the adaptive replication algorithm proposed considers popular data file and creates replica of it. Due to increase in number of replicas there are chances of these replicas being stored on single datacenter and cause contention. Thus all those data blocks accessed concurrently must be placed on different nodes to reduce contention.
- In [9,10], replication mechanism is used within data center, but any replication strategy is not used among data centers. To achieve high availability, replication similar to inter-stamp and intra-stamp in Windows Azure architecture can be used in scientific workflow cloud systems. In all the above mentioned replication strategies, there are many copies of data files stored at different locations but none of them considers synchronization of data and consistency of data. Synchronization results in network bandwidth expenses but still data must be synchronized periodically.
- In data intensive cloud applications, computations are performed and large data caching can be used which will reduce the processing time and database access time drastically. For these data files to be consistent TTL (Time to Live) can be set for each of the files and updated timely. But again there is disadvantage of using cache since there are many nodes with same data files resulting in duplicates. Hence it is used when reducing execution time is of main concern.
- High availability and fault tolerance can be achieved by using the approach followed in AWS by creating availability zones. One or more availability zones can be created for disaster recovery and failover.

IV. Conclusion & Future work

The algorithms proposed must be evaluated on test bed once they have given appropriate results in simulated environments. At present a scheduler alone is used in DARE [8] approach which aims at improving data locality. Further, it can be used in parallel with other schemes that improve data locality. A better heuristic algorithm can be followed to determine future data access request from cloud applications to improve replica placement strategy. Genetic algorithms can be used to find best replication in less time. for it, then replicas of such files must be deleted.

The replication strategies proposed in the existing literature typically optimize for energy efficiency, minimal data movement and load balancing in single cloud scenario. However, multiclouds, hybrid clouds and federation of clouds, are seen as the ways in which cloud computing will be used in the coming years. There is a need to devise replication strategies which are interoperable across geographically distrusted data centers belonging to same or different cloud providers. This gives rise to challenges in replica metadata management too.

References

[1] Jawwad Shamsi, Muhammad Ali Khojaye, Mohammad Ali Qasmi, "Data-Intensive Cloud Computing: Requirements, Expectations, Challenges, and Solutions". Received: 6 February 2012 / Accepted: 28 March 2013 © Springer Science+Business Media Dordrecht 2013.

[2] Ling Liu, M. Tamer Özsu, "Encyclopedia of Database Systems" 10.1007/978-0-387-39940-9_110 © Springer Science+Business Media, LLC 2009

[3] http://aws.amazon.com.rds

[4] http://blogs.msdn.com/b/windowsazurestorage/archive/2010/12/30windows-azure-storage-architecture-overview.aspx

[5] S. Ghemawat, H. Gobioff, S. T. Leung. "The Google file system". ACM SIGOPS Operating Systems Review, 2003, 37(5): 29-43.

[6] R. M. Rahman, K. Barker, R. Alhajj, "Replica placement design with static optimality and dynamic maintainability". In Proc. the 6th IEEE International Symposium on Cluster Computing and the Grid, Singapore, May 16-19, 2006, pp.434-437.

[7] T. Nguyen, A. Cutway, W. Shi, "Differentiated replication strategy in data centers" In Proc. the IFIP International Conference on Network and Parallel Computing, Zhengzhou, China,Sept. 13-15, 2010, pp.277-288.

[8] Cristina L. Abad, Yi Luy, Roy H. Campbell, “DARE: Adaptive Data Replication for Efficient Cluster Scheduling” University of Illinois at Urbana-Champaign.

[9] D. Yuan, Y. Yang, X. Liu, J. Chen, “A data placement strategy in scientific cloud workflows”, Future Generation Computer Systems (2010), doi:10.1016/j. future. 2010. 02.004

[10] Esma Insaf Djebbar, Ghalem Belalem, “Optimization of Tasks Scheduling by an Efficacy Data Placement and Replication in Cloud Computing” University of Oran, Oran, Algeria.

[11] Dejene Boru, Dzmitry Kliazovich, Fabrizio Granelli, Pascal Bouvry, Albert Y. Zomaya, “Energy-Efficient Data Replication in Cloud Computing Datacenters” Globecom 2013 Workshop - Cloud Computing Systems, Networks, and Applications: 978-1-4799-2851-4/13/$31.00 ©2013IEEE.

[12] Q Wei, B. Veeravalli, B. Gong, L. Zeng, D. Feng. “CDRM: A cost-e®ective dynamic replication management scheme for cloud storage cluster”. In *Proc. 2010 IEEE International Conference on Cluster Computing*, Heraklion, Crete, Greece, Sept. 20-24, 2010, pp.188-196.

[13] Mohamed-K HUSSEIN, Mohamed-H MOUSA, “A Light-weight Data Replication for Cloud Data Centers Environment” International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297: 2007 Certified Organization) Vol. 2, Issue 1, January 2014.

[14] K. Ashwin Kumar, Abdul Quamar, Amol Deshpande, Samir Khuller, “SWORD: workload-aware data placement and replica selection for cloud data management systems” The VLDB Journal DOI 10.1007/s00778-014-0362-1, Received: 23 September 2013 / Revised: 6 April 2014 / Accepted: 4 June 2014 © Springer-Verlag Berlin Heidelberg 2014.

[15] I. Stoica,R. Morris, D. Karger, M. F. Kaashoek, H Balakrishnan, “Chord: A scalable peerto-peer lookup service for internet applications” In: ACM SIGCOMM 2001 (2001)

[16] Siba Mohammad, Sebastian BreB, Eike Schallehn, “Cloud Data Management: A Short Overview and Comparison of Current Approaches” 24th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 29.05.2012 - 01.06.2012, Lübbenau, Germany.

[17] E. Hewitt. “Cassandra The Definitive Guide” O Reilly Media, Inc, 2010.

[18] Priya Deshpande, Aniket Bhaise, Prasanna Joeg, “A Comparative analysis of Data Replication Strategies and Consistency Maintenance in Distributed File Systems” International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-2, Issue-1, March 2013.

[19] 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.