Replication: A Technique for Scalability in Cloud Computing

Nagamani H Shahapure
JSS Academy of Technical Education
Uttarahalli-Kengeri Main Road
Bangalore-60

P Jayarekha, Ph.D
BMS College of Engineering
Bull Temple Road, Basavanagudi
Bangalore-19

ABSTRACT
Cloud computing is a technology which produces and consumes huge amount of data every day. This makes the cloud to store tons of applications. The demand for these resources is on the rise. Multi cloud environment is used to satisfy these demands. If multiple providers cooperatively work together, the availability of resource can be improved. The replication of data across multiple places in cloud has become an effective solution to achieve good performance in terms of load balancing, response time and availability. Replication of data is a good way to achieve reliability and improve performance in a distributed system. The rising popularity of cloud computing is an alternative to classic information processing systems. This has increased the importance of its correct and continuous operation even in the presence of faulty components. Fault tolerance is a major concern to guarantee availability and reliability of critical services as well as application execution. In order to minimize failure impact on the system and application execution, failures should be anticipated and proactively handled.

General Terms
Cloud Computing, Scalability

Keywords
Cloud, Distributed Systems, Replication, Availability, Fault Tolerance

1. INTRODUCTION
The main apprehension for the users storing the data in the cloud is to preserve the data and recover it whenever required. Any server failure should not result in data loss. Cloud applications include gaming, voice and video conferencing, online office, storage, backup, social networking. The performance of these applications depends largely on the availability of high performance communication resources and network efficiency [1] [2]. Data replication is a commonly used technique to increase the data availability. It requires a high bandwidth data throughput path. Cloud replicates the data and stores them strategically on multiple servers located at various geographic locations. Replication ensures consistency; improves availability and reliability by creating multiple copies of the same data on different storage devices and geographical locations. Replication and availability play a major role in fault tolerance.

Data is distributed across the cloud. This has to be made available to the applications that want to use it. The performance must not be degraded. The data access speed should be increased, keeping the load balanced in the system. Scalability and availability are the two major factors to improve the performance of the cloud. Creating replicas is also one of the important strategies to achieve the above. Replication also reduces access latency and bandwidth consumption. Some of the concepts related to replication are discussed in the next section.

2. REPLICATION
Replication is creating multiple copies of an existing entity [15]. Replication increases availability of resources. It also provides consistency and reliability by creating multiple copies of the same data on different sites. Replication also provides minimum access cost, shared bandwidth utilization and delay time by replicating data. The value of replication is to provide transparent, flawless access to resources in the event of a system failure. Replication can be extended across a computer network so that storage devices can be located in physically separated facilities.

Users access nearby replicas and increase the throughput in case of failure to maintain the transmission of data. There are advantages of storing the data at more than one site. If a server with the required data fails, a system can operate using replicated data. This concept maintains availability. The data is stored at multiple sites. The requested data is fetched from the nearest source from where the request originated. This increases the performance of the system. The benefits of replication do not come without overheads of creating, maintaining and updating the replicas. Replication can greatly improve the performance [11].

There is a performance overhead in replication technique as it takes time to recover data from other sites and restart the service again. The advantage is fault can be tolerated and availability can be increased [10].

2.1 Data Availability
The availability of a service is given by the quantity of the time that the service is in use by the clients in both normal and abnormal conditions. The high availability arises from the fact that its clients need a permanent access to the service [7]. The unavailability of some services has a negative impact for their clients. This is in case of banking institutions, telecommunication companies, military applications or hospitals. Cloud infrastructures offer availability above 99.9%; therefore performance degradation is a more serious concern than resource failures in such environments [8]. The rise in the demand for continuous availability of high performance computing (HPC) systems is evident. This is a major step towards capability computing, where scientific applications desire considerable amounts of time (weeks and months) without interruption on the fastest HPC machines available [9]. These high end computing (HEC) systems must be able to run in the event of failures in such a manner that its potential is not rigorously degraded. High availability (HA) computing has for a long time played an important role in mission critical

applications, such as in the military, banking, and telecommunication sectors. To achieve high availability a number of replicas are stored across multiple servers. If unavailability of a server is exceeds a particular time then the replication process is automatically started. This ensures continuous service.

2.2 Fault Tolerance

The vulnerability to failures is one of the problems in cloud computing systems. Indeed, whenever a single node crashes, availability of the whole system may be compromised [3]. However, the distributed nature of those systems provides the mean to increase the reliability of the system. A fault tolerant system is a configuration that prevents a computer system from failing in the event of an unexpected problem. Fault tolerance is the ability to preserve the delivery of expected services despite the presence of fault causing errors in the results within the system [4]. It aims at the avoidance of failures in the presence of faults. Errors are detected and corrected in a fault tolerant system. Permanent faults are located and removed while the system continues to deliver acceptable services. It is concerned with all the techniques necessary to enable a system to tolerate software faults remaining in the system after its development.

Fault tolerance can be classified into reactive fault tolerance and proactive fault tolerance [16].

2.2.1 Reactive Fault Tolerance

This technique reduces the effect of failures on application execution when the failure effectively occurs. One of the techniques applied is checkpoint/restart. When a task fails, it is allowed to restart from the point of failure instead of the beginning. Another method is retrying. In this method the failed method is retried in the same resource. Task resubmission is another approach. Whenever a failed task is detected, it is resubmitted either to the same resource or to a different resource at runtime. Another process of achieving this is through replication.

2.2.2 Proactive Fault Tolerance

The principle of proactive fault tolerance is to avoid recovery from faults. The faulty components are replaced with working components. One way of achieving this is by using a technique called software rejuvenation. Here the system is designed for periodic reboots. The system is restarted in a clean state. One more method used for proactive fault tolerance is through self healing. Multiple instances of an application run on multiple virtual machines. When one VM fails the other immediately takes over.

Fault tolerance, reliability, scalability and availability are directly proportional to each other. All these parameters are critical to ensure correct and continuous system operation. Availability is measured in terms of mean time between failures and mean time to repair [5]. High availability is addressed by means of replicating servers and storage [6].

Challenges in Replication [11]:

- Data Consistency: Maintaining data integrity and consistency in a replicated environment is most important. High precision applications may require strict consistency of the updates made by transactions.
- Downtime during new replica creation: If strict data consistency is to be maintained, performance is severely affected if a new replica is to be created. Sites will not be able to fulfill request due to consistency requirements.
- Maintenance overhead: If the files are replicated at more than one site, it occupies more storage space. This requires more additional maintenance. This becomes an overhead in storing multiple files.
- Lower write performance: Performance of write operations can be considerably lower in applications requiring high updates in replicated environment, because the transaction may need to update multiple copies.

The main aim of using replication is to reduce access latency and bandwidth consumption. The other advantages of replication are that it helps in load balancing and improves reliability by creating multiple copies of the same data. These can generally be classified as static and dynamic. In static replication, a replica exists until it is deleted by users or its duration is expired. Static replication is used to copy data to other datacenter where it is most popularly requested. The drawback of static replication is evident when client access patterns change greatly in the data. The drawback with static replication is that it cannot adapt to changes in user behavior. The replicas have to be manually created and managed if one were to use static replication. When events affecting a primary location where the data resides occur, data can be recovered from the secondary location to provide continued service, fault tolerance, higher availability. The other type is dynamic replication. In this method replica creation, deletion and management are done automatically. Dynamic strategies have the capability to adapt to the changes in the user behavior [19].

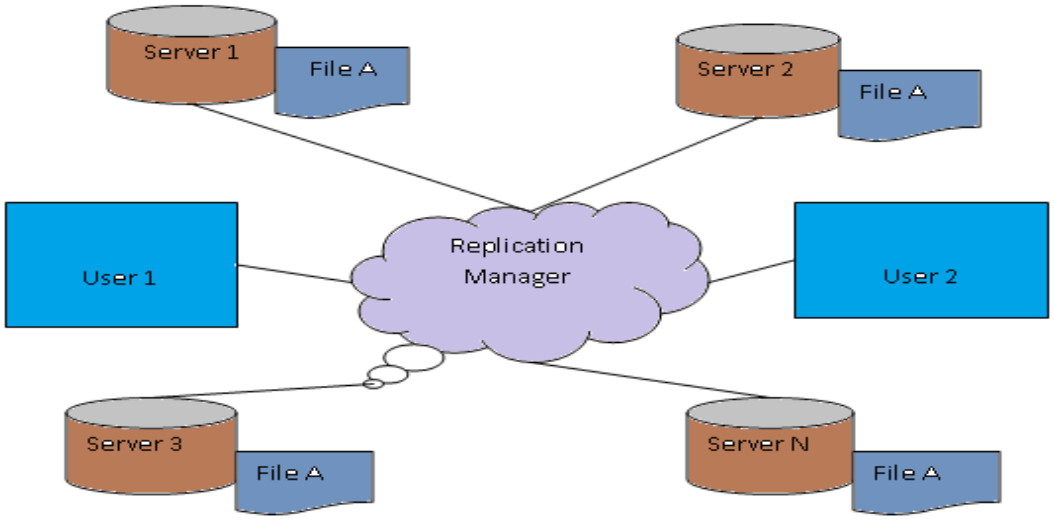

Fig 1: Replication Architecture

3. TYPES OF REPLICATION TECHNIQUES [3] [6]

- Active Replication: This is also called the state-machine approach. The behavior of the replicas is independent of each other. It is ensured that all the replicas receive the requests in the same order. In this technique the response time is low, even in the case of a crash. The disadvantages of this method: (1) the redundancy of processing implies a high resource usage (2) the requests have to be handled in a deterministic manner. Determinism means that the result of an operation depends only on a replica's initial state and the sequence of operations it has already performed. Multithreaded servers typically lead to non-determinism.

- Passive Replication: This is also called primary-backup [12] [13]. The primary (master) replica receives the requests from the clients and returns responses. The backups (slaves) interact with the primary replica only. They receive state update messages from this replica. This consumes less power than active replication. The implementation of passive replication requires a mechanism to agree on which resource has to be the primary replica (e.g., a group membership service). If the master fails, one of the slaves takes over. If the primary crashes before sending a response to the client, the clients will eventually time-out. A new master takes over. Then the request is reissued. This leads to a significantly increased response time in the case of failure that makes it unsuitable in the context of time-critical applications. Moreover, passive replication does not totally mask failures to the client.

- Quoram Replication: Quorum replication is based on Quorum consensus (Agrawal & El Abbadi 1991; Malkhi & Reiter 1997) where requests are processed by a quorum of replicas before returning to the client. A simple quorum (Q) is defined as any majority of nodes such that Q > N=2, where N is the total number of replicas. In the ideal case all operations would be processed in a majority before returning to the client in order to guarantee that each client always accesses the most recent version in the system. However, some systems allow the configuration of the size of read (R) and write (W) quorums in order to achieve better performance. To ensure the same properties as when a majority is used, the quorums must intercept in some node such that W + R > N. Otherwise, the quorums may not overlap and stale versions can be observed.

- Primary Backup Replication: In the primary-backup approach updates are made on a single replica (master) and then propagated to the remaining replicas. This propagation can be either made in an impatient or lazy way. In the impatient approach, the client issues an update and the master only replies after the update has been propagated to all the replicas. This method provides strong consistency. In the primary backup approach it is simple to control the concurrency of updates. A single master can be seen as a bottleneck in the system.

- Lazy Replication: In this approach it is assumed that the master replies to the client after the update has been applied locally. If the client issues a read operation to one of the other replicas before the propagation of the update the stale data can be seen.

- Chain Replication: Chain Replication is a specialization of the primary backup approach that allows building a data store that provides high throughput and availability. This approach assumes that replicas are structured in a series/chain. A chain consists of a set of nodes where each node has a successor and a predecessor except for the first (head) and last (tail) nodes of the chain.

The following points are to be considered for implementing replication [14]:

- The time of creation of the replicas
- The files that are to be replicated
- The placement of the replicas

The goals of replication:

- Decrease latency time
- Reduce the bandwidth
- Balance the workload

- Minimize execution time
- Minimize the maintenance cost
- Achieve high scalability and availability
- Fault tolerant

4. REPLICATION ALGORITHMS [14]

- **Best Client:** A replica is created at the client that has generated the most requests for a file. This client is called the best client. At a given time interval, each node checks to see if the number of requests for any of its file has exceeded a threshold. If it has exceeded then the best client for that file is identified.
- **Cascading Replication:** It supports tree structure. The data files are generated in the top level. Once the number of accesses for the file exceeds the threshold, then a replica is created at the next level. This is created on the path to the best client. The same continues on all levels until it reaches the best client itself.
- **Fast Spread:** In this method a replica of the file is stored at each node along its path to the client. When a client requests a file, a copy is stored at each layer on the way. This leads to a faster spread of data. When a node does not have enough space for a new replica it deletes the least popular file that had come in the earliest.
- **Least Frequently Used (LRU):** This strategy always replicates files to local storage system. In LRU strategy the requested site caches the required replicas. If the local storage is full, the oldest replica in the local storage is deleted in order to free the storage. However, if the oldest replica size is less than the new replica, the second oldest file is deleted and so on.
- **Bandwidth Hierarchy Replication (BHR):** It is a novel dynamic replication strategy which reduces data access time by avoiding network congestions in a data grid network. This strategy benefits from "network-level locality" which represents that required file is located in the site which has broad bandwidth to the site of job execution.
- **Weight Based Dynamic Replica Replacement:** This strategy calculates the weight of replica based on the access time in the future time window on the last access history. After that, calculate the access cost which embodies the number of replicas and the current bandwidth of the network. The replicas with high weight will be helpful to improve the efficiency of data access, so they should be retained and then the replica with low weight will not make sense to the rise of data access efficiency, and therefore, should be deleted.
- **Agent-based replica placement algorithm:** This determines the candidate site for the placement of replica. For each site that holds the master copies of the shared data files will deploy an agent. The main objective of an agent is to select a candidate site for the placement of a replica that reduces the access cost, network traffic and cumulative response time for the applications. Furthermore, in creating the replica an agent prioritizes the resources in the grid based on the resource configuration, bandwidth in the network. It insists for the replica at their sites and then creates a replica at suitable resource locations.
- **Efficient Replacement Strategy:** It is a replication strategy for dynamic data grids, which take into account the dynamic of sites. This strategy can increase the file availability, improved the response time and can reduce the bandwidth consumption. Moreover, it exploits the replicas placement and file requests in order to converge towards a global balancing of the grid load. This strategy will focus on read-only-access as most grids have very few dynamic updates because they tend to use a load rather than update strategy.

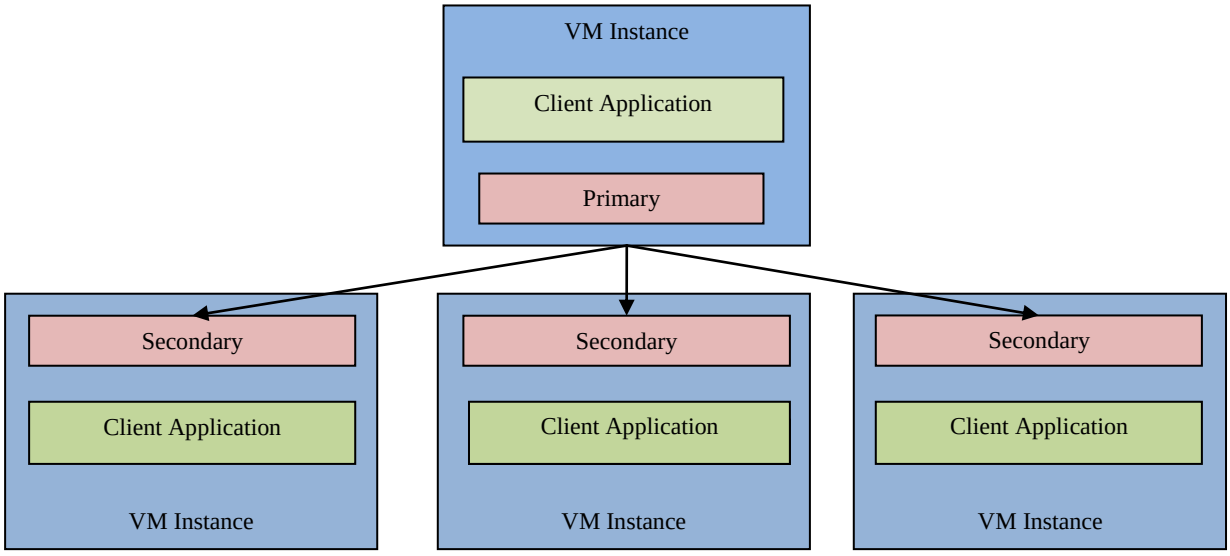

Fig 2: Replication with Fault Tolerance

Table I gives a comparative study of the different replication strategies [14]

Table I: Comparison of the Replication Algorithms

Replication Technique	Method	Performance Metric	Type	Scalability
Best Client	Replicates file to site that generates maximum number of requests	Response time , Bandwidth conservation	Dynamic	Medium
Cascading	If number of requests exceeds threshold then replica trickles down to lower tier	Response time , Bandwidth conservation	Dynamic	Medium
Fast Spread	If a client requests a file then a replica of file stores at each node along the path toward the client	Response time , Bandwidth conservation	Dynamic	Medium
Caching	A requesting client receives the file and stores a replica of it locally	Response time , Bandwidth conservation	Static	Medium
Least Frequently Used (LFU)	Always replicates files to local storage , if no space : delete least accessed files	Job execution time	Static	High
Least Recently Used (LRU)	Always replicates files to local storage , if no space : delete oldest file in the storage	Job execution time	Static	High
Weight-based replication]	Calculates the weight of replica based on the access time in the future time window, based on the last access history	Effective network usage, Mean job execution time	Static	Medium
Agent based replication	An agent holds the master copy for each site, select a candidate site for the placement of replica that exceeds the conditions	Execution time test, Data availability test	Static	Low
Efficient replication strategy	Takes into account the dynamic of sites. Exploits the replicas placement and file request in order to converge towards a global balancing of grid load	Response time, Effective Network Usage	Dynamic	Medium
Enhanced Fast Spread (EFS)	Uses a dynamic threshold that determines if the requested replica should be stored at each node along its path to the requester. Keeps only the important replicas while other less important replicas are replaced with more important replicas	Total response time, Total bandwidth consumption	Dynamic	Medium
Bandwidth Hierarchy Replication (BHR)	Replicates files which are likely to be used frequently within the region in near future	Total job execution time	Static	Medium

4.1 Implementation Methodology

The requirement for high availability solutions has generated a lot of design policies to provide a consistent service. These approaches involve systems such as heartbeat and cluster computing. The basic mechanism for synchronizing two or more servers and detecting server failures is the heartbeat. It monitors the data flow on a network shared by a pair of servers. Heartbeat is used in a system with many nodes. One machine is designated as the primary node and the other nodes are considered the secondary nodes. It is a signal which is sent from all the nodes at regular intervals so that each system will come to know the status of availability of other node. If the primary node fails or requires downtime, the secondary node can take over the primary role. This concept of heartbeat is used in clustered computing. A computer cluster is a group of connected computers. They work together as a single computer. Clusters are usually set up to improve performance and/or availability over that of a single computer. They are more cost-effective than single computers of comparable speed or availability.

5. CONCLUSION

Cloud is a form of distributed computing that is gaining high popularity. With most of the applications deployed in cloud, the number of users accessing the cloud has increases exponentially. Scalability and availability are required for such applications. Replication of the services is one of the ways to achieve these two parameters. To implement replication and availability fault tolerance is one of the important concepts. This can be implemented by making use of heartbeat. This idea of heartbeat in cluster computing can be used in cloud computing. This is based on an active-passive high availability solution. Each service under high availability needs at least two identical servers: a primary host, on which the service run, one or more secondary hosts, able to recover the application in less than one second. A heartbeat system is used to monitor the health of the nodes in the cluster. It is able to detect errors or failures on one node and automatically transfer workload or applications to another active node in the same cluster. Such systems typically have redundant hardware and software that make the applications available, despite failures.

6. BIBLIOGRAPHY

[1] Dejene Boru, Dzmitry Kliazovich, Fabrizio Granelli, Pascal Bouvry, and Albert Y. Zomaya, "Energy Efficient Data Replication in Cloud Computing Data Centers", Eco Cloud Project.

[2] D. Kliazovich, J. E. Pecero, A. Tchernykh, P. Bouvry, S. U. Khan, and A. Y. Zomaya, "CA-DAG: Communication-Aware Directed Acyclic Graphs for Modeling Cloud Computing Applications," IEEE International Conference on Cloud Computing (CLOUD), Santa Clara, CA, USA, 2013.

[3] Xavier D´efago, Andr´e Schiper and Nicole Sergent, "Semi-Passive Replication", IEEE Computer Society Press. Appeared in Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems, 1998.

[4] Manjula Dyavanur and Kavita Kori, "Fault Tolerance Techniques in Big Data Tools: A Survey", International Journal of Innovative Research in Computer and Communication Engineering IJIRCCE, Vol.2, Special Issue 2, May 2014, ISSN (Online): 2320-9801.

[5] R. Jemina Priyadarsini and L. Arockiam, "Failure Management in Cloud: An Overview", International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 10, October 2013, ISSN (Print): 2319-5940, ISSN (Online): 2278-1021.

[6] Sergio Filipe Garrau dos Santos Almeida, "Geo Replication in Large Scale Cloud Computing Applications", Dissertation submitted for Master Degree in Information Systems and Computer Engineering, November 2012.

[7] Adrian Coles and Bica Mihai, "An Adaptive Virtual Machine Replication Algorithm for Highly- Available Services", Proceedings of the Federated Conference on Computer Science and Information Systems pp. 941–948, ISBN 978-83-60810-22-4, 978-83-60810-22-4/$25.00 c 2011 IEEE.

[8] Rodrigo N. Calheiros and Rajkumar Buyya, "Meeting Deadlines of Scientific Workflows in Public Clouds with Tasks Replication", IEEE Transactions On Parallel And Distributed Systems, Vol. 25, No. 07, September 2013, Digital Object Indentifier 10.1109/TPDS.2013.238, 1045-9219/13/$31.00 © 2013 IEEE.

[9] C. Engelmann, S. L. Scott, C. Leangsuksun and X. He, "Towards High Availability for High-Performance Computing System Services: Accomplishments and Limitations", This work was done at Oak Ridge National Laboratory and Tennessee Tech University and was partially sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory.

[10] Dhananjaya Gupt, Mrs.Anju Bala, "Autonomic Data Replication in Cloud Environment", International Journal of Electronics and Computer Science, IJECSE, Vol.2 No.2, ISSN 2277-1956/V2N2-459-464.

[11] Sushant Goel and Rajkumar Buyya, "Data Replication Strategies in Wide Area Distributed Systems", Grid Computing and Distributed Systems Lab, University of Melbourne, Australia.

[12] Laiping Zha, Yizhi Ren, Yang Xiang, and Kouichi Sakurai, "Fault-Tolerant Scheduling with Dynamic Number of Replicas in Heterogeneous Systems", 2010 12th IEEE International Conference on High Performance Computing and Communications, 978-0-7695-4214-0/10 $26.00 © 2010 IEEE, DOI 10.1109/HPCC.2010.72.

[13] Bakhta Meroufela, Ghalem Belalem, "Managing Data Replication and Placement Based on Availability", 2013 AASRI Conference on Parallel and Distributed Computing Systems, AASRI Procedia 5 (2013) 147 – 155 © 2013 The Authors. Published by Elsevier B.V.

[14] Sheida Dayyani and Mohammad Reza Khayyambashi, "A Comparative Study of Replication Techniques in Grid Computing Systems", (IJCSIS) International Journal of Computer Science and Information Security, Vol. 11, No. 9, September 2013.

[15] Arindam Das and Ajanta De Sarkar, "On Fault Tolerance of Resources in Computational Grids", International Journal of Grid Computing and Applications, Vol.3, No.3, Sept. 2012, DOI: 10.5121/ijgca.2012.3301.

[16] Thakur Kapil Singh, Godavarthi Tarakarama Ravi Teja and Padmavathi Srinivasa Pappala, "Fault Tolerance-Challenges, Techniques and Implementation in Cloud Computing", International Journal of Scientific and Research Publications, Vol.3, Issue 6, June 2013, ISSN 2250-3153.