International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2015): 6.391

Optimization of Data Storage and Fault Tolerance Strategies in Cloud Computing

Dr. C. Vijay Kumar

Department of Computer Applications, MITS College, Madanapalle, India

Abstract: ***Internet provides to us content in the forms of videos, emails and information served up in web pages. With Cloud Computing, the next generation of Internet will allow us to "buy" IT services from a web portal, drastic expanding the types of merchandise available beyond those on e-commerce sites such as eBay, Flipkart and Snapdeal. Our focus is to improve the performance and optimize the storage usage by providing the DaaS on the cloud, 1) Analyzing and modeling the relationship between system availability and the number of replicas 2) Evaluating and identifying the popular data and triggering a replication operation when the popularity data passes a dynamic threshold 3) Calculating a suitable number of copies to meet a reasonable system byte effective rate requirement and placing replicas among data nodes in a balanced way 4) Designing the optimized data replica algorithm in a cloud. Experimental results shows the efficiency and effectiveness of the replication by the proposed strategy in a cloud.***

Keywords: Cloud Computing, DaaS, Replications, Virtualization, optimized data replica

1. Introduction

Cloud computing has gained significant traction in recent years. Companies such as Google, Amazon, IBM, Facebook and Microsoft have been building massive data centers over the past few years. These data centers tend to be built out of commodity desktops with the total number of computers managed by these companies being in the order of millions. By leveraging economies of scale, these data centers can provision cpu, networking, and storage at substantially reduced prices which in turn underpins the move by many institutions to host their services in the cloud. The cloud provides a variety of services for its users such as providing software as a service (SaaS), infrastructure as a service (IaaS), and platform as a service (PaaS). Providing infrastructure as a service within the cloud includes providing data (DaaS) or computational resources services[1]. There are many areas that can be improved in the cloud services such as the security of the data, the reliability of the cloud services, and the usage of distributed third party servers without including the client in the back-end processes. Moreover, a problem that still needs to be addressed for providing DaaS in the cloud is improving the techniques used to make the data available and quickly accessible (and downloadable) for the client. Another problem is data redundancy on the cloud servers, which is resulting in a huge increase of storage needs.

Database Replication is the frequent electronic copying of data from a database in one computer or server to a database in another so that all users share the same level of information. The result is a distributed database in which users can access data relevant to their tasks without interfering with the work of others. However data replication is a fascinating topic for both theory and practice. On the theoretical side, many strong results constraint what can be done in terms of consistency: e.g., the impossibility of reaching consensus in asynchronous systems the blocking nature of CAP theorem, and the need for choosing a suitable correctness criterion among the many possible[6] .On the practical side, data replication plays a key role in a wide range of contexts like caching, back-up, high availability, wide area content distribution, increasing scalability, parallel processing, etc. Finding a replication solution that is suitable in as many such contexts as possible remains an open challenge.

2. Virtualization

Virtualization, automation and standards are the pillars of the foundation of all good cloud computing infrastructures. Without this foundation firmly in place across the servers, storage and network layers, only minimal improvements on the adoption of cloud services can be made; conversely, with this foundation in place, dramatic improvements can be brought about by "uncoupling" applications and services from the underlying infrastructure to improve application portability, drive up resource utilization, enhance service reliability and greatly improve the underlying cost structures. However, this "uncoupling" must be done harmoniously such that the network is "application aware" and that the application is "network aware"[2]. Specifically, the networks both the data center network and the data center interconnect network need to embrace virtualization and automation services. The network must coordinate with the upper layers of the cloud to provide the needed level of operational efficiency to break the lock between IT resources in today's client-server model. The advantage of cloud computing is the ability to virtualize and share resources among different applications with the objective for better server utilization. In non-cloud computing three independent platforms exist for three different applications running on its own server. In the cloud, servers can be shared, or virtualized, for operating systems and applications resulting in fewer servers.

3. Fault Tolerant Strategies

The most dominant storage and fault tolerant strategies that are currently being used in cloud computing settings are several unifying themes that underlie the systems

Theme 1: Voluminous Data
The datasets managed by these systems tend to be extremely voluminous. It is not unusual for these datasets to be several terabytes. The datasets also tend to be generated by programs, services and devices as opposed to being created by a user one character at a time. In 2000, the Berkeley "How Much Information?" reported that there was an estimated 25–50 TB of data on the web. In 2003 the same group reported that there were approximately 167 TB of information on the web. The Large Hadron Collider (LHC) is expected to produce 15 PB/year The amount of data being generated has been growing on an exponential scale there are growing challenges not only in how to effectively process this data, but also with basic storage[4].

Theme 2: Commodity Hardware
The storage infrastructure for these datasets tend to rely on commodity hard drives that have rotating disks. This mechanical nature of the disk drives limits their performance. While processor speeds have grown exponentially disk access times have not kept pace. The performance disparity between processor and disk access times is in the order of 14,000,000:1 and continues to grow.

Theme 3: Distributed Data
A given dataset is seldom stored on a given node, and is typically distributed over a set of available nodes. This is done because a single commodity hard drive typically cannot hold the entire dataset. Scattering the dataset on a set of available nodes is also a precursor for subsequent concurrent processing being performed on the dataset.

Theme 4: Expect Failures
Since the storage infrastructure relies on commodity components, failures should be expected. The systems thus need to have a failure model in place that can ensure continued progress and acceptable response times despite any failures that might have taken place. Often these datasets are replicated, and individual slices of these datasets have checksums associated with them to detect bit-flips and the concomitant data corruptions that often taken place in commodity hardware.

Theme 5: Tune for Access by Applications
Though these storage frameworks are built on top of existing file systems, the stored datasets are intended to be processed by applications and not humans. Since the dataset is scattered on a large number of machines, reconstructing the dataset requires processing the metadata fcc to identify the precise location of specific portions of the datasets. Manually accessing any of the nodes to look for a portion of the dataset is futile since these portions have themselves been modified to include checksum information.

Theme 6: Optimize for Dominant Usage
Another important consideration in these storage frameworks is optimizing the most general access patterns for these datasets. In some cases, this would mean optimizing for long, sequential reads that puts a premium on conserving bandwidth while in others it would involve optimizing small, continuous updates to the managed datasets[8].

Example, let's we take partition-tolerant system with two nodes **A** and **B**. Let's suppose there is some network error between **A** and **B**, and they can no longer communicate with each other, but both can still connect to clients. If a client were to write a change a file v hosted on both **A** and **B** while connected to **B**, the change would go through on **B**, but if the client later connects to **A** and reads v again, the client will not see their changes, so the system is no longer consistent. You could get around this by instead sacrificing availability if you ignore writes during a network partition.

4. Google File System

The Google File System (GFS) is designed by Google to function as a backend for all of Google's systems. The basic assumption underlying its design is that components are expected to fail. A robust system is needed to detect and work around these failures without disrupting the serving of files. GFS is optimized for the most common operations as long, sequential and short, random reads, as well as large, appending and small, arbitrary writes. The major goal in designing GFS was to efficiently allow concurrent appends to the same file. As a design goal, high sustained bandwidth was deemed more important than low latency in order to accommodate large datasets [9].

In a GFS cluster, there are three components, multiple clients, a single master server, and multiple chunk servers, as shown in Figure.1. Files are stripped into one or many fixed size chunks, and these chunks are stored in the data centers, which are managed by the chunk servers. The master server maintains all the meta data of the file system, including the namespace, the access control information, the mapping from files to chunks, and the current locations of chunks. Clients interact with the master for metadata operations, but all data bearing communication goes directly to the chunk servers. There are usually also several master replicas, as well as shadow masters which can handle client reads to help reduce load on a master server. The chunk servers hold data in 64 MB-sized chunks[11].

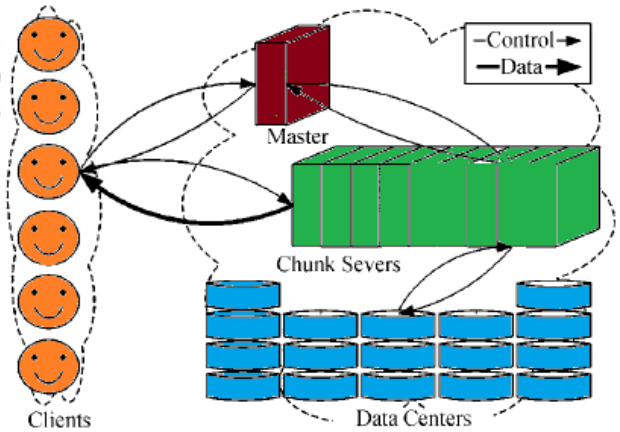

Figure 1: GFS architecture

4.1 Check Pointing

In GFS, the master server will keep logs tracking all chunk mutation. Once a log file starts to become too big, the master server will create a checkpoint. These check-points can be

used to recover a master server, and are used by the master replicas to bring a new master process up[16].

4.2 Replication

By default, all GFS maintains a replication level of 3. Users can designate different replication levels for different regions of the file namespace. For example, a temp directory generally has a replication level of 1, and is used as a scratch space[14]. The master server is responsible for ensuring that the replication level is met only involves copying over chunks if a chunk server goes down, but also removing replicas once a server comes back up. The master server will try to place replicas on different racks.

4.3 Failures

The master server regularly exchanges heart beats with the chunk servers. If the master server does not receive a heartbeat from a chunk server in time, it will assume the server has died, and will immediately start to spread the chunks located on that server to other servers to restore replication levels a chunk server recover, it will start to send heart beats again and notify the master that it is back up. At this point the master server will need to delete chunks in order to drop back down to replication level. Because of this approach, it would be possible to wreak havoc with a GFS instance by repeatedly turning on and off a chunk server. Master server failure is detected by an external management system. Once this happens, one of the master server replicas is promoted, and the master server process is started up on it[17-18].

5. Optimized Data Replication Strategy

The Optimized Data Replication Strategy ODRS has three important phases

1) Which data file should be replicated and when to replicate in the cloud system to meet clients requirements such as waiting time reduction and data access time.
2) How many suitable new replicas should be created in the cloud system to meet a given availability requirement.
3) Where the new replicas should be placed to meet the system task successful execution rate and bandwidth consumption requirements.

5.1 Data Replica Status

Given the fact that a more recently accessed data file might be accessed again in the near future according to the current status of data access pattern, which is called temporal locality, a ranking data file is determined by analyzing the access to the data from users. When the ranking of a data file passes a dynamic threshold, the replication operation will be triggered.

Ranking Degree: The ranking degree of a block bk is defined as the access frequency based on time factor. During the period from the start time ts to the present time tp, the popularity degree Rk of a block bk can be calculated by

$$b_k = \sum_{ti=ts}^{tp} (ti, ti+1)\ X\ (ti * tp)$$

where (t_i, t_i+1) is the number of accesses during the time interval t_i to t_i+1.

Replica Factor: The replica factor is defined as the ratio of the ranking degree Rk and the total number of bytes of data file f_i requested by all tasks under given constraints. It is used to determine whether the data file f_i should be replicated, denoted as RF_i.

$$RFi = \frac{Rk}{(Fi * RNi)}$$

Where Rk is Ranking degree , Fi is file size and RNi is number of replications .

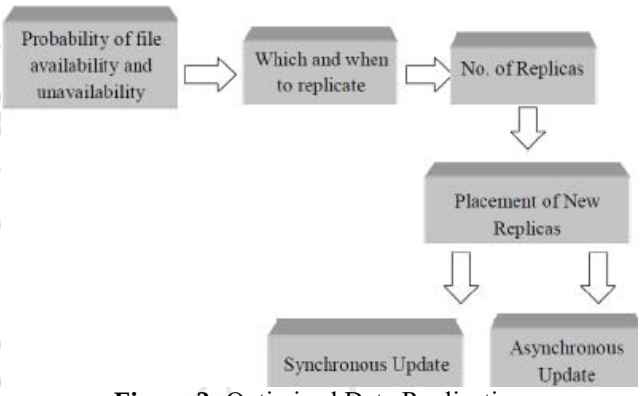

Figure 2: Optimized Data Replication

5.2 New Replication Placement Allocation

To meet the system task successful execution rate and bandwidth consumption requirement, different tiers of data centers which have the selected replica data file *fi* will decide the replica placement and the placement of new replicas to be created according to the access information of directly connected data centers. The number of new replicas created at the directly connected data center *dck* is calculated based on the total number of new replicas *bni*(*inc*)

$$bn_i = \frac{RF(dck)}{RFi}\ x\ \ dck$$

where *bni*(*dck*) is the number of new replicas to be created at the directly connected data center *dck*, *RFi*(*dck*) is the replica factor of data file *fi* of the datacenter *dck* directly connected, and *RFi* is the replica factor of the data file *fi*.

Method Upload File To Cloud

(1) Require: FileSize i, FileName
(2) Declare: BlockSize,
(3) While i between {0, (i/NOC) i NOS}
(4) IF i % FileSize i
(5) BlockSize = i
(6) End IF
(7) EndWhile
(8) InsertRow FileID, FileName, FileSize, BlockSize
(9) End of method

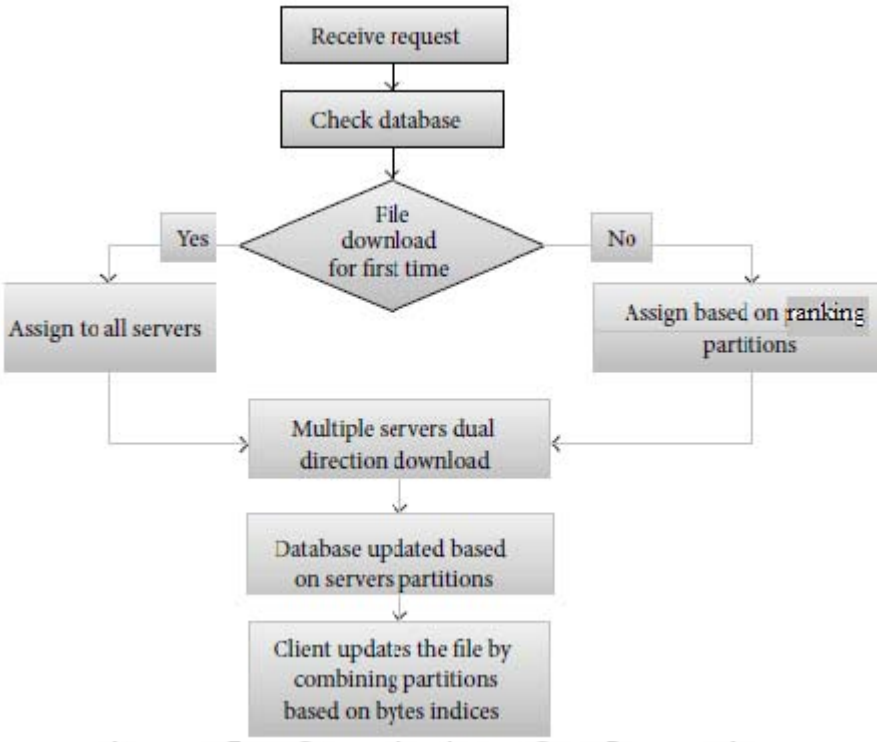

Figure 3: Optimized Data Replication Algorithm (ODRA)

6. Optimized Data Replication Algorithm (ODRA)

Algorithm. ODR Algorithm
Input: the available probability *p*(*bak*) and unavailable probability *p*(*bak*) of all replicas of block *bk* of the data file *fi*, the number of replicas *bni*, the block size *bsi* and the number of accesses *ank*(*ti; ti*+1) within time interval *ti* to *ti*+1 for each block of data file *fi*.
Output: system byte effective rate *R*(*SBER*).
Step 1 : Initialize available and unavailable probability of each replica of block *bk p*(*bak*) and *p*(*bak*).
Step 2 : for each data file *fi* at all data centers *DC* do
Step 3 : Calculate the ranking degree *Rk* of a block *bk* of data fi le *fi* by .
Step 4 : Calculate replica factor *RFi* of data file *fi* . end for
Step 5 : Calculate replica factor *RFsys* of the cloud.
Step 5 : for each data ¯le *fi* at all data centers *DC* do
Step 6 : if *RFi* > min (1 + *bck*) , *RFsys* , max *RFk* then
Step 7 : The replication operation of the data file *fi* will be triggered. end if
Step 8 : end for
Step 9 : for each data file *fi* at all data centers *DC* do
Step 10 : Calculate the old fule availability *P*(*FAi*) of data file *fi* by .
Step 11: end for
Step 12 : for each triggered data file do
Step 13: Calculate the new file availability *Pnew* (*FAi*) of data file *fi* .
Step 14 : Calculate the number of new replicas needed *bni*(*inc*) .
Step 15 : end for
Step 16 : for each triggered data file and *bni*(*inc*) > 0 do
Step 17 : for each directly connected data center *dck* do
Step 18: Calculate the number of new replicas *bni*(*dck*) to be created at the directly connected Step 19 : data center *dck* .
Step 20: end for
Step 21: Determine the replica placement according to *bni*(*dck*).
Step 22: if the storage space of the target data center *DCobj* is not enough then
Step 23: Quick sort all data file in descending order by replica factor of data file at data center *DCobj* .
Step 24: Delete the data file with the smallest replica factor. end if
Step 25: Calculate the new system byte effective rate *R*(*SBER*)
Step 26: returur R(SBER)

7. Conclusion

Storage optimization is an important issue as the demand for storage space especially for big data is rapidly increasing. However, data replication is also important for various reasons such as increasing reliability, fault tolerance, and enhanced performance and also for backup purposes. High availability, high fault tolerance and high efficiency accesses to Internet based cloud data centers where failures are normal rather than exceptional are significant issues, and are often considered more valuable than high performance. Data replication allows reducing user waiting time and speeding up data access. It increases data availability by providing users with different replicas of the same service, and all of them in a coherent state.

8. Acknowledgment

This research was supported by the Madanapalle Institue of Technology and Science, Madanapalle (UGC-Autonomous), Andhra Pradesh.

References

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," *National Institute of Standards and Technology*, vol. 53, no. 6, p. 50, 2009.

[2] J. Wu, L. Ping, X. Ge, Y. Wang, and J. Fu, "Cloud storage as the infrastructure of Cloud Computing," in *Proceedings of the International Conference on Intelligent Computing and Cognitive Informatics (ICICCI '10)*, pp. 380–383, IEEE, June 2010.

[3] K. Al Nuaimi, N. Mohamed, M. Al Nuaimi, and J. Al-Jaroodi, "A survey of load balancing in cloud computing: challenges and algorithms," in *Proceedings of the IEEE 2nd Symposium on Network Cloud Computing and Applications (NCCA '12)*, pp. 137–142, London, UK, December 2012.

[4] W. Zeng, Y. Zhao, K. Ou, and W. Song, "Research on cloudstorage architecture and key technologies," in *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology Culture and Human (ICIS '09)*, pp. 1044–1048, November 2009.

[5] J. Al-Jaroodi and N. Mohamed, "DDFTP: dual-direction FTP," in *Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid '11)*, pp. 504– 513, May 2011.

[6] N. Mohamed and J. Al-Jaroodi, "Delay-tolerant dynamic load balancing," in *Proceedings of the 13th IEEE International Conference on High Performance Computing and Communications (HPCC '11)*, pp. 237–245, IEEE, Banff, Canada, September 2011.

[7] N. Mohamed, J. Al-Jaroodi, and A. Eid, "A dual-direction technique for fast file downloads with dynamic load balancing in the Cloud," *Journal of Network and Computer Applications*, vol. 36, no. 4, pp. 1116–1130, 2013.

[8] C.Vijaya Kumar and Dr.G.A Ramachandra, "Energy Conservation for Datacenters in Cloud Computing using Genetic Algorithms, pp. 577–583, International Journal of Science and Research (IJSR), December 2014.

[9] L.Wang,G.VonLaszewski, A. Younge et al., "Cloud computing: a perspective study," *New Generation Computing*, vol. 28, no. 2, pp. 137–146, 2010.

[10] E. Hamburger, *Google Drive vs. Dropbox, SkyDrive, SugarSync, and Others: A Cloud Sync Storage Face-Off*, The Verge, 2012.

[11] Shvachko K, Hairong K, Radia S, Chansler R. The Hadoop distributed ¯le system. In *Proc. the 26th Symposium on Mass Storage Systems and Technologies*, Incline Village, NV, USA, May 3-7, 2010, pp.1-10.

[12] Wang S S, Yan K Q, Wang S C. Achieving e±cient agreement within a dual-failure cloud-computing environment. *Expert System with Applications*, 2010, 38(1): 906-915.

[13] C.Vijaya Kumar and Dr.G.A Ramachandra, Optimization of Large Data in Cloud computing using Replication Methods, International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 3034-3038.

[14] Kim Y H, Jung M J, Lee C H. Energy-aware real-time task scheduling exploiting temporal locality. *IEICE Transactions on Information and Systems*, 2010, 93(5): 1147-1153.

[15] Wei Q, Veeravalli B, Gong B, Zeng L, Feng D. CDRM: A cost-effiective 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.

[16] Bonvin N, Papaioannou T G, Aberer K. A self-organized, fault-tolerant and scalable replication scheme for cloud sto- rage. In *Proc. the 1st ACM Symposium on Cloud Computing*, Indianapolis, IN, USA, June 10-11, 2010, pp.205-216.

[17] C.Vijaya Kumar and Dr.G.A Ramachandra, Thrusting Energy Efficiency for Data center in Cloud Computing Using Resource Allocation Techniques,International Journal of Emerging Trends & Technology in Computer Science (IJETTCS),ISSN 2278-6856, pp 173-185.

[18] Nguyen T, Cutway A, Shi W. 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.

[19] Dr.K. Fayaz, Dr.C. Vijay Kumar, Optimization of Data Storage and QOS in Data Replication using the Priority Datasets, Volume 5, Number1, January-March'2016.