A UNIQUE REPLICA REPLACEMENT METHOD THAT CONSIDERS FILE AVAILABILITY AMONG SOCIAL NETWORK USERS

VENKATASUBRAMANYAM Y [1] K ANUSHA [2] M.ANANTHA LAKSHMI [3] B.ANJANEYULU [4]

[1]ASSOC.PROFESSOR, DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,
[2] ASST. PROFESSOR, DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,
[3]ASST. PROFESSOR, DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,
[4]ASST.PROFESSOR, DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,
[1,2,3,4] SRI MITTAPALLI COLLEGE OF ENGINEERING

Abstract:

An increasing number of academics and businesspeople are using the term "cloud computing" in their discussions. When it comes to the cloud, replica management is a top concern as it provides reliable, high-availability, and quick data access times. Assuming a sufficiently dispersed set of requests and replicas, keeping all replicas active may increase the successful execution rate of system tasks. It is far more difficult, however, to deploy replicas appropriately in large-scale, dynamically expandable, fully virtualized data centers. A novel replica placement is suggested to provide cost-effective availability, minimize application response time, and make load balancing for cloud storage. Five crucial metrics—mean service time, failure probability, load variation, latency, and storage usage—form the basis for replica placement. Because each site has a limited amount of storage space, replication should be handled with caution. Because of this, the site can only save the most crucial copies. In addition, we provide a novel approach to replacing replicas that takes into account the file's availability, the amount of access, the size of the copy, and the last time the replica was requested. In compared to other methods, ours provides greater performance in terms of mean response time, effective network utilization, load balancing, replication frequency, and storage consumption, as evaluated using the CloudSim simulator.

Introduction:

There is enough data produced in only two days1 to fill an entire database with every word ever uttered by humans. In addition, by 2025, the data created would have surpassed 175 zettabytes2, according to the most recent study by International Data Corporation (IDC) [13]. A wide variety of sources, including sensors, social media, and scientific models, provide this data. Businesses and academic institutions are both enriched by this data flood, which in turn affects our daily lives. For instance, Argonne National Laboratory runs complicated universe simulations to expand human understanding, while Google and Facebook analyze their daily acquired data to enhance users' experiences [5]. Therefore, it is critical to provide effective and scalable data management in order to convert these massive data sets into actionable insights. Massive infrastructures and scalable data management strategies are required to handle the massive amounts of Big Data. As time has progressed, cloud computing has become the standard platform for applications that rely heavily on data. Cloud computing gives the impression of having limitless, inexpensive resources at one's fingertips. In an effort to make Big Data administration easier, cloud providers like Amazon, Microsoft, and Google have recently deployed infrastructures with millions of servers spread throughout the globe. Take Amazon

Web Services (AWS) as an example; they host about 5 million servers in total [144]. 1 Daily production is estimated at 2.5 exabytes [11], while the total amount of all human uttered words is 5 exabytes [11]. With millions of services going live daily, there are hundreds of data centers across five continents housing 21 zettabytes, or 1021 bytes, or 1 million petabytes [6]. Data storage and access solutions on a massive scale have been mushrooming in recent years, with the advent of cloud computing. In order to provide their combined storage capabilities and dependable, rapid data access, they operate on top of thousands of computers. One example is the cloud-based Windows Azure Storage (WAS) system, which processes over two billion transactions daily and hosts over an exabyte of data [2, 4]. Hadoop [19] and Spark [20], two popular Big Data analytics frameworks, provide a perfect foundation for supporting distributed infrastructures and storage systems, allowing for efficient running of data-intensive applications. For example, Facebook's data centers do over one million Hadoop tasks every month, handling data quantities in the tens of petabytes range [9]. Virtual machines (VMs) or containers are the standard means of deploying cloud services, including data analytics services (such as Amazon Elastic MapReduce [14], Microsoft Azure HDInsight [12], etc.). In order to provide a service for use with cloud applications, the host server must have the service image saved locally. This image contains the operating system, the service software, and all of its dependencies. If not, the provisioning time will be extended while the matching picture is sent to the target server via the network. Service image management is crucial for cloud service provisioning due to the growing size and quantity of service images, as well as the rising demand for fast service provisioning (for example, AWS offers over 20,000 unique public images3, and their sizes could reach dozens of gigabytes [6]). In addition, there are additional difficulties in supplying virtual machines and containers due to the widespread use of edge computing, which allows in-place processing (i.e., by relocating computation near data sources) and the present trend towards multi-site deployment, which is made possible by geographically dispersed clouds [12]. This is because to the storage capacity constraints in Edge-servers, the restricted bandwidth in wide area network (WAN) connections, and the heterogeneity of these connections. Management of virtual machines (VMs) and container images must be efficient and scalable if distributed cloud and edge system service provisioning is to be facilitated quickly and easily. Distributed storage systems have long relied on replication to keep data accessible. Furthermore, Spark [20], Flink [18], and Hadoop [19] are data analytics frameworks that make heavy use of replication. This helps with machine failures by allowing tasks to be re-executed using other copies of the data [14] and with improving the performance of cloud-based data-intensive applications by increasing the likelihood of scheduling computation tasks on the machine that hosts the input data [12]. However, replication has grown costly in terms of storage and hardware due to the widespread use of high-speed but expensive storage devices (such as SSDs and DRAMs) in storage systems and the incessant expansion of Big Data [10]. On the other hand, erasure codes (EC) provide minimum storage overhead while providing great data availability. As a result, many distributed storage systems are using them right now [11]. For instance, in comparison to replication, Microsoft's cloud-based object store experiences a storage overhead reduction of over 50% when EC is used [13]. Although running data-intensive apps on EC might reduce storage cost, it could create a huge amount of data transmission as the input data for each compute activity (like Hadoop's map tasks) is spread out over several machines. In order to ensure rapid service delivery, the majority of major cloud providers make use of numerous geographically scattered data centres to duplicate their Virtual Machine Images (VMIs). This includes Amazon and Microsoft. The distribution of virtual machine images (VMIs) and the

available bandwidth between locations determine the logistics of service provisioning, which may include transferring VMIs across a wide area network (WAN). However, network heterogeneity in geographically dispersed clouds is ignored by current approaches to VMI retrieval, which in turn hinders VMI management. To address this, we developed, deployed, and assessed Nitro, an innovative VMI management system that aids in reducing VMI transfer times over heterogeneous WANs. Nitro uses two characteristics that work together to do this. When there are a lot of similarities inside and across photos, it uses deduplication to cut down on the quantity of data exchanged. Second, Nitro has a data transfer method that takes network intelligence into account, so it can speed up provisioning by taking advantage of networks with high bandwidth during data acquisition. Our network-aware data transmission method provides the best approach for getting VMIs with the least amount of overhead, according to the experimental findings. In addition, when compared to cutting-edge VMI storage solutions like OpenStack Swift, Nitro achieves performance improvements of up to 77%.

Related work

Internet protocols, P2P networks, ad hoc networks, sensor networks, and mesh networks have all made extensive use of replication technology to boost the efficiency of many applications in recent years [9, 10]. Grid [14] and Cloud [15], two examples of large-scale distributed systems, have rekindled interest in data replication as a field of study. There has been a lot of recent study on alternative replication algorithms, and they are especially relevant in cloud settings because to the large amounts of sophisticated scientific data and applications. A large number of businesses have established cloud computing infrastructures, including Amazon and Google. Clouds, in contrast to more conventional large-scale storage systems, are user-and workload-aware and primarily concerned with providing storage services over the Internet [16–18]. Cloud computing relies on distributed systems, such as GFS and HDFS, as its primary infrastructure and essential component.

Distributed storage systems employ data replication to decrease cloud system bandwidth usage, improve user waiting time, and increase data availability by distributing user tasks to several clones of the same service that are in a coherent state [19]. Many research have examined data replication and replica management in distributed systems as various technologies have progressed and developed. Static replication [20] and dynamic replication [2-5] are the two main categories of data replication techniques. In contrast to dynamic replication, which may dynamically create and delete duplicates in response to changing access patterns, static replication methods have the replication strategy pre-set. Reference [2] presents a GFS static distributed data replication approach. An approach to centralized data replication using p-median statistics is detailed in Reference [6]. In order to minimize the overall distance between the nodes that are seeking copies and the replication nodes that are keeping them, the p-median model finds p replica placement nodes.

To address this issue, Hussein et al. [7] designed an ARS for use in the cloud. Based on predictions of user access to each file's blocks, the approach examines how to improve the data files' dependability and how efficiently each file is accessible in the data centre. Furthermore, it uses heuristic search for each replication to dynamically install large-scale distinct file replicas on diverse data locations with low cost. The technique uses HLES time series to look at the files' recent data access history and identify the most popular ones for replication. Replication will

begin as soon as a popularity-based replication factor falls below a certain level. After doing a heuristic search for the optimal replication factor for each file, the adaptive method determines the appropriate replication site. According to the results of the experiments, the adaptive technique works as expected to increase the study cloud system's availability. The algorithm's one and only flaw is that it doesn't take replica placement's popularity level into account. On top of that, they disregard the system's load balance. An approach to enhance cloud data availability suggested by Rajalakshmi et al. [8] is DRSP, or dynamic replica selection and placement. The two primary components of the plan are the application of files and the replication procedure. In the first stage, you'll use the catalogue and index to find and create the replica. Whether you're keeping the file locally or remotely, the index will help you find it. The position of both the master and slave duplicates is also kept by the indexer. The algorithm is sent to a different site if the number of accesses exceeds the threshold value (Tα). Finding the value of the threshold (Tα) is the goal of Equation (1), which is the ratio of the total number of requests to the replication threshold (RT). The Tα value is equal to the RT value of the NumbReplica. (1) In the second stage, we check to see whether the destination has enough space to store the requested file. The suggested solutions are built in the cloud using the Eucalyptus platform. Data access speed and bandwidth utilization are both enhanced by the experimentally proven method of replica selection and placement, which transparently stores data in geographic places. Their main flaw is that they ignore most of the variables that really matter when deciding whether or not to replicate. In addition, they don't deal with the storage use problem or data file consistency; they only work on making the system run faster.

System model

Clouds are typically made up of massive data centres that use a lot of power but are necessary to meet the scalability and flexibility needs of their clients. A lot of these centres handle MapReduce and other large-scale data-intensive applications. Distributed storage systems like GFS and HDFS are foundational to cloud computing. Assumed in this work are m separate heterogeneous data nodes D1,..., Dj,..., Dm that make up the cloud storage cluster and hold a combined total of n distinct files f1,..., fi,..., fn. In Figure 1, you can see our replication management system paradigm. A total of m data nodes are assigned n files using the replication approach. The standard assumption in most file systems is that each access to file fi reads the whole file sequentially [3]. Since we won't be taking file partitioning into account, each file will need to occupy a whole data node. Since we can handle each file segment independently, this doesn't restrict the applicability of our technique. In a cloud storage cluster with set service times and Poisson arrival rates for file accesses, the issue we looked at is how to statically or dynamically assign files without partitions. Since no data consistency technique is required in this research, we only assume that all data are read-only. We provide an optimized data replication technique for cloud storage that aims to accomplish many objectives adaptively. In order to understand the connection between replica layout and these performances, we consider a number of characteristics like mean service time, load volatility, storage use, failure probability, and latency. We aim to optimize file availability, mean service time, load variation, and latency to discover various trade-offs within these goals. To discover the best replication factor and replication architecture for every file, we are using multi-objective optimization, which is necessary since there is more than one aim. We may then look for solutions that provide parameter values that are near to ideal thanks to this. This technique may be adjusted by users according to their requirements by defining weights, for example, giving greater weight to their

most important performance metrics. We will now describe our novel replication technique, ADRS, which incorporates both an efficient replica placement approach and a replica replacement strategy. Both components make up the ADRS: 1) Placing new copies at sites that dynamically adapt to current cloud settings is important to achieve the requirements of high availability, high fault tolerance, and high efficiency. It is possible that certain sites are heavily used while others lie idle due to the access skew caused by random replica placement. This might lead to low performance, poor parallelism, and an uneven distribution of loads across cloud storage. Nevertheless, it becomes much more complex when dealing with ultra-large-scale, dynamically scalable, fully virtualized data centers when it comes to proper duplicate placement. The generation and storage of fresh copies at the light-load locations may enhance the response time. It must take into account the following parameters when it is ready to produce a new replica and locate the optimal location for the replica:

• MST, or mean service time A system's processing pace may be seen in its mean service time. You may increase the average service time of the system by storing often used files in the sites with higher performance and seldom used files in the sites with relatively poor performance. The predicted service time of file fi on data node Nj (1 j m) is denoted as S T(i, j). When the file fi is present on data node Nj, the decision variable d(i, j) is set to 1, and when it is not, it is set to 0. Here, sizei is the size of file fi and tpj is the transfer rate of data node Nj. This may be expressed as S T(i, j) = d(i, j) × sizei tpj.

• LV, or load variance Variance in data node load measures the degree to which the system balances the load across all the data nodes in a cloud storage cluster. It is calculated as the standard deviation of data node load. Load balancing is considered successful when the variation between loads is small. The following is the definition of the load Load(i, j) of fi on the data node Nj, as the combination of the file's access rate and service time appropriately represents its load: The formula Load(i, j) = Acc(i, j) × S T(i, j)(7) may be used to the case when data node Nj requests file fi and the access rate of read requests is Acc(i, j). Due to poor performance in reading and writing operations under heavy demand, ADRS should disregard this site if the node is overwhelmed.

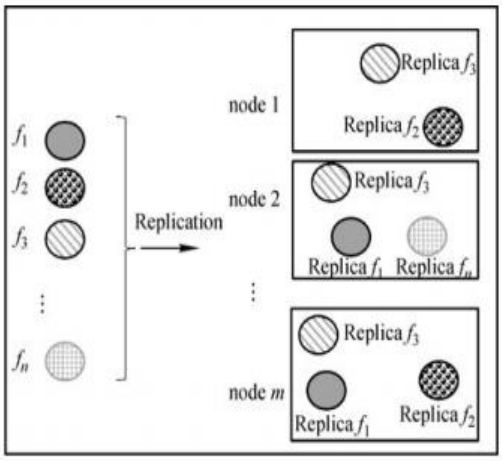


Fig. 1 System model of replication management

• Space utilisation To reduce wait times, it is recommended to store frequently used files on data nodes that have low storage utilisation.

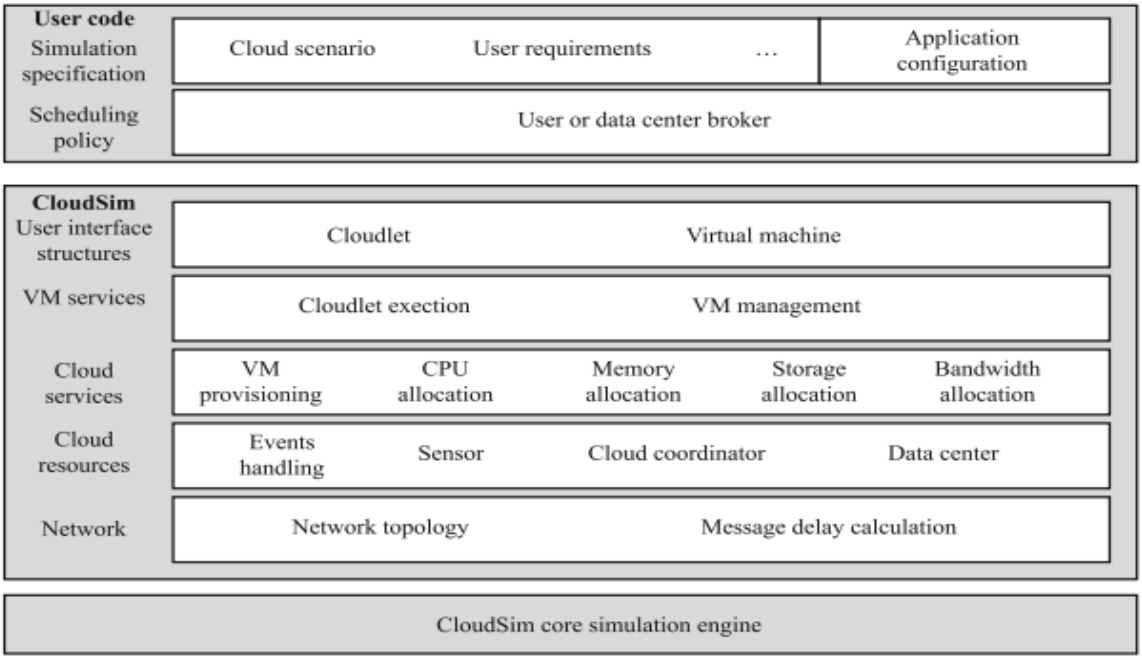
• The likelihood of failure (FP) If a node goes down, the requests will still be processed by creating a copy of the service on another node. Consequently, to reduce their delay, popular files should be stored on data nodes with a low failure risk.

• Wait time (L) Every storage system must prioritise minimising latency. Using high bandwidth capacity is essential for latency minimisation since channels with more bandwidth result in lower latency. Consequently, to

reduce their delay, data nodes with high bandwidth are preferable for storing files. Read latency is the only metric we used in our investigation.

Experimental results:

With the help of the CloudSim simulation layer, you can create virtualized cloud-based data center infrastructures complete with VM management interfaces, storage, bandwidth, and memory. Important concerns include administering applications, assigning hosts to virtual machines, performance and managing the status of the dynamic system. At this tier, a cloud



provider may put his rules into action if he wants to test out various tactics for virtual machine (VM) provisioning, or the process of assigning hosts to virtual machines. The basic VM provisioning capability may be programmatically extended to do such an activity. Regarding the supply of hosts to virtual machines, there is a clear difference at this tier. The quality of service (QoS) standards specified by a SaaS provider might dictate how many virtual machines (VMs) can operate on a single cloud server at the same time. At this level, you can see the features that a developer working on a cloud app may add to do advanced workload profiling and performance evaluations. At the very top of the CloudSim stack is the User Code, which supplies the fundamentals for hosts (machine count, specifications, etc.), applications (task count, requirements), virtual machines (VMs), users (user type, number of users), and broker scheduling methodologies. A developer of cloud applications can accomplish the following by working with the main entities provided at this layer: 1) build a variety of workload request distributions and application configurations; 2) model scenarios of cloud availability and conduct

robust tests using the custom setting; and 3) implement cloud and federation-specific application provisioning methods.

Conclusion

Because of its cheap cost, high availability, and scalability, cloud storage services are drawing more and more attention, contributing to the meteoric rise in popularity of cloud computing. One of the most popular ways to boost cloud storage performance is via replication. To help cloud computing data centres respond faster, we provide an adaptive data replication technique (ADRS). Improving cloud performance and load balancing is as simple as strategically placing replicas. Because of this, we have come up with a new replica placement strategy to help spread workloads among cloud nodes more effectively. Considering factors such as average service duration, failure probability, load volatility, latency, and storage utilisation, ADRS determines the optimal place to store replicas. In addition, we provide a novel approach to replacing replicas that takes into account the file's availability, the amount of access, the size of the copy, and the last time the replica was requested. A new feature called adaptive data replication has been added to the CloudSim platform via its extension. The five current algorithms that we examine are build-time, ARS, DRSP, CIR CDRM, and ADRS. Usage of storage, effective network utilisation, load variation, replication frequency, and mean response time are the performance indicators taken into account. When compared to the other algorithms, ADRS performs better in the experiments. In particular, as the number of tasks increases, its performance improves. The next stage of the project will investigate how well adaptive data replicas management works in a real-world cloud setting. Along with this, we want to investigate the trade-off issue in relation to quality factors like resource availability and cost.

References

1. Mi H B, Wang H M, Zhou Y F, Rung-Tsong Lyu M, Cai H, Yin G. An online service-oriented performance profiling tool for cloud computing systems. Frontiers of Computer Science, 2013, 7(3): 431–445
2. Fu X, Zhou C. Virtual machine selection and placement for dynamic consolidation in Cloud computing environment. Frontiers of Computer Science, 2015, 9(2): 322–330
3. Chen T, Bahsoon R, Tawil A R. Scalable service-oriented replication with flexible consistency guarantee in the cloud. Information Sciences, 2014, 264: 349–370
4. Wu H, Zhang W B, Zhang J H, Wei J, Huang T. A benefit-aware on-demand provisioning approach for multi-tier applications in cloud computing. Frontiers of Computer Science, 2013, 7(4): 459–474
5. Al-Fares M, Loukissas A, Vahdat A. A scalable, commodity data center network architecture. Computer Communication Review, 2008, 38: 63–74
6. Amazon-S3.Amazon simple storage service (Amazon s3). http://www.amazon.com/s, 2009
7. Ghemawat S, Gobioff H, Leung S. The Google file system. In: Proceedings of the 19th ACM Symposium on Operating Systems Principles. 2003

8. Calheiros R N, Ranjan R, Beloglazov A, Rose C, Buyya R. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience, 2011, 41(1): 23–50
9. Qiu L L, Padmanabhan V N, Voelker G M. On the placement of Web server replicas. In: Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies. 2001, 1587–1596
10. Aazami A, Ghandeharizadeh S, Helmi T. Near optimal number of replicas for continuous media in ad-hoc networks of wireless devices. In: Proceedings of the 10th International Workshop on Multimedia Information Systems. 2004
11. Intanagonwiwat C, Govindan R, Estrin D. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking. 2000
12. Tang B, Das S R, Gupta H. Benefit-based data caching in ad hoc networks. IEEE Transactions on Mobile Computing, 2008, 7(3): 289–304
13. Jin S D, Wang L M. Content and service replication strategies in multihop wireless mesh networks. In: Proceedings of ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems. 2005
14. Dabrowski C. Reliability in grid computing systems. Concurrency Practice and Experience, 2009, 21(8): 927–959
15. Bonvin N, Papaioannou T G, Aberer K. Dynamic cost-efficient replication in data clouds. In: Proceedings of the 1st Workshop on Automated Control for Datacenters and Clouds. 2009
16. Milani B A, Navimipour N J. A comprehensive review of the data replication techniques in the cloud environments: major trends and future directions. Journal of Network and Computer Applications, 2016, 64: 229–238
17. Bonvin N, Papaioannou T G, Aberer K. A self-organized, fault tolerant and scalable replication scheme for cloud storage. In: Proceedings of the 1st ACM Symposium on Cloud Computing. 2010, 205–216
18. Nguyen T, Cutway A, Shi W. Differentiated replication strategy in data centers. In: Proceedings of the IFIP International Conference on Network and Parallel Computing. 2010, 277–288
19. Ahmad N, Fauzi A C, Sidek R M, Zin N M, Beg A H. Lowest data replication storage of binary vote assignment data grid. In: Proceedings of the 2nd International Conference on Networked Digital Technologies. 2010, 466–473
20. Bin L, Jiong Y, Hua S, Mei N. A QoS-aware dynamic data replica deletion strategy for distributed storage systems under cloud computing environments. In: Proceedings of the 2nd International Conference on Cloud and Green Computing. 2012, 219–225
21. Anusha, Pureti, T. Sunitha, and Mastan Rao Kale. "Detecting and Analyzing Emotions using Text stream messages." *ECS Transactions* 107.1 (2022): 16913.
22. Aharonu, Mattakoyya, et al. "Entity linking based graph models for Wikipedia relationships." *Int. J. Eng. Trends Technol* 18.8 (2014): 380-385.