Global Journal of Information Technology: Emerging Technologies

Volume 13, Issue 1, (2023) 46-54

www.gjit.eu

Investigation of Efficient Backup Techniques to Reduce Late in Cloud Systems: A Modeling Study

Ezgi Pelin YILDIZ[a] * Kafkas University/Kazim Karabekir Vocational School of Technical Sciences Computer Programming Department, Kars, TURKEY.

Şahap ALTINBAŞ[b]*, Kafkas University Faculty of Economics and Administrative Sciences, Department of Management Information Systems, Kars, TURKEY.

Suggested Citation:

Yıldız, E., P. & Altınbas, S. (2023). Investigation of Efficient Backup Techniques to Reduce Late in Cloud Systems: A Modeling Study. *Global Journal of Information Technology: Emerging Technologies*. *13*(1), 46-54. https://doi.org/10.18844/gjit.v13i1.8863

Received from; October 11, 2022, revised from; January 23, 2023 and accepted from February 14.

Selection and peer review under responsibility of Assoc. Prof. Dr. Ezgi Pelin YILDIZ, Kafkas University, Turkey.

Abstract

From the formation stages of the information society to the present day, new developments have been experienced in many areas such as data formation, processing, storage and sharing of data in electronic environments. In this context, many new technological concepts have been included in our lives. When it comes to data storage environments, the concept of "Cloud Computing", which has been developing and increasing in popularity in recent years, comes to our minds. In this sense, it is possible to consider three areas of cloud infrastructure; cloud computing, distributed storage and streaming communications. One of the first solutions that comes to mind is duplicating a task on multiple machines and waiting for the earliest copy to finish can reduce service latency; but intuitively, it costs additional computing resources and increases the queue load on servers. In this context, the effect of redundancy on the tail will be analyzed in the first part of the research. In a similar way, by requesting more than one copy of a file and waiting for any of them to arrive, the cloud storage requests will be accelerated for content download. In the second part of the research, generalization will be made from replication to coding and the (n, k) fork-join model will be studied to analyze the delay in accessing a storage system with (n, k) deletion code. This analysis will provide practical information that several users can access a content at the same time and provide faster service to the relevant users. Achieving low latency in streaming communication is even more difficult, as packets must be transmitted quickly and sequentially. Based on this structure, in the third part of the research, it is aimed to develop deletion codes to transmit redundant packet combinations and ensure smooth playback. In general, the aim of the research is to blend various mathematical tools from queuing, theory coding and renewal processes. It is foreseen that the techniques and insights to be developed with these dimensions of the research can be applied to other systems with stochastically changing components, and integrated studies can be made and contribute to the literature in this context.

Keywords: Cloud Computing, Latency, Efficient Backup Techniques, Experimental Modeling.

*** ADDRESS FOR CORRESPONDENCE:** Ezgi Pelin YILDIZ * Kafkas University/Kazim Karabekir Vocational School of Technical Sciences Computer Programming Department, Kars, TURKEY. E-mail: yildizezgipelin@kafkas.edu.tr

I. INTRODUCTION

Cloud computing is a structure that allows the user to access, back up and transfer data over a remote server at any time, or to perform desired operations using applications running on a remote server in the internet environment (Marston, et al. 2011). Via the cloud computing, it is possible to access the desired data from anywhere and by using any electronic communication device, in line with the authority possessed (Uslu, et al. 2020).

Cloud computing service types consist of 4 types as private, public, hybrid (Hybrid) and community, according to their usage patterns. A public cloud is a type of service that provides general use to users over the internet by cloud computing service providers, or in other words, is open to the public (Shawish and Salama, 2014). Private cloud is a type of service where the security measures of users are higher than other cloud types in cases where the data security of companies is important. Companies generally prefer to purchase only in-house services by closing all access ways outside the institution (Garber, Malik & Fazio, 2013). Hybrid (Hybrid) cloud is a type of service where more than one cloud is used together. Hybrid cloud type is defined as a cloud cluster formed by shaping other cloud types according to needs (Tripathi & Jalil, 2013). A community cloud is a type of cloud that a particular organization or community uses together. The aim here is to bring together institutions in common working areas (Henkoglu, Kulcu, 2013).

Cloud computing is divided into three service models recommended by the National Institute of Standards and Technology (NIST). It is in the form of infrastructure service (Infrastructure as a Service: IaaS), platform service (Platform as a Service: PaaS) and software service (Software as a Service: SaaS) and allows to use one or more of these models. The cloud software service provides users with multiple web-based software and applications, allowing users to use them without installing software. This service model is a preferred model in terms of cost and time because it offers a fast, easy and ready platform to users (Adhikari M. & Amgoth, 2018). The cloud platform service provides all the necessary facilities for the software development environment as well as allowing its users to develop new programs and manage their software (Yasrab, 2020).

Cloud infrastructure service is a cloud service model expressed as a resource cloud among users. It is a model that provides services such as dynamic data storage, virtual presentation of processor resources to be used, and network services by providing secure access according to the type of network the user needs (Keskın, et al. 2019).

Today, the rapidly increasing applications are now hosted in the cloud and latency is an important quality indicator in these services. Users expect fast response from cloud services as seamless as using a personal computer to run the application. This requirement becomes more of a necessity as more interactive and collaborative applications emerge.

The biggest advantage of hosting applications in the cloud is that large-scale sharing of resources provides scalability and flexibility. However, the side effect of sharing flexible and coordinated resources is the latency variability experienced by users.

This is because other jobs with higher priority in the queue occupy the cache, server outages, etc. may be caused by various factors such as. The problem is exacerbated when users run a business with several parallel tasks in the cloud, resulting in the slowest task to deliver the service. Therefore, providing uninterrupted, low-latency service to the end user is a challenging challenge in cloud systems.

It has been used to reduce these delays with the use of redundancy, which is an important method in recent years. In cloud computing, running a task on multiple machines and waiting for the earliest copy to finish can significantly reduce latency (Dean & L. Barroso, 2013). However, redundancy in user requests can lead to increased use of resources such as computation time and network bandwidth. For example, on platforms like Amazon EC2 and Microsoft Azure that offer computing as a Service, server time spent is proportional to money spent renting machines.

Cloud services are changing the world by giving people low-cost access to the computing power of data centers. Storing and processing data on shared servers in the cloud gives these services scalability and flexibility. However, large-scale resource sharing also results in unexpected changes in response times for individual servers. In this research we use replication as a tool to combat this variability. We examine three areas of cloud infrastructure: Cloud Computing, Distributed Storage and Stream Communication.

In Cloud Computing, copying a task on multiple machines and waiting for the earliest copy to complete can help reduce latency service. But intuitively, additional computational resources are required and queue load on servers increases. In the first part of this research, we examine the effect of duplications on sequence loads. Surprisingly, copying not only reduces the latency service, but also reduces the sorting overhead, thus making the system more efficient. Similarly, it can speed up downloading of content from cloud storage systems by requesting multiple copies of a file and waiting for any of them. In the second part of the research, it generalize replication to encoding, and the (n, k) fork-join model will be used to examine access time latency in a storage system with (n, k) delete codes. This research provides useful information on the number of users who can access a material at the same time and how quickly it can be achieved. Low latency is much more difficult to achieve during streaming communication because packets must be sent quickly and in the correct order. The third part of this research generates erasure codes to send redundant packet combinations and guarantee lag-free playback.

This research; it combines various mathematical techniques such as refresh processes, queuing and coding theory. Although we concentrate on cloud infrastructure, all systems with stochastically variable components can benefit from methodologies and insights.

II. METHOD

To address variation in response times for individual servers and reduce average latency, it is a good way to distribute a task among several servers and wait until the first copy is complete. However, replication can result in higher account resource costs and queue latency due to higher traffic load. Therefore, answering basic design questions becomes an important question:

1. How many copies should be started?
2. Which servers should the copies be assigned to?
3. When should the same jobs be started and stopped?

This section presents a paradigm to answer these questions about redundant queues and explain how task replication can achieve cost-effective latency reduction. One key finding is that the 'daily arc' characteristic of the task service distribution plays a critical role in deciding whether replication is beneficial. In the case of adding maximum replication, service distribution log-convex, latency (wait time and service time) and cost are reduced. Also, log-concave is more effective for reduced repetition and cancellation of early overactivity.

In this direction, the system model to be created with the formulation of the problem:

Consider a distributed system with n statistically identical servers. Tasks arrive at the system at a rate of λ per second, taking into account the Poisson process. On one or more servers, all incoming tasks are assigned to first-come, first-served queues. The number of copies and the method of assigning and canceling them are the same for all tasks.

Once a task reaches the head of its queue, the time it takes to serve it can be random due to various factors such as disk seek time, network congestion, and the sharing of computing resources among multiple processes. We model this with the aggregate distribution function (CDF) FX(x) and the service time of X > 0, and we assume it is between requests and servers. The inter-server service time dependency can be modeled by adding a constant value to X service time. We use $F^{-}X(x) = Pr(X>x)$ to show the tail distribution of X (inverse CDF). and Xk:n notation, random variables X1, X2, . . . represents the kth smallest value of Xn.

Conclusion; recent publications have examined a number of system model generalizations, while others remain unexplored.

Parallel Computing: Straggler Multiplication

In the second part, we focused on task duplication in parallel computing and found a near-ideal replication approach assuming it is used by each task. Maximum parallelization is a technique used by computational frameworks such as MapReduce/Hadoop (Dean & Ghemawat, 2008) and Apache Spark (Ghemawat et al. 2005). This technique is used to divide a large task into many smaller tasks and perform them simultaneously on multiple machines. These frameworks can only be used to run optimization and machine learning algorithms that can easily split independent parallel tasks, such as mutual routing multipliers method (ADMM) (Boyd et al. 2011) and Markov Chain Monte Carlo (MCMC) (Neiswanger, Wang, Xing, 2013).

The delay in waiting for the slowest tasks or "strays" to finish is a major barrier to completing a job that consists of many simultaneous processes. The latency of performing many parallel tasks can be much larger (140 ms) than the average delay of a single task (1 ms). MapReduce and Apache Spark start a "backup" copy of scattered tasks to speed up work (Dean & Ghemawat, 2012). This is also called "speculative execution". A series of system studies (Ananthanarayanan et al. 2012) and references here further developed this idea. For example, Apache Spark implements "speculative execution" to allow slow-running tasks to be restarted (Apache Software Foundation Report, 2016).

Although task duplication has been studied in the systems literature and adopted in practice, there is not much work on the mathematical analysis of replication strategies. In this section, we develop a mathematical framework for analyzing such rambling replication strategies. In particular, the choice of a rambling replication strategy involves optimizing the following aspects: How many copies to start for each job that is still unfinished; the percentage of tasks identified as stray and whether the original copy will terminate.

By identifying how these affect the trade balance between latency and computational cost, it is possible to identify situations where duplicating a small number of tasks can significantly reduce latency while also saving computational cost. These insights allow the application of optimization to look for programming policies based on their sensitivity to computational latency and computational cost.

In this direction, the system model to be created with the formulation of the problem:

In this section, we define some of the notations used. Lowercase letters (for example, x) represent a particular value of the corresponding random variable, and capital letters (for example, X) represent the random variable. We denote the total distribution function of the random variable X as FX(x). Its complementary tail distribution is denoted as F¯_x (x)_=^Δ 1-F_x (x). High end point of FX:

It define ω F_x 〖(x) 〗 _(=)^Δ sup {x: FX(x) <1}. For random variables X1, X2, · · ·, Xn, it define the j-th smallest n random variable as the j-th ordinal statistic, Xj:n.

If we model the system; Let's consider a job with n parallel tasks, n large and each task is assigned to a different machine. We use the Fx probability distribution to model random variations in machine response time and consider the causes of these variations as factors such as density, sequencing, virtualization, and competing jobs running on the same machine. We also assume that this runtime distribution is independent and uniform across machines. Assuming the same distribution of FX, the tasks in this job are distributed to machines with processing power proportional to the task size, and the distribution of homogeneous tasks to homogeneous machines is the simplest example. The assumption of independent distribution of FX can be met when machine response times change independently of time, or when each new task (or copy) is assigned to a new machine that is not used for previous tasks. From the user's perspective, when a user rents a machine from a cloud computing service, they often have little or no control over other loads that share resources. We consider the uncertainty captured by FX as an exogenous element.

A scheduling policy or scheduler assigns one or more copies of each task to different machines, perhaps at different times. The scheduler receives instant feedback when a machine completes its assigned task. There is no intermediate feedback indicating the processing status of a task. When a notification is received that at least one copy of each of the n tasks has been completed, the scheduler immediately terminates all other copy runs. We focus on a set of policies that matter, called one-fork policies, and are defined as:

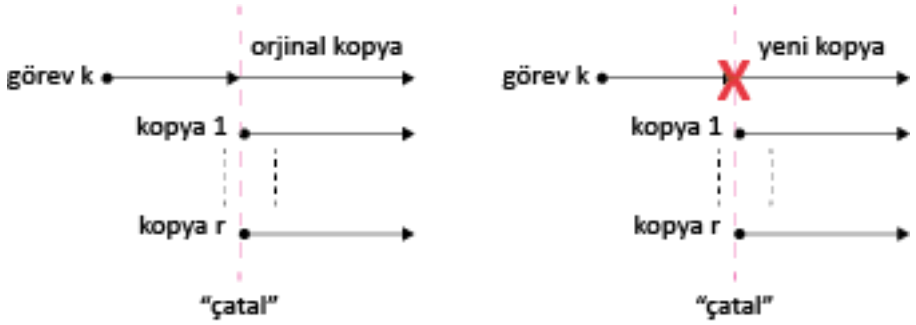
Description 1 (Single fork scheduling policy). Single fork scheduling principle π(p,r) starts all n tasks at time 0. (1 - p)n waits for its tasks to finish. For the remaining pn tasks, it chooses one of the following two actions:

- Make and keep backups of the original copy (πkeep (p, r)): r make a new backup;
- make backups of the original copy and kill the original copy (πkill (p, r)): kill the original copy and make r + 1 new backup.

When one of the earliest backups of a task runs out, the other remaining backups of the same task are terminated.

In both scenarios a total of r+1 backups work after forking. Figure 1-1 shows whether to keep the original copy of a task or to kill it. For simplicity of notation, it is assumed that p is an integer of pn. It indicates that p = 0 requires n tasks to run and finish in parallel, which is the base case where no backups or original tasks are killed.

Remark 1 (Replacement Tasks in MapReduce). The "backup" tasks in Google's MapReduce [11] and the "speculative run" idea in Apache Spark [19] are equivalent to r = 1 and πkeep with the one-fork principle. The p value is set dynamically and therefore is not specified in [11]. The spark.speculation.quantile configuration corresponds to the p in the single fork policy.



(a) Keep original copy (πkeep) (b) Özgün kopyayı öldür (πkill)

Figure 1-1: One Fork Principle Illustration

While we have focused on single fork principles in this section, it can also be generalized to multi-fork principles where backups of a new task are run multiple times during run execution. While forking multiple times provides better latency-cost trading, the increased costs of running and killing backups may exceed the benefit of this increase.

III. RESULT AND SUGGESTIONS

The cloud flexibly and scalably provides compute access from thousands of shared servers in data centers. Fast and fluent responsiveness is one of the key metrics of performance. As apps have become more interactive, we users have become responsive, even to sub-second delays. Ensuring such low latency is a difficult process as service times can vary due to factors such as virtualization, crashes, packet loss. And it's difficult to centrally monitor servers to accurately predict their status. In this research, we use backups to reduce latency statelessly and cost-effectively.

Broader Future Directions:

More generally, it is known that the amount of data in the cloud is increasing at an alarming rate. Therefore, the need to build infrastructure to store and process this data quickly, efficiently and reliably has become urgent. This research develops a new mathematical framework for understanding how backups can help with random service delays. Having this framework, we can approach problems with stochastic variability in a wide variety of applications. Some areas of particular interest are described below.

Large-Scale Machine Learning: Machine learning algorithms make up the bulk of computing performed in the cloud today. These algorithms require training large neural networks using millions of training examples. For speed and scalability, training can be parallelized by creating model backups (Dean et al. 2012). These distributed learning frameworks are affected by service variability and persistence.

Mass Equity: People are becoming part of the cloud infrastructure. Human-intelligent tasks can be assigned to employees through platforms such as Amazon Mechanical Turk. Although today mostly used for small tasks (eg, image tagging), crowdsourcing has the potential to change how large projects are performed by humans. Backup techniques based on this research can be applied to reduce the variability of employees waiting to try and complete tasks.

Noisy Logic Devices: The building blocks of all cloud infrastructure are logic gates and memory devices. Some newer devices, such as spintronic gates, have an interesting accuracy-speed trade-off; their accuracy gets better as we give them more time. We are interested in techniques for developing a circuit design that balances accuracy and speed.

Energy Systems: In electrical grids, matching supply and demand is important. This matching becomes even more difficult due to greater integration of stochastically varying renewable resources. At the same time, data from smart meters offers opportunities to direct demand to an increasing oversupply. The probability modeling and analysis techniques used in this research can be applied to build data-based scheduling and improve energy efficiency.

Consequently, in this research, we explored the use of backup to reduce latency in cloud systems. Replication, and more generally coding backup, has been used for several decades in areas such as communications, network routing. However, scheduling in cloud systems has hardly been explored. This research presents a theoretical framework for operating redundant queuing systems and reduces latency in a cost-effective way.

REFERENCES

Adhikari M. & Amgoth T. (2018). Heuristic-Based Load Balancing Algorithm for Iaas Cloud, *Future Generation Computer Systems, 81,* 156-165.

Apache Software Foundation (2016). "Apache spark configuration - scheduling (version 1.5.2)." https://spark.apache.org/docs/1.5.2/configuration.html.

C. Reiss, A. Tumanov, G. Ganger, R. H. Katz & M. A. Kozuch (2012). Towards understanding heterogeneous clouds at scale: Google trace analysis, Intel Science and Technology Center for Cloud Computing, Tech. Rep.

Garber D., Malik, J., Fazio A. (2013). *Windows Azure Hybrid Cloud*, John Wiley & Sons, Indianapolis.

Godse M. & Mulik S. (2009). An Approach For Selecting Software-As-Aservice (Saas) Product, In 2009 *IEEE International Conference on Cloud Computing,* 155- 158.

Henkoğlu T. & Külcü Ö. (2013). Cloud Computing as an Information Access Platform: A Study on Risks and Legal Conditions, *Information World, 14(*1).

J. Dean. & L. Barroso (2013). *The Tail at Scale, Communications of the ACM*, 56 (2), pp. 74–80.

J. Dean. & S. Ghemawat, "MapReduce: simplified data processing on large clusters," ACM Commun. Mag., vol. 51, pp. 107–113, Jan. 2008; M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault tolerant abstraction for in-memory cluster computing," in Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, pp. 15–28.

J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Aurelio Ranzato, A. Senior, P. Tucker, K. Yang, Q. V. Le & A. Y. Ng (2012). Large scale distributed deep networks, *in Advances in Neural Information Processing Systems* 25, pp. 1223–1231.

Keskin N., Kıran, A. N., Eğdemir, F. K. & Eren T. (2020). Service Provider Selection with Multi-Criteria Decision Making Methods According to Cloud Computing Security Requirements, *International Journal of Information Security Engineering,* 6(1), 45-60.

Marston S., Li Z., Bandyopadhyay S., Zhang, J. & Ghalsasi, A. (2011). Cloud Computing: the Business Perspective, *Decision Support Systems. 51*(1), 176-18.

M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker & I. Stoica (2010). Spark: Cluster computing with working sets, in Proceedings of the 2nd USENIX conference on Hot topics in cloud computing.

Shawish A. & Salama M. (2014). Cloud Computing: Paradigms and Technologies*, In Intercooperative Collective Intelligence: Techniques and Applications, 495*, 39-67.

S. Boyd, N. Parikh, E. Chu, B. Peleato & J. Eckstein, (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers, *Foundations and Trends in Machine Learning, 3* (1) 1–122.

Uslu B., Gür Ş., Eren T. & Özcan E.C. (2019). Cloud Service Provider Ranking with Multi-Criteria Decision Making Methods, Pamukkale *Journal of Business and Information Management, 6* (1), 20-34.

Uslu B., Eren, T., Gür Ş., Özcan E. (2019). Evaluation of The Difficulties in The Internet of Things (IoT) with *Multi-Criteria Decision-Making, Processes, 7*(3), 164.

Tripathi A. & Jalil M.S. (2013). Data Access and Integrity with Authentication in Hybrid Cloud, *Oriental International Journal of Innovative Engineering Research, 1* (1), 30.

W. Neiswanger, C. Wang, and E. Xing (2013). Asymptotically exact, embarrassingly parallel MCMC, arXiv:1311.4780.

Yasrab R. (2020). Platform-As-A-Service (Paas): The Next Hype of Cloud Computing, Arxiv Preprint Arxiv: 1804, 10811, 2018; Sanaj M.S., Prathap P.J., Nature Inspired Chaotic Squirrel Search Algorithm (CSSA) for Multi Objective Task Scheduling in An Iaas Cloud Computing Atmosphere, Engineering Science and Technology, An International Journal, 23(4), 891- 902.