International Journal of Engineering Science Invention (IJESI)
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org ||Volume 7 Issue 5 Ver.I II || May 2018 || PP 72-77

A highly fault tolerant Multimedia Cloud Streaming approach for private cloud

Divya G[1], Preetha Evangeline D[2], Anandhakumar P[3]
[1](Assistant Professor, SRM Institute of Technology, Kattankulathur, Chennai, India
[2](Assistant Professor (Sr.Gd), Department of CSE, Vellore Institute of Technology, Vellore, India
[3](Professor, Department of CT, Madras Institute of Technology, Anna University, Chennai, India
Corresponding Auther: Divya G

Abstract:*Traditional multimedia services include the various services such as Streaming, Video conferencing, content sharing, real-time monitoring and broadcasting. If the multimedia services rely on the client systems for their processing the results are very drastic as these services computationally require high processing capabilities. On the other hand, the development of cloud based environment was centered to be highly distributed and to share the workload between the number of servers in the cloud. Hence, there is a great hope for multimedia services to efficiently use the cloud resources and deliver content to the specified user in the manner which he expects. This gave rise to a new area of much interest - Multimedia Cloud, which entirely deals with how the user requests are processed and the needed resources are allocated. Though this may seem easy, there are difficulties additionally in the multimedia cloud, which do not occur in traditional cloud environments. One critical issue is the reliability due to the downgrade of the cloud services. We propose a novel fault tolerance approach to detect reliability vulnerabilities and take necessary actions accordingly. The results of the fault tolerance approaches have been analysed.*

Keywords – *Fault Tolerance, High Performance Computing, Multimedia Cloud, Reliability, Streaming*

Date of Submission: 07-05-2018 Date of acceptance: 22-05-2018

I. Introduction

Multimedia Cloud Streaming is one of the latest developments in the area of cloud computing. It overcomes the limited resources problem found in the classical streaming environments. The reason behind this is cloud computing environments are elastic in nature. Hence they can balance off the load according to the fluctuations in the network. It is also easy to re-commission the required services in case of failures. But downgrade of services is possible. The reasons for the downgrade of services are many. Hardware failure is one of the chief causes. Queue overflow during re-commissioning is another. Whatever the cause, the effect of downgrade is drastic on the customer. Hence for the satisfaction of the customer that there will not be any serious service downgrades, a service level agreement is provided to the customer with parameters such as availability of the services. But there is not much guarantee on the QoS of the services provided. In case of drastic loads, the services are delivered to the user at worst QoS levels. Though the QoS level is not satisfactory to the users, the services are still up and running. A step higher fault tolerance of a system is the property by which the system can recover from a failure. In this paper, we mathematically model the metrics to provide fault tolerant streaming services. The fault tolerance metrics are computed using the variation in a few parameters rather than a straight-forward approach in which it is necessary to compute numerous parameters. And we use our own above defined metrics to evaluate the reliability of a multimedia cloud environment and make it fault tolerable [1]. The organization of the paper are as follows, Section 2, lists out recent work on reliability and fault tolerance approaches and brought out the features and demerits of those approaches. Section 3, elaborates on the metrics of the proposed work have been defined. Section 4 brings out the details of the implementation and the results that are obtained. For better perception, the results of the analysis are depicted in the form of graphs. In Section 5 concludes the contribution and its future work.

II. RELATED WORK

As already mentioned, multimedia cloud services require computational resources to a great extent. [2] deals with the adaptive mobile computing environment for providing multimedia services. The paper deals with high-quality graphically intensive gaming services played from a mobile device. Based on the user's actions on the device, the user interface is rendered accordingly. This is usually of a three dimensional rendering and hence this is

performed on the server side. The results of the computations needed for rendering is sent to the user in

a compressed format. The user device has only the overhead to decompress the data and make use of it. [3] proposes a queue based methodology to optimally provision the QoE and allocate the required resources. It deals with optimization problems such as minimizing the response time(The time lapse between the submission of requests and the start of reply from the server). Also the method includes minimizing the amount of resources required for the QoE provisioning. [4] deals with the design of a novel stream dispatcher. The responsibility of this dispatcher is to analyse the incoming requests and identify fluctuations. The content to be delivered is segments and based upon the current characteristics of the environment, the dispatcher module makes an optimal decision to send the most appropriate segment depending on the scenario. [5] proposes a multimedia content recommendation framework. At first, the various services available and the various social network sub-groups were listed out and analysis was performed on them. Based on the results of the analysis, the content can be recommended to a group of users dynamically. [6] discusses about the quantification of availability in general sense. The authors define availability in terms of the service time and downtime.

(1)

And as a step higher, the authors also define reliability as follows:

(2)

As a variant the authors in [7], have defined reliability as a probability of an entity to perform its desired function. Reliability metric governs that the services are provided within the expected time and also the services are accurate as expected. If either one of these two is compromised, the system is considered to be not reliable. The authors in [8] have identified the vulnerable spots where reliability issues and failures occur within a cloud environment. A cloud environment consists of bare metal servers, and the hypervisors that run on them and other components. In such an environment failure mostly occurs at the storage systems which include RAID controllers. Also in cloud environments, we avoid the single point-of-failure by setting application nodes in virtual machines and the number of virtual machines can be decided by the load balancer. As discussed before, the fault tolerance metrics are computed using the variation in a few parameters rather than a straight-forward approach in which it is necessary to compute numerous parameters. And we use our own above defined metrics to evaluate the reliability of a multimedia cloud environment and make it fault tolerable.

In [9], a time stamp based event tracker was developed using .NET programming. This is based upon building a directed acyclic graph. The vertices of the graph are the nodes and these vertices are constantly monitored. Whenever a vertex presents itself to have failed, a new node is started up and the failed one is replaced by the newer one. The streaming resumes as normal. Timestamp is used in the process of ordering. One major disadvantage in this approach is that the re-commissioning must be done before a new node is set up. If this is not done, the system comes to a complete stop.

In [10], storm was deployed in a cloud environment and the authors tried to achieve fault tolerance and performance. The approach sacrificed availability for attaining performance. The resources were assumed to be distributed and the client had a SLA provided by the cloud provider. Finally they summed up that cloud increases availability but we have to implement more stringent fault tolerance mechanisms to handle failures in a distributed manner. Also it is found out that most cloud providers provide their clients with SLA. A few providers provide a range for the reliability measure and accordingly provides services. A few providers achieve this by preventing failures. Whatever the approach used, it entirely depends on the underlying hardware.

III. Proposed Fault Tolerant Approach

The aim is to come up with a fault tolerance mechanism based on custom designed metrics. Our approach adopts fault tolerance with less latency. We represent the relationship between the various components of the multimedia cloud streaming service interconnected with logic gates. The streaming services is thus represented in a network of components with AND and OR gates bridging them together. This has been derived from the fact that basically an AND gate return true when all of its inputs are said to be true. On the other hand, OR gate returns false only when all of its inputs are said to be false. A failure of the entire system occurs when any of the components connected by AND give out failures. And in case of those connected by OR gates, failure occurs when all of the components fail. This is more critical than the previous case. We process this circuit diagram into an intermediate form convenient for representation and processing. Reliability issues occur due to one of the following reasons. There may be a lost connection or the inability of the master server to establish a connection when needed. The

stream connections or requests may be sometimes rejected and hence leading to failure. Timeouts, errors and exceptions are also causes for the degradation of the reliability of a system.

The reliability is denoted as $\gamma(v,t)$ which is dependent on time t and streaming video v. The current value of γ is not only taken into consideration but also the previous measures. Let the start time be T and end time be S.

$$S = T + t \qquad (3)$$

Let ρ(t,s) denote the reliability measure between t and s. We take average of all such intervals and we much achieve γ(v,t) more or less equal to this average value, such that the system is stable. Now we compute the variance as follows:

$$V(\gamma) = \frac{\sum_{i=1}^{n}(\gamma(v,i) - \gamma_{avg})^2}{n} \qquad (4)$$

Services can be classified into normal services, downgrade allowable, broken and it is observed that the value of

V(γ) drops to zero in the broken stage. It varies continually in the downgrade allowable stage and is expected to be stable in the normal stage. Now next concern is the method by which the computation is carried out where the values form the logic gates map.

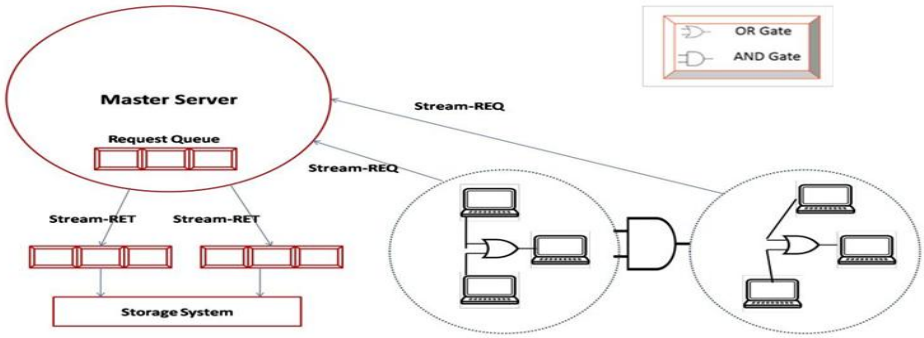

Figure. 1. Overview of the system

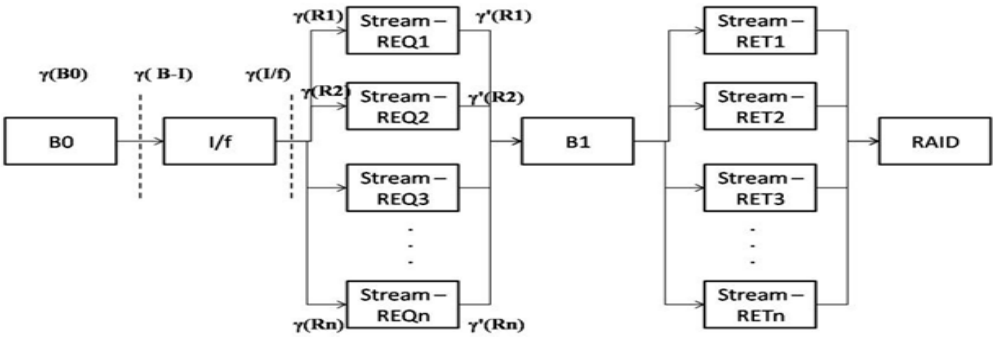

Figure. 2. Intermediate Diagram

The request processing takes care of two functions. One at the Master server to receive various requests and process them and based on their request to access the storage system. And also helps in retrieving the video content from the Storage Server and forwarding to the devices. The fault tolerance module, in case of a failure occurring in a particular desktop, the fault tolerance fault tolerance is achieved by Logic gate circuit based approach. Stream-REQ and Stream-RET are two important messages which request Streaming content from server and retrieve the streaming content from the storage system respectively.

Fig. 2 shows the intermediate representation considered for further processing. The block B0 is the load balancer. It is interconnected with the various request blocks (Stream-REQ) inside the Master Server. Following it, is the block B1 which represents the storage system communicator. It is activated during the retrieval of the content to be streamed. These are represented by Stream-RET blocks. The storage system consists of multiple disks on RAID.

We calculate γ(t,t+T) as follows: (5)

The computation takes into account both the software components as well as the hardware aspects of the streaming system. Moreover, we bother only on how to reduce the value of V[γ(t,t+T)]. As discussed before,

reliability issues occur due to one of the following reasons. There may be a lost connection or the inability of the master server to establish a connection when needed. The stream connections or requests may be sometimes rejected and hence leading to failure. Timeouts, errors and exceptions are also causes for the degradation of the reliability of a system. We now optimize our model to fit these scenarios. The streaming occurs for a time interval between t and t+T where T represents the time lapse between the start of the streaming and the end of the streaming. We define a variable ε which has a value 1 if there is no error in the system and has a value 0 if the system encounters an error. Error includes even the logical errors which produce unexpected results. An error and an logical errors differ in the fact that errors have a standard code defined in the system whereas logical errors are handled by specific mechanisms. Moreover the logic gate circuit should have a lesser recovery time. This is guaranteed by reducing the value of V[γ(t,t+T)].

One of the threats to guaranteed reliability is due to a hike in the rate of requests sent to the system. If the number of requests exceeds the threshold value, more requests are rejected. But due to the high intensity of the requests, there may be a downgrade of the system. The following graph in Fig. 3 shows the sample of the utilization of the master server for the requests received in the master queue.

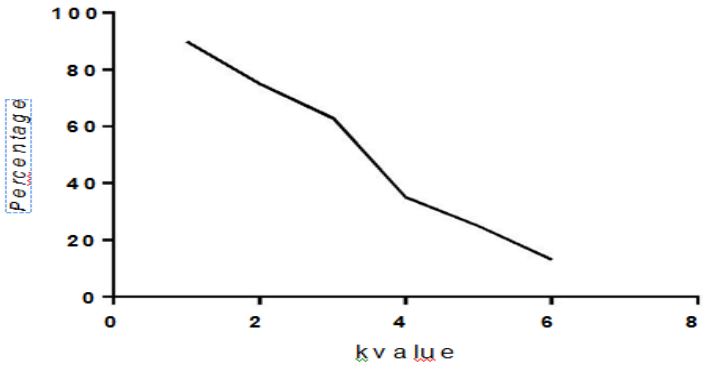

Figure. 3. Utilization of Master server

Now we elaborate the queue at the master server. The queue is assumed to have input following the Poisson distribution. We consider the number of requests to be infinite but the number of service counters to be k. On account of a virtual server failing, a new server is spawned to replace it. Let q be the time to spawn a new server. If the event occurred at t, the new server would be up and running at t+q. If one server had failed, the number of service counters drops by one, i.e, k-1. As servers continue to fail, the number of service counters goes on as k-

1,k-2,….1. But if k is reduced to k-1, (1/k) of the entire streaming requests would be timed-out. Moreover, longer the processing time, longer the request has to be waiting in the queuing system. The graph in figure 3 clearly shows the change in the server utilization with variation in the k values.

The following graph shows the various parameters such as service rates, waiting time, server utilization of various server counters.

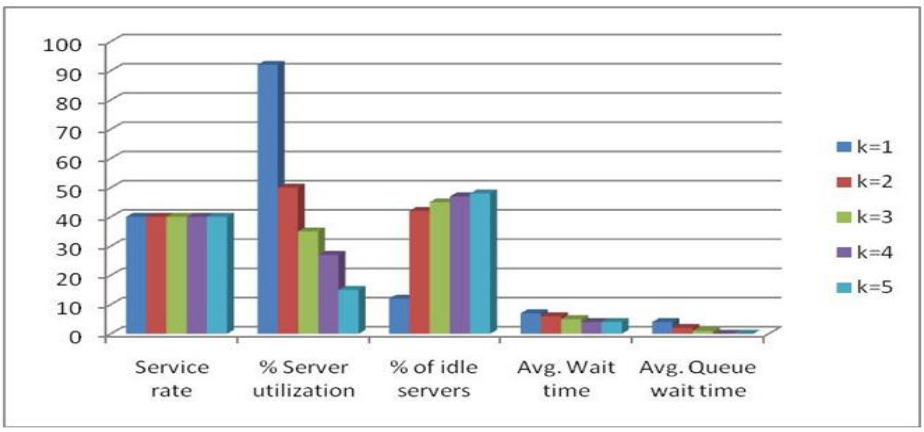

Figure. 4. Variation of Queue Parameters with change in k-value

The results are compared against different k-values as before. It is interesting to note that the service

rate is constant whatever the value of k is. But as k-value increases, the server utilization decreases. This is evident of the fact that the number of servers increases and hence the load is balanced among them and hence lower utilization. And conversely, the number of idle servers increases with increase in the k-value. The average waiting time in the system and the average waiting time in the queue also decreases with increase in k-value.

On processing of the request in the queue, the video is retrieved from the storage system. One striking feature is that the storage system is put on a physical storage rather than a virtual entity to get high responsiveness and throughput. A sequence field can be used to track the various requests in the system.

IV. Performance Analysis

The proposed a framework and underlying metrics to evaluate the reliability of a streaming system. A cloud environment would provide a distributed and elastic properties to streaming environments. We have surveyed numerous papers related to reliability and have analysed out and designed a new metric for reliability. The method of attaining reliability was trying to maintain the variance measure without any change as much as possible. This measure was obtained from logic gate circuit diagram and converting it into an intermediate form for processing. While analysis proved that when the storage system is shared, contention may give rise to performance downgrade. Also a longer threshold time value would lead to congestion in the cloud environment. So instead of this timeout threshold value, we could use some means to monitor out the various activities to avoid the need of the threshold.

Another way to ensure reliability doesn't downgrade is that we introduce separate channel queues after the request has been received. Thus the streaming requests are processed in a distributed manner among the other channels. Even if one fails, the others are not affected. Also during analysis, we observed that the queue could also make into account the possible failures in the future. We also analysed the SNMP protocol data of all existing systems in the context of cloud manager and the load balancer. According to the test, when a server crosses utilizing the seventy-fifth TCP port, the user could demand another new server to be spawned from the cloud.

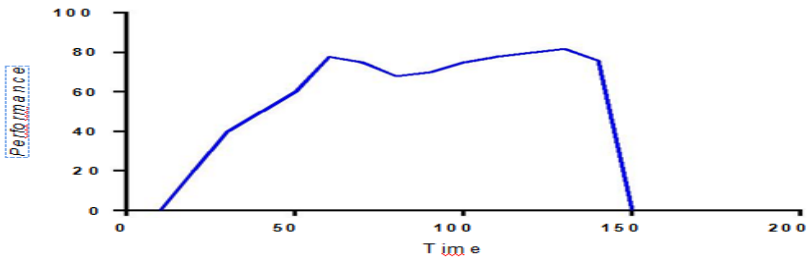

Figure 5. Performance of existing system

The above graph in Fig. 5 shows the performance variation as time progresses. Note at around time t=50, there is a dip in the graph before it rises again. At around this time there was an intended failure. The system performance downgraded due to the occurrence of this failure.

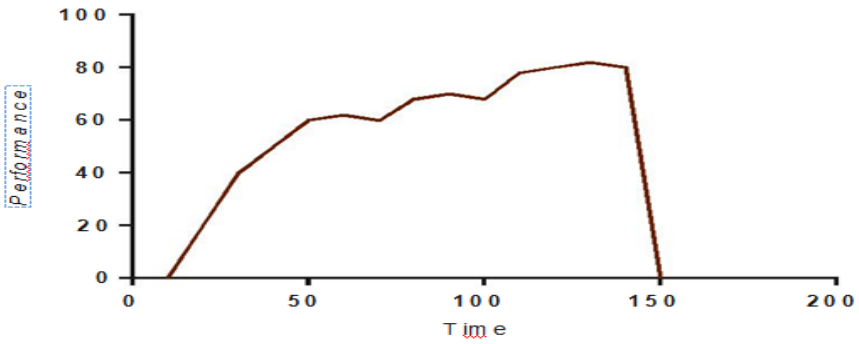

Figure. 6. Performance of the proposed system

The above graph shows the same scenario where one of the server undergoes a failure at time t=50. But the proposed system does not downgrade in its performance and copes up even though a server has failed. The graph in Fig. 7 shows the variation of the response of the system under various Stream-REQ scenarios.

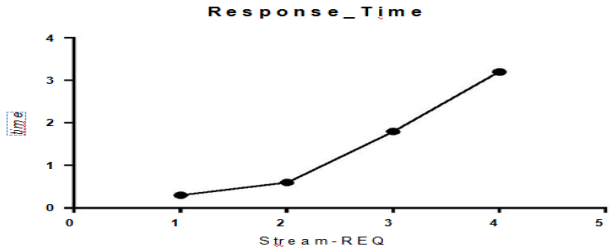

Figure. 7. Response Time of the system

V. Conclusion And Future Work

The proposed Framework is an application-specific solution. The work concentrates entirely on the reliability of multimedia streaming application. The entire experiment was analysed in the private cloud setup of MaaS Laboratory, Dept of Computer Technology, MIT, Anna University. Streaming systems have intense bandwidth and resource requirements. Although the cloud environment provides us with a enhanced way to support this intensive need of multimedia streaming using its elastic features, still there are other issues to be addressed. A novel metric for reliability have been proposed. The proposed approach has a superior performance and intended to focus on other issues, especially the contention problem that occurs on a shared storage system, which is the cause for degradation of performance due to system downgrade. And the cloud computing environment can also be altered from private to public or even hybrid clouds.

References

[1]. Wang Shaoxuan, Dey Sujit, Adaptive mobile cloud computing to enable rich mobile multimedia applications, IEEE Transactions on Multimedia 15(4), 2013 ,870- 883.
[2]. Goudarzi Pejman, Ranjbar Mohammad R Nezami, "Bandwidth allocation for video transmission with differentiated quality of experience over wireless networks", Computer and Electronics Engineering Journal, 35, 2011, 75-90.
[3]. Fajardo Jose Oscar, Taboada Ianire, Liberal Fidel, "QoE -driven and network-aware adaptation capabilities in mobile multimedia applications", Mutimedia Tools Applications Journal, 17(1), 2014, 104-117.
[4]. Wei J, Juarez E, Garrido MJ, Pescador F, "Maximizing the user experience with energy-based fair sharing in battery limited mobile systems", IEEE Transactions on Consumer Electronics, 59(3), 2013,690-698.
[5]. M.A. Rahman, H.N. Kim, A.E. Saddik, W. Gueaieb, "A context-aware multimedia framework toward personal social network services", Multimedia Tools and Applications, 71(3), 2014, 1717-1747.
[6]. E. Bauer, R. Adams, "Reliability and Availability of Cloud Computing" (John Wiley & Sons, 2012).
[7]. R. Denning, Applied R&M Manual for Defense Systems, 2012. [Online] Available: http://www.sars.org.uk/old-site-archive/BOK/Applied%20R&M%20Manual%20for%20Defence%20Systems%20(GR-77)/p0c00.pdf
[8]. K.V. Vishwanath, N. Nagappan, "Characterizing cloud computing hardware reliability", Proc: of first ACM Cloud Computing Symposium", 2010, 193-204.
[9]. Z. Qian, Y. He, Z. Wu, H. Zhu, Y. Zhang, N. Wang, L. Zhou, "Timestream: Reliable Stream Computation in Cloud", Proc: Eighth ACM European Conference on Computer systems, 2013, 1-14.
[10]. P. Bellavista, A. Reale, A. Corradi, S. Koutalas, "Adaptive fault-tolerance for dynamic resource provisioning in distributed stream processing systems", Proc: International conference on Extended Database Technology, 2014, 85-96.

Divya G "A highly fault tolerant Multimedia Cloud Streaming approach for private cloud "International Journal of Engineering Science Invention (IJESI), vol. 07, no. 05, 2018, pp 72-77