Article can be accessed online at http://www.publishingindia.com

Secure, Reliable and High Performance Multicloud Storage

Santhosh R*, Pradeep J**

Abstract

Many organizations are moving their IT infrastructure to cloud and many are getting ready to move their IT infrastructure to cloud. Leading players in cloud market Amazon and Microsoft are reporting huge increase in the sale of their cloud products. Still many organization have questions on the security and being dependent on the vendor to deliver their service. To overcome some of the concerns of the single cloud storage Multicloud storage was introduced. In this paper we have proposed the Secure, Reliable and High performance cloud storage above multicloud environment..

Keywords: Multi-Cloud, Vendor, Security

Introduction

In cloud storage we have a concerns like security, vendor lock in. In our reference papers [1] [2] Authors have proposed a 2 methods based on multi cloud storage to overcome these concerns by using RAID and Erasure code based storage, so we would be in a position to reconstruct the data using Erasure code even if one of the provider is down.

In [1] authors have proposed the method which would divides the available data objects into chunks in the multiple of available providers and use separate thread to store the allocated chunks to the respective provider. As mentioned by authors, this method has provided better security and provided a method to overcome the vendor lock in problem but it has performance concerns as the different providers would have different upload and download time. The upload or download process would be considered success only if the operations on each provider is completed irrespective of their operating speed.

In [2] authors have proposed a method to overcome the performance concerns mentioned in [1] by measuring the operating speed of the provider and allocate the particular number of chunks to the providers such that the operations on each provider completes at the same time. This method ensures better performance. This method has the disadvantage that chunks are not distributed uniformly so there is a good chance that more number of chunks getting stored in the fast provider and less number of chunks in slow providers. This would make the reconstruction difficult if one of the fast provider is down. It also requires predetermination of the operating speed of the provider which is difficult to predetermine because of its dynamic nature.

We need a method which has following characteristics, 1. Equal distribution of chunks 2. Dynamic adoption to provider service speed 3. Good performance. Proposed method satisfies all these requirements.

Proposed Method

Architecture

Basic block of this method is shown in figure [1]. This contains 3 main blocks. 1. Thread Manager 2. Queue 3. Provider specific file uploader. Here file which needs to be uploaded to the cloud is divided into packets and placed in the queue after encryption.

* Department of Computer Science and Engineering, Faculty of Engineering, Karpagam University, Coimbatore, Tamil Nadu, India. Email: santhoshrd@gmail.com

** PG Scholar, Department of Computer Science and Engineering, Faculty of Engineering, Karpagam University, Coimbatore, Tamil Nadu, India. Email: pradeep7437@gmail.com

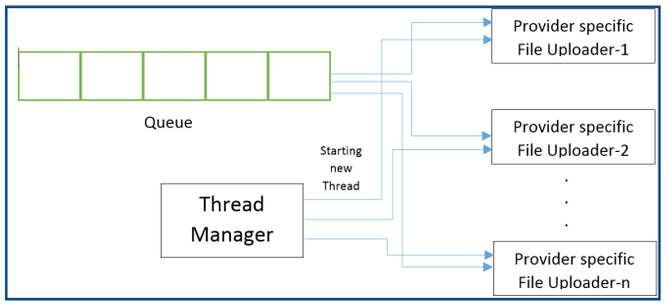

Fig. 1: Basic Blocks of the Proposed Method

Thread manager handles the threads specific to the providers. Initially it starts with n thread, where n is total number of providers available. So one thread per provider. Based on the performance of the individual provider, thread manager decides whether to start a new thread for a particular provider or continue with the single thread. Provider specific file uploader is a component which is develop using API specific to each provider. In this case we have considered 3 providers Microsoft Azure, Google cloud and Amazon Web services.

Queue

Queue holds all the packets which needs to be upload to the provider. File to be uploaded is divided into multiples of n, where n is the number or providers available to store the data. Minimum size of the packet is configured in the system and the system divides the packets based on this minimum size condition.

Thread Manager

Algorithm for Thread Manager is shown in Figure [2]. Input to this algorithm is the collection of objects representing all the cloud service providers available to store the data. Algorithm compares the number of packets stored for each provider with number of packets stored for the fastest provider. If the packets stored in fastest provider is twice more than the provider under consideration then new thread will be started for the provider under consideration and the process continues. In this way we will have multiple threads uploading the packets for slower providers. Algorithm for the parameter Fastest_Provider which is used in Thread Manager algorithm is shown in figure [3].

```
Input: Cloud_Providers, Fastest_Provider, adjustment_parameter
begin
        For each Provider in Cloud_Providers do
                if(Provider. Packets_Stored*2<= Fastest_Provider.Packets_stored)
                then
                        Start_New_Thread(Provider);
                end if
        end for
end
```

Fig. 2: Algorithm for Thread Manager

```
Input: Cloud_Providers
Output: Fastest_Provider
begin
        for each Provider in Cloud_Providers do
                if (Fastest_Provider==Null) then
                        Fastest_Provider=Provider;
                else if (Provider.Packets_Stored>Fastest_Provider.Packets_Stored)
                        Fastest_Provider=Provider;
                end if
return Fastest_Provider;
end
```

Fig. 3: Algorithm to Find the Fastest_Provider

Provider Specific file Uploader

These are the components developed using the provider specific API. Algorithm for file uploader is shown in figure [4]. It ensures that the number of packets distributed for the providers are equal. It starts by checking whether the queue is empty and number of packets stored in the particular provider doesn't violate uniform distribution condition. If the conditions are true then it call the provider specific methods to upload the file to cloud.

```
Input: Share_Per_Provider, Queue, Packets_Uploaded
begin
declare boolean Limit_Reached=false;
while (Limit_Reached) do
        if(Packets_Uploaded<Shares_Per_Provider AND Queue!= Empty)
        {
                Upload(Pop(Queue));
                Packets_Uploaded++;
        }
        else
        {
        Limit_Reached=true;
        }
end while
end
```

Fig. 4: Algorithm for Provider Specific File Uploader

Experiment and Result

We tested this method in a computer which has 4GB RAM, 320GB Hard disk, Windows 8.1 Enterprise (64 bits) OS and 12Mbps internet connection. Testing location was Coimbatore, India. We considered 3 cloud service providers for this testing 1. Microsoft Azure, 2. Google cloud 3. Amazon web service. For testing we took a file of 90MB size and it was divided into 9 packets.

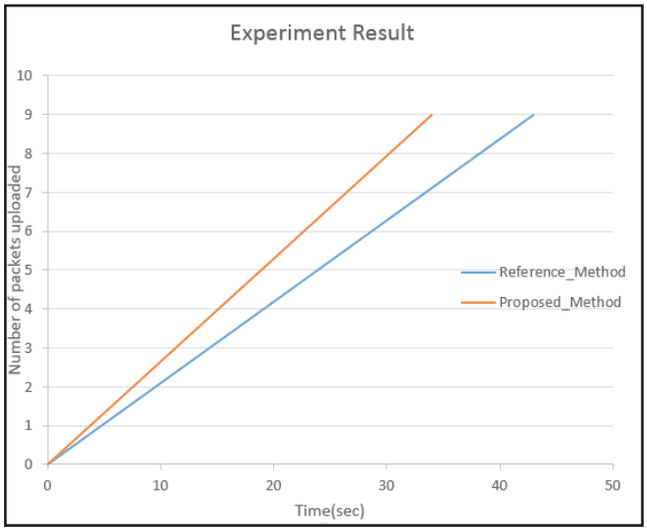

Fig. 5: Experiment Eesults-Comparison of Reference Method and Proposed Method

So for uniform distribution 3 packets needs to upload to each cloud. We run the test several times using existing method and proposed method. Average of the results are shown in the figure [5].

Conclusion

Results showed that the proposed method has provided better performance when compared to the method proposed in [1] and [2]. Results also showed that this method provides better performance when we have large file to upload to cloud. This method may not provide significant performance improvement for smaller files.

References

Schnjakin, M., & Meinel, C. (2013). Scrutinizing the state of cloud storage with cloud-RAID: A Secure and reli able storage above the clouds *IEEE 6th International Conference on Cloud Computing.*

Schnjakin, M., & Meinel, C. (2013). Evaluation of cloud-RAID: A secure and reliable storage above the clouds. *22nd International Conference on Computer Communication and Networks*.

Li, J., Li, Y. K., Chen, X., Lee, P., & Lou, W. (2015). A hybrid cloud approach for secure authorized deduplication. IEEE Transactions on Parallel and Distributed Systems, May, 26(5), 1206-1216.

Hongbing, C., Chunming, R., Kai, H., Weihong, W., & Yanyan, L. (2015). Secure IG data storage and sharing scheme for cloud tenants. Communications, China, June, 12(6), 106-115.

Schnjakin, M., Goderbauer, M., Krueger, M., & Meinel, C. (2013). Cloud storage and IT-security. *Proceedings of the 13th Deutscher IT-Sicherheitskongress.*

Hussain, S. K., & Sreenivasulu. (2012). An efficient and economical multi-cloud storage in cloud computing. *3rd International Conference on Sustainable Energy and Intelligent Systems (SEISCON 2012).*

Papaioannou, T. G., Bonvin, N., & Aberer, K. (2012). Scalia: An adaptive scheme for efficient multi-cloud storage. *SC '12 Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*.

Alqahtani, H. S., & Sant, P. (2016). A multi-cloud approach for secure data storage on smart device. *6th International Conference on Digital Information and Communication Technology and its Applications (DICTAP)*.

Fan, Y., Qiao, Z., & Xiao, M. (2014). Design and evaluation for a multi-cloud based storage system with privacy preserving. *9th IEEE International Conference on Networking, Architecture, and Storage*.

Wu, C. H., & Wang, P. H. (2014). Secure multi-key file-sharing for cloud storage with erasure coding. *International Conference on Computer, Information and Telecommunication Systems (CITS)*.

Libardi, R. M. de O., Reiff-Marganiec, S., Nunes, L. H., Adami, L. J., Ferreira, C. H. G., & Estrella, J. C. (2015). MSSF: User-Friendly Multi-cloud Data Dispersal. *IEEE 8th International Conference on Cloud Computing Year*.

Libardi, R. M. de O., Bedo, M. V. N.. Reiff-Marganiec, S., Estrella, J. C. (2014). MSSF: A Step towards User-Friendly Multi-cloud Data Dispersal. *IEEE 7th International Conference on Cloud Computing.*

Schnjakin, M., & Meinel, C. (2013). Implementation of cloud-raid: A secure and reliable storage above the clouds. *Proceedings of 8th International Conference on Grid and Pervasive Computing - GPC 2013*.

Wang, H. (2015). Identity-based distributed provable data possession in multi cloud storage. *IEEE Transactions on Services Computing*, March, 8(2), 328-340.