# SafeBoX: Overcoming Data loss and Improved Multiclient Consistency via Cyclic Redundancy Check in Cloud Data Synchronization

Muhammad Majid Sharif, Sajid Umair, Mian Muhammad Hamayun
School of Electrical Engineering and Computer Science (SEECS)
National University of Science and Technology (NUST) Islamabad Pakistan
{14mscsmsharif, 14mscssumair, mian.hamayun}@seecs.edu.pk

***Abstract***— ***Cloud computing is one of the most popular achievements in the field of computers in recent years, where back-end service providers provide various services to their clients. Cloud storage is one of the services where clients can store their data on service provider's storage through internet. To avoid the propagation of corrupted data is one of the main problems faced by service providers. If we can detect corruption correctly then we can use some methods to counter these problems. In this paper, we propose the use of Cyclic Redundancy Check (CRC) to achieve the goal of detecting corruption. CRC will be combined with EXT-4 file system to make it compatible to as many systems as possible. The proposed EXT4-CRC method will help us detect corruption better with lesser data overhead and we can use a restoration mechanism based on in-memory views to recover from data losses.***

***Keywords— Cloud storage; CRC; Dropbox; Seafile; ViewBox***

## I. INTRODUCTION

Cloud computing deals with the computation and storage of data in response to requests from remote clients. In the past people used to execute programs from a physical computer and also used same programs on server machines in the offices. For using the same data, either they have to take that data with them physically in some hard/flash drive or had to transfer it using remote copy operations i.e. scp command in Linux. Cloud computing provides the facility to the users to store and access their data with the help of internet from anywhere[1]. Cloud computing provides a lot of benefits to the users like flexibility, automatic software update, increased collaboration, work from anywhere, document control, security, competitiveness and elasticity [2]. A typical cloud computing model is presented in Fig. 1.

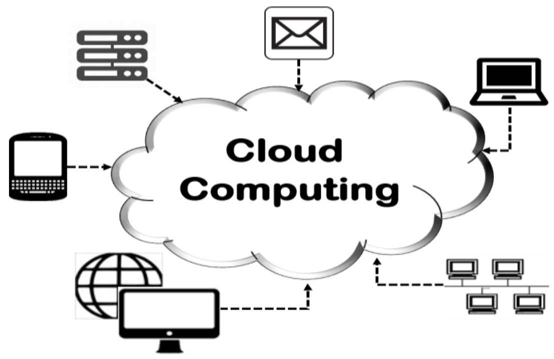

**Fig. 1** Cloud Computing Model.

Cloud computing provides different dynamic and scalable services to its users like Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [3, 4]. In this paper, we will discuss our work on cloud storage. Cloud storage means that we store data in the cloud based virtual machines. A cloud storage facility allows companies and clients to store their data and access it from different locations simultaneously. Cloud storage provides many benefits like reliability, accessibility, synchronization etc. [1, 5]. Some other benefits are rapid deployment, data backup and lower storage costs [5]. The cloud storage model is shown in Fig. 2.

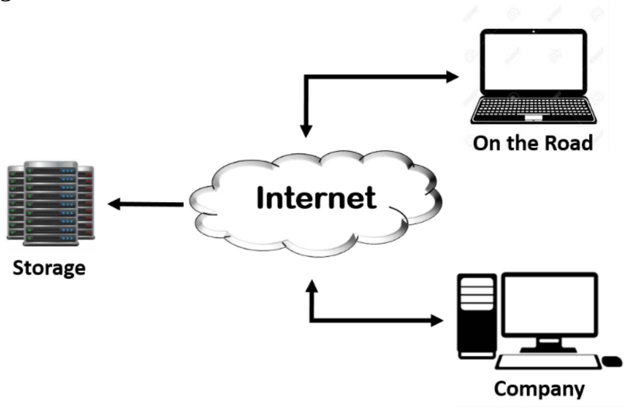

**Fig. 2** Cloud Storage Model.

## II. LITRATURE REVIEW

We have studied the data synchronization trends in various papers and found that these techniques contains many drawbacks. We will discuss these draw backs one by one.

### A. Data Overhead

Checksum perfecting is important only for sequential reads; if it is disabled, we can achieve the speed up of 15% due to lesser workload. Ext4-cksum decreases data flow rate on network and the bandwidth is wasted as too much data is consumed in checksum as shown in Table 1.

**Table 1** Data Transfer Overhead. This table highlights the difference between data transfer speed of normal ext-4 vs ext-4 checksum techniques.

| Work_ Load | Ext_4 | Ext_4_Cksum | Slow_ down |
|---|---|---|---|
| **File_ Server** | 79.58 **MB/s** | 66.28 **MB/s** | 16.71% |
| **Varmail** | 2.90 **MB/s** | 3.96 **MB/s** | -36.55% |
| **Web_ Server** | 150.28 **MB/s** | 150.12 **MB/s** | 0.11% |

*B. Less Consistency for Multiuser System*

Unlike single-client consistency, multi-client consistency requires the cloud server to be aware of Views, not just the client. For open source services like Seafile it can, but not for closed source services like, Dropbox. If multiple clients are trying to modify a file on the cloud at the same time, ViewBox fails to cater for this problem as highlighted in Fig. 3 [6].

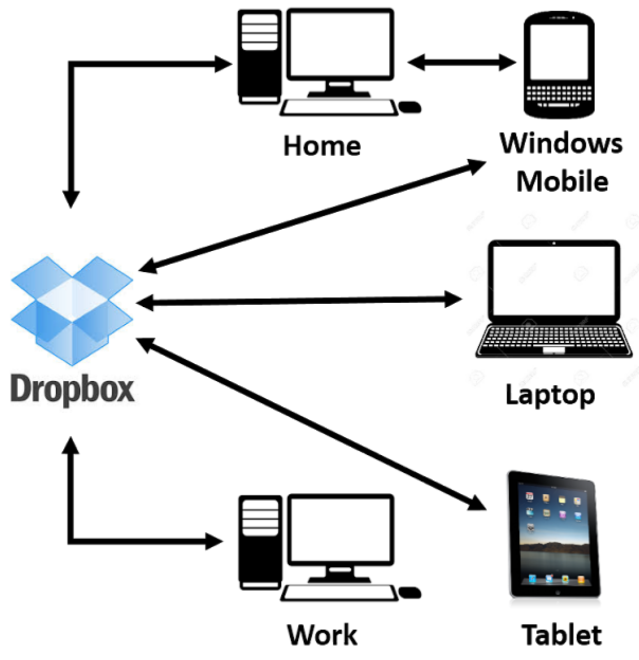

**Fig. 3** Simultaneous multiple users on a single cloud account.

*C. Cold Cache Latency*

If there are too many cache misses, View box and other could storage applications' efficiency decreases drastically. As shown in the Table 2.

**Table 2** Latency in Copy on Write (CoW). This table shows the response time in normalized form for Copy on Write operations.

| Operation | Normalized Response Time | |
|---|---|---|
| | Before C O W | After C O W |
| **Unlink (cold)** | 484.49 | 1.07 |
| **Unlink (warm)** | 6.43 | 0.97 |
| **Truncate (cold)** | 561.18 | 1.02 |
| **Truncate (warm)** | 5.98 | 0.93 |
| **Rename(cold)** | 469.02 | 1.10 |
| **Rename(warm)** | 6.84 | 1.02 |
| **Overwrite (cold)** | 1.56 | 1.10 |
| **Overwrite(warm)** | 1.07 | 0.97 |

*D. Checksum EXT-4 Compatibility*

EXT-4 is compatible with maximum number of users. Whereas ext4-cksum is not compatible with all of these users because it cannot be assured that every newer and older system takes this change from ext4 to ext4-chksum positively. Some systems may recognize ext4 file system but not an ext4-chksum file system. It is limiting the user base, which was claimed by ViewBox as an advantage over Dropbox and Seafile.

*E. Data Loss*

In case of unexpected application behaviour ViewBox rolls back to previous image of complete data, so client will lose all modifications made after that image synchronization [7]. There must be some step by step rolling back mechanism so that user's latest modifications don't get lost [8]. So users cannot rely on such methods for their important data.

## III. IMPLEMENTATION AND RESULTS

To overcome the problems discussed in the previous section, we propose a new tool called SafeBox, which uses Cyclic Redundancy Check [9] to detect errors in data, while storing it in cloud. We have chosen EXT4 [10] file system instead of ZFS file system, which is typically used in Dropbox and Seafile system, as it is compatible with the older systems that provides us with broad range of users. We have used the basics of Dropbox and implemented our own method to overcome the problems of data consistency and have tried to minimize the problems in ViewBox. We have used Ext4-crc file system to provide better Corruption detection and lesser data overhead, as shown in Fig. 4. Ext4 file system does not provide us usable detection of data corruption and information about consistency. Ext4-crc is used to overcome this limitation of data consistency faced in Dropbox as well as the problem of data overheads. Cyclic Redundancy Check [11] has very high level of error checking accuracy. In fact a lot of technology related experts believe that it is the most accurate error checking solution, when it comes to checking data in the form of blocks [11]. Additionally CRC gives us very little data overhead. CRC is much more reliable than other methods as a 16 bit CRC can detect 99.998% errors, which means only 0.002% of error propagation. If we use 32 bit CRC, than it will consume a little bit more data, can detect errors with up to 99.999999997% accuracy and only 0.000000003% error rate. This accuracy level is much better than what is typically achieved by other synchronization service providers.

SafeBox uses the (snapshots) image of file system, instead of saving full data of file system as backup. This method uses minimum band width while sending data to the backup server, so causes less data over head. To provide more consistency, we save backup of data at two point, one at remote user machine and another at our server storage. We keep frequent backups of our data at user machine and less frequent backup on the server storage. Both of these backups contain only one image of the file system each. Saving only snapshot of file system as backup, saves a lot of space that would have been used to keep all the data as backup.

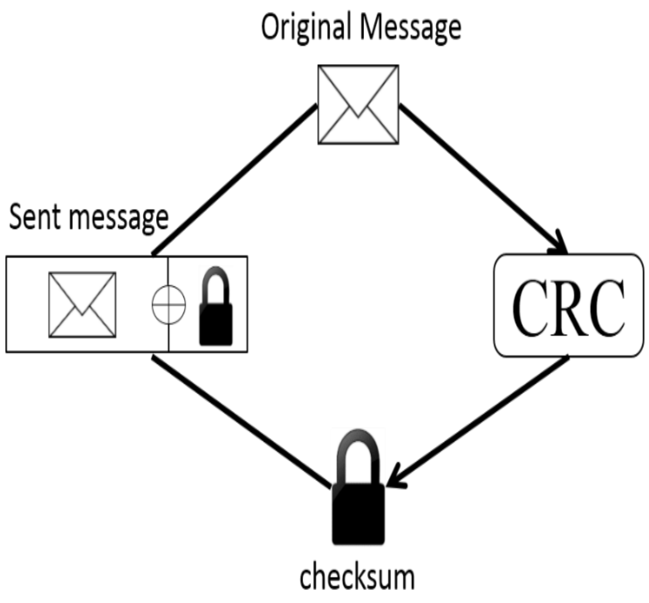

**Fig. 4** Cyclic Redundancy Check (CRC) Model.

*A. Storing Data on Cloud*

In order to store data on the cloud, we take an image of the file system of the data and save it as a backup on cloud storage system. CRC code is attached to this data's file system and this package (data+ext4-crc) is sent to the cloud storage. At the cloud's end CRC is checked if it indicates an error than a new file system image is taken and this data is not stored (saved as draft) on the cloud and a report including new image is sent back to client.

There are two types of errors that can occur during data transfer i) CRC code gets corrupted, ii) Data is corrupted. We take both cases one by one and check on client side that where the problem has occurred. First we compare the new image and the older one i.e. the one stored as backup, if these images are identical it means that the data was transferred successfully but the problem occurred in the CRC code section. In this case there is no need to send the data again and data from the draft can now be used to store data on the cloud. If images do not match, then we send the data from the backup on client side to the cloud again, as shown in Fig. 5. This process is repeated until data is correctly stored at the cloud. Every time a successful submission occurs at the cloud end a new image of the file system is taken and it overwrites the previous backup image (first we save new image if successful then we delete older backup image).

*B. Self-Consistency*

When the data is saved on the cloud, to maintain self-consistency i.e. data is not changed by some virus attack or by system failure on service provider's end, we keep on taking new file system images from time to time and store these file system images on a backup storage. If it does not match then the restoration mechanism is initiated to restore the original data.

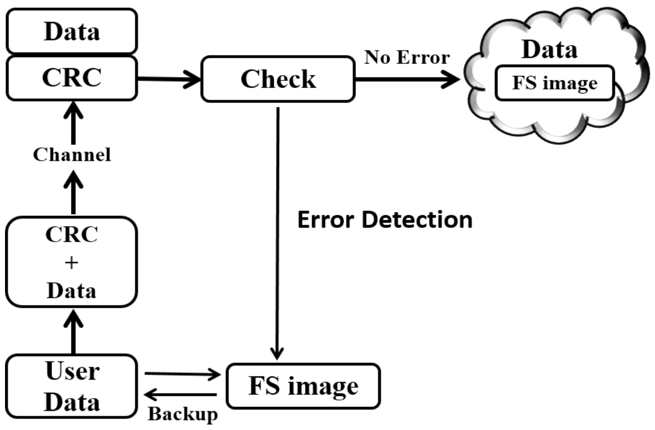

**Fig. 5** Storing Data on the Cloud.

*C. Retrieving Data from Cloud*

When cloud server receives a request from user to retrieve data from cloud, user downloads data from the cloud and also downloads its file system image (snapshot) backup stored on server. User also takes a snapshot of downloaded data and matches both, downloaded and new snapshot taken at user's end, if they do not match that means some data corruption has been taken place at server's end or during transfer of data. During the download CRC is again attached with file system image to detect errors as shown in Fig. 6. If this corruption is not detected, it can lead to data inconsistency due to corruption propagation, i.e. if the corrupted data is modified and then saved back to the cloud and this process is repeated, the original data may be permanently lost.

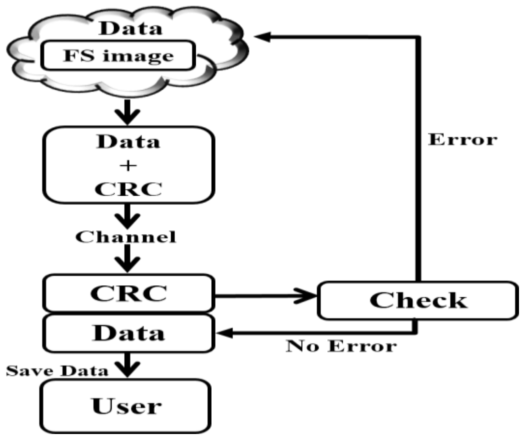

**Fig. 6** Retrieving Data from Cloud.

*D. Modifying Data on Cloud*

Modifying data on the cloud is basically sum of the two processes which have been discussed above. To modify data on the cloud we first have to retrieve data, then we alter it on remote client and save it back to the cloud. For multi-client user we only provide backup at the server end because user is changing its working environment frequently and cloud services do not have access to all the systems that users may be using at the time. So providing backup at user level is not possible in case of multi-client users. We use the backup provided at the cloud server because it is authentic and it does not conflict with any of the users' data.

## IV. CONCLUSION

In this paper we have presented a cloud storage solution that uses cyclic redundancy check to detect errors during data synchronization, we have found that error propagation is reduced significantly. Self-consistency check ensures much needed data consistency so that data does not change undesirably over time. Due to better error detection, our method uses less number of bits for error detection, which reduces data overheads that were faced in previously used methods. Our method reduces synchronization time and bandwidth usage during data transfer.

## ACKNOWLEDGMENT

The authors would like to thank Dr. Asad Waqar Malik, Assistant Professor, School of Electrical Engineering and Computer Science (SEECS), National University of Sciences and Technology (NUST), Pakistan for his help, support and guidance. The authors would also like to thank High Performances Computing (HPC) Laboratory members for providing assistance and hardware required to implement this research project.

## REFERENCES

[1] S. Umair, U. Muneer, M. N. Zahoor, and A. W. Malik, "Mobile computing: issues and challenges," in *2015 12th International Conference on High-capacity Optical Networks and Enabling/Emerging Technologies (HONET)*, 2015, pp. 1-5.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, pp. 50-58, 2010.

[3] A. R. Mohammad, K. Mohiuddin, M. Irfan, and M. Moizuddin, "Cloud the mainstay: growth of social networks in mobile environment," in *Cloud & Ubiquitous Computing & Emerging Technologies (CUBE), 2013 International Conference on*, 2013, pp. 14-19.

[4] S. Umair, U. Muneer, M. N. Zahoor, and A. W. Malik, "Mobile Cloud Computing Future Trends and Opportunities," *Managing and Processing Big Data in Cloud Computing*, p. 105, 2016.

[5] A. T. Velte, T. J. Velte, R. C. Elsenpeter, and R. C. Elsenpeter, *Cloud computing: a practical approach*: McGraw-Hill New York, 2010.

[6] Y. Zhang, C. Dragga, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "ViewBox: integrating local file systems with cloud storage services," in *Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST 14)*, 2014, pp. 119-132.

[7] R. Dobkin, R. Ginosar, and C. P. Sotiriou, "Data synchronization issues in GALS SoCs," in *Asynchronous Circuits and Systems, 2004. Proceedings. 10th International Symposium on*, 2004, pp. 170-179.

[8] T. Baicheva, S. Dodunekov, and P. Kazakov, "Undetected error probability performance of cyclic redundancy-check codes of 16-bit redundancy," *IEE Proceedings-Communications*, vol. 147, pp. 253-256, 2000.

[9] E. Posner and P. Merkey, "Optimum Cyclic Redundancy Codes for Noisy Channels," 1986.

[10] (July 2016). *Ext4*. Available: https://ext4.wiki.kernel.org/index.php/Ext4_Howto

[11] G. Castagnoli, S. Brauer, and M. Herrmann, "Optimization of cyclic redundancy-check codes with 24 and 32 parity bits," *IEEE Transactions on Communications*, vol. 41, pp. 883-892, 1993.