International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 www.ijisae.org Original Research Paper

File System RPO Snapshots in Near Real-Time with Asynchronous Replication

Sure Ravindra Reddy[1], Dr G Pardha Saradhi Varma[2], Prof Peri Srinivasa Rao*[3]

Submitted: 29/01/2024 **Revised**: 07/03/2024 **Accepted**: 15/03/2024

Abstract: In the File System level Disaster Recovery configurations, asynchronous data replication is often used between the local Primary File System and the remote Recovery File System to avoid latency to the applications running on the local Primary File System. The individual data updates of the local Primary File System are replicated to the remote Recovery File System in the background asynchronously after applying some delay called asynchronous delay. During the asynchronous delay, optimization methods like coalescing smaller contiguous write operations into a more extensive write operation and eliminating short-lived data updates are applied to the data updates to reduce the network bandwidth requirement for data replication. However, the asynchronous delay likely causes the delay in taking periodic Recovery Point Objective (RPO) snapshots on the Recovery File System for data consistency because of the large amount of data pending replication to the remote Recovery File System before taking RPO snapshots. This delay in taking RPO snapshots could cause more data loss, causing RPO violations if disaster hits the local Primary File System. Taking RPO snapshots strictly at RPO intervals is critical. This paper describes a new efficient procedure for taking RPO snapshots close to the RPO interval without delay by replicating pending data updates to the remote Recovery File System earlier without waiting for the asynchronous delay. Based on pending data replication, the aggregated network bandwidth between the local primary location and the remote recovery location, and the aggregated rate of data generated by applications, the early replication time before the next RPO time without waiting for the asynchronous delay is calculated.

Keywords: *RPO snapshots, asynchronous delay, RPO violation, Disaster Recovery, RPO snapshots near real-time*

1. Introduction

Computerized data has become critical to companies. Companies must recover their data should there be a disaster, such as floods, earthquakes or any other technical disruption that could destroy the local File System and cause data loss. To avoid data loss and inaccessibility to data in case a disaster hits the local File System, a replica of the local File System is configured at a remote Disaster Recovery (DR) location. The data changes on the local Primary File System are replicated to the remote Recovery File System for high availability. With the emergence of cloud technologies, the remote Recovery File System can be configured on a cloud [1][2]. The data replication from the local Primary File System to the remote Recovery File System can be synchronous or asynchronous. In the case of synchronous replication, the applications that are making updates on the local Primary File System need to wait till the changes are replicated on the remote Recovery File System, causing high IO latency and low throughput to the applications. In the case of asynchronous data replication, the application's IO latency is less but could cause data inconsistency [9] in the remote Recovery File System. For data consistency, the periodic snapshots [4] are taken at local and remote File Systems, which are called peer snapshots, to generate peer-consistent points at local and remote File Systems.

Once the disaster hits the local Primary File System, the applications fail over to the remote Recovery File System. The remote Recovery File System must be restored to the most recent snapshot before being used by applications to eliminate any inconsistency [9] in the File System. Restoring the File System to the most recent snapshot causes data loss (most recent changes). The amount of data loss is measured with RPO (Recovery Point Objective), which is defined as the maximum data loss acceptable to applications measured in time. The RPO defines the local and remote File Systems to take peer snapshots at the RPO interval; otherwise, the data loss would be more than acceptable, causing an RPO violation.

In asynchronous data replication to the remote Recovery File System from the local Primary File System, the data can be replicated online continuously as data is modified at the local Primary File System [8]. An asynchronous delay is

[1] *Research Scholar, AU College of Engineering, Andhra University, Vishakhapatnam, India*
ORCID ID=: 0009-0009-9913-4829
[2] *Professor, Computer Science and Engineering, KL University, Green Fields, Vaddeswaram, Guntur, India*
ORCID ID: 0000-0002-4885-1678
[3] *Professor, Computer Science and Systems Engineering, Andhra University, Vishakhapatnam India*
* *Corresponding Author Email:peri.srinivasarao@yahool.com*

applied for data changes before copying them to the remote Recovery File System. The asynchronous delay helps to reduce the amount of data replicated by not replicating the short-lived (example file created and deleted within asynchronous delay) data [5]. The asynchronous delay is also helpful for write-intensive applications because, in the asynchronous delay, multiple smaller contiguous writes are combined into a single extensive write operation. However, asynchronous delay causes the RPO violation because there would be some significant amount of data pending to be replicated to the remote Recovery File System before taking the RPO peer snapshot on the remote Recovery File System. Using asynchronous delay for better network bandwidth utilization by not copying short-lived data could cause RPO violations.

This paper introduces a new procedure in which asynchronous delay is used for data updates, but when it is close to taking snapshots, the pending data and new data are replicated without waiting for asynchronous delay so that no or little pending data would be present when it is time to take the RPO snapshot on remote Recovery File System. Based on the network's aggregated bandwidth and the aggregated data generation rate by the local Primary File System applications, early time to replicate data without asynchronous delay is calculated.

The following are our contributions.

1. We developed a method to start replicating the pending data and any new data on the local File System without waiting for the asynchronous delay.
2. The moving aggregated network bandwidth and moving aggregated rate of data generation are maintained.
3. When should data replication start without asynchronous delay before the next RPO start time? This is calculated by considering the aggregated network bandwidth and aggregated rate of data generated by applications.

In this work, we make the following assumption.

1. The remote Recovery File System is active, so data is replicated continuously.

This paper is organized as follows: Section 2 describes some related work. Section 3 briefly describes the IBM Storage Scale [7] and Asynchronous Disaster Recovery (ADR) [6]. Section 4 describes the procedure for starting replicating data without asynchronous delay so that RPO snapshots are taken at RPO intervals. Section 5 shows the experiments and results. Section 6 concludes.

2. Related Work

2.1. snapdiff-based replication

H. Patterson et al. [5] propose data replication from the local File System by replicating the modified data blocks to the remote File System asynchronously using snapdiff. The snapdiff-based replication is a mechanism that finds the differences between two snapshots at the local File System and replays them to the remote File System. The snapdiff takes a File System and two snapshots, S1 and S2. Internally, it runs an inode scan on the S2 and checks for all the inodes and directory entries created, deleted, and modified after S1. The output of snapdiff is a list of modified inode and directory entries. These modified snapshot difference entries are converted to filesystem operations, which can be replayed on the remote File System as part of snapdiff-based replication. The process identifies the modified blocks, and the data replication is done after the completion of the RPO interval, which is a violation of meeting RPO requirements. However, for the data block modified multiple times within the RPO interval, only the latest modification is replicated in the remote file system, thus optimizing the network bandwidth.

Umesh Deshpande et al. [1] propose incremental snapshot backup where the snapshot captures the changes performed on the volume from the previous snapshot. The snapshot changes are transferred to the backup repository for long-term retention. The incremental snapshots are scheduled more frequently than the RPO defined to meet the RPO guarantee. This incremental snapshot replicates more snapshots diffs to the backup repository, which consumes more data space on the backup repository.

2.2. Asynchronous Replication

Chao Wang et al. [3] proposed asynchronous data replication to the backup location, where data updates are scheduled to replicate to the backup location before the next RPO interval. Each data update is assigned the latest time before which they would be replicated to a backup location. The latest time is calculated from "time stamp of data update + RPO time – Travel Time". The travel time is calculated based on network bandwidth. This proposed method minimizes the data pending replication to a backup location just before taking an RPO snapshot. However, the latest replication times could be just before the next RPO time for most messages. More data updates could be pending to replicate before taking the following RPO snapshot. Also, the new data generated before the next RPO time may cause network contention, causing violations taking the following RPO snapshot.

2.3. Async delay in Asynchronous Replication

IBM's Active File Management [6] based ADR, File System or fileset level data replication system, uses Asynchronous Delay to delay the data replication to the remote Recovery File System for the user configured Asynchronous Delay time. During the asynchronous delay, multiple writes to the same set of files are replaced with a single write containing the latest data, which helps optimize the network bandwidth by not replicating any short-lived data to a remote File System. However, based on asynchronous delay, there could be some data pending to be replicated to the remote Recovery File System just before the RPO interval, which needs to be replicated before taking a peer snapshot at the remote site. The replication of the pending data causes a delay in taking RPO snapshots at remote sites, causing violations in meeting RPO requirements.

3. Background

The method introduced in this paper is implemented and verified in IBM Storage Scale [7] File System and Asynchronous Disaster Recovery [6]. Hence, this section reviews the IBM Storage Scale [7] and Asynchronous Disaster Recovery [6].

3.1. IBM Storage Scale

IBM Storage Scale, formally known as GPFS (General Parallel File System) [7], is IBM's high-performance shared disk cluster Parallel File System. Files are wide-striped across all disks in the File System for load-balancing. It also provides higher input or output performance by striping blocks of data from individual files over multiple disks and reading and writing these blocks in parallel.

The network that connects File System nodes to disks may consist of a general-purpose network, storage area network (SAN), Fiber Channels, or iSCSI using I/O server nodes. IBM Storage Scale uses a distributed locking mechanism to synchronize access to shared disks where all nodes share responsibility for data and meta-data protection and consistency while simultaneously providing parallel access to data and meta-data. The individual files of the File System are accessed in parallel, and different byte ranges of the same file are also accessed in parallel by different nodes. The distributed locking mechanism synchronizes the access to the same byte range of an individual file. By providing parallel access to files from different nodes, the throughput of the File System is maximized.

3.2. Asynchronous Disaster Recovery

IBM’s Active File Management [6] based ADR is a scalable and high-performance clustered File System or fileset level data replication system **Error! Reference source not found.** specially designed for parallel data-intensive applications. It is implemented and integrated within IBM Storage Scale to consistently replicate data and meta-data from the local Primary File System to the remote Recovery File System. Data updates are copied to the remote Recovery File System asynchronously in the background using either pNFS (industry-standard protocol for transferring data between the local and the remote site) or NFS or Aspera file transfer (high-speed file transfer protocol over WAN) or using S3 protocol to a cloud environment.

The updated data is replicated from the local to the remote File System asynchronously after some delay (asynchronous delay) but continuously as updates are made on the local Primary File System. If the local Primary File System experiences a site failure, the remote Recovery File System does not have all changes, nor does the data reflect any consistent state. However, a DR environment requires consistency. To provide consistent data replication, the user can define how often the consistent copies or snapshots should be taken so that the user can restore to the most recent consistent point or snapshot on the remote Recovery File System as required. These requirements are defined in RPO (Recovery Point Objective) settings. Based on these requirements, a snapshot (consistent point) is taken at the local Primary File System. Once all the data in the snapshot is pushed to the remote Recovery File System, a corresponding snapshot is created at the remote Recovery File System. This pair of peer snapshots reflects a consistent point for Disaster Recovery, which is taken periodically based on RPO value.

4. Synchronized RPO Snapshots in near real-time

We enhanced IBM's Active File Management [6] based Asynchronous Disaster Recovery (ADR) by proposing an early replication of the pending data to the remote Recovery File System such that at the next RPO time, there would be close to zero pending data to replicate to the remote Recovery File System, which helps to take the peer RPO snapshot at the remote File System close to real RPO time.

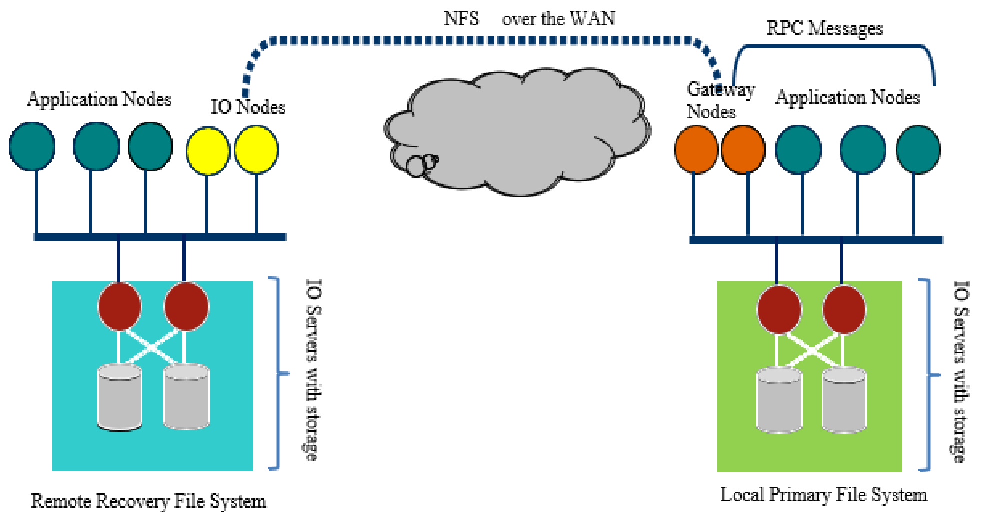

Fig 1. Disaster Recovery Architecture

In IBM's Active File Management [6] based ADR, a dedicated Gateway node for each fileset runs at the local Primary File System, as shown in Fig 1. Every node of the local Primary File System has access to storage disks, which enables the parallel applications to access and update the data from multiple nodes within a File System. The local Primary File System application nodes connect to a Gateway node. The application nodes send I/O operations executed locally on the local Primary File System to the Gateway node using a remote procedure call (RPC), as shown in Fig 1. Once the I/O operations are sent to the Gateway node, the application nodes return to perform the respective applications. However, the Gateway node stores the asynchronous data replication operations received from the application nodes in a queue. The queue manages the acquired operations by processing the operations received in a first-in-first-out (FIFO) manner. A single Gateway node can support multiple filesets for replicating the modified data from the local Primary File System to the remote Recovery File System asynchronously and continuously as data gets modified. The Gateway node also maintains the moving average rate (bytes updated or generated per second) of data generated and the bandwidth (sent per second to the remote Recovery File System) for individual filesets.

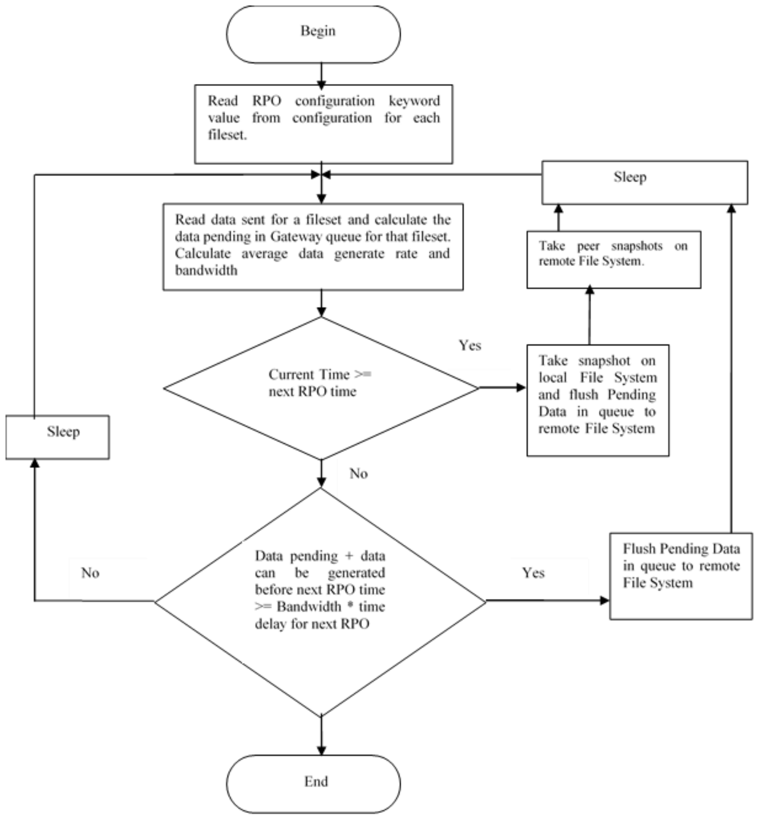
Begin
Read RPO configuration keyword value from configuration for each fileset.
Sleep
Read data sent for a fileset and calculate the data pending in Gateway queue for that fileset. Calculate average data generate rate and bandwidth
Take peer snapshots on remote File System.
Current Time >= next RPO time
Yes
Take snapshot on local File System and flush Pending Data in queue to remote File System
Sleep
No
No
Data pending + data can be generated before next RPO time >= Bandwidth * time delay for next RPO
Yes
Flush Pending Data in queue to remote File System
End

Fig 2. Taking peer RPO snapshots at near real-time RPO intervals

The average rate of data generated is calculated as the average data written or updated per unit of time (second) from the starting time (T0) of data replication of a fileset or File System to the current time (T1) using equation (1). The moving average of data generated is obtained by continuously recalculating the average data generated.

$$Rx = \frac{\sum \ size\ of\ data\ generated}{(\mathrm{T1}-\mathrm{T0})} \tag{1}$$

The bandwidth at which the data is transmitted to the remote File System is calculated based on the replication times of individual data updates replicated to the remote File System using equation (2). The transmission bandwidth is also periodically recalculated to get a moving average transmission bandwidth.

$$Bw = \frac{\sum \ Data\ size\ of\ data\ updates}{\sum Transmission\ time\ of\ data\ updates} \tag{2}$$

The Gateway node keeps the data update or written operations in the queue for Asynchronous delay time. It applies the Asynchronous delay optimized techniques to reduce the network bandwidth requirements before replicating them to the remote Recovery File System at the end of the Asynchronous delay. The flow chart in Fig.2 shows that the Gateway node runs an RPO Snapshot Manager, which reads the Recovery Point Objective as a configuration parameter for filesets. It monitors the data size pending in the queue for individual filesets. It also calculates the data that can be replicated to the remote File Recovery System before the next RPO time based on bandwidth and time to the next RPO. Similarly, the potential data needs to be replicated to the remote Recovery File System before the next RPO time is calculated based on the moving average of data generated. The sum of the data pending in the queue and the data that applications could generate before the next RPO time is calculated. As shown in Fig.2, for any fileset at any time, if the sum of data pending in the queue to be sent and the potential data could be generated by applications before taking the next RPO snapshot is greater than the data that can be replicated to remote Recovery File System before next RPO time, the queue is flushed by over-writing the asynchronous delay to ensure that the next RPO peer snapshot is taken close to real-time on remote Recovery File System without delay in taking peer snapshot at remote Recovery File System to meet the RPO time.

For example, the next RPO time is known and stored as the variable T_n and the current time is identified and stored as the variable T_l. In this case, the prediction for data generated is calculated using the formula "$R_x * (T_n - T_l)$", where R_x is the moving average rate of data generated by applications. If the amount of data pending in the queue is D, then the amount of data needed to be replicated before the next RPO time is calculated using the formula "$D + Rx * (T_n - T_l)$". The combined amount of data from the prediction and the pending data updates in the queue are compared to the amount of data that the Gateway can replicate before the next RPO time. The amount of data that could be replicated prior to the next RPO time is calculated using the formula "$BW * (T_n - T_l)$", where Bw is the calculated moving average transmission bandwidth. This value is compared with the $[D + Rx * (T_n - T_l)]$ value referenced above to determine if the transmission bandwidth is sufficient to move all the data pending and generated before the next RPO time, as shown in Fig 2. If not, the queue is flushed by overwriting the asynchronous delay to ensure that the following RPO peer snapshot is taken close to real-time on the remote Recovery File System without delay in taking a peer snapshot at the remote Recovery File System to meet the RPO time.

5. Experiments and Results

We performed some experiments to find the delay or lag in taking peer RPO snapshots at the remote Recovery File System after the corresponding local RPO snapshots at the local Primary File System. The experiments are done in two scenarios: the first time using asynchronous delay in data replication and waiting for pending data to be replicated before taking an RPO snapshot at the remote Recovery File System, and the second time with early flushing of the pending data in the queue by overwriting asynchronous delay before taking RPO snapshots.

We set up a local Primary File System and a remote Recovery File System, each having one node with IBM Storage Scale running. Both systems run the same RedHat OS level; the storage drives are HDD drives directly connected to the IO servers. The two nodes are connected over LAN using TCP/IP protocol. We created a fileset on the local Primary File System and another fileset on the remote Recovery File System. Then, an ADR (Asynchronous Disaster Recovery) relationship is established between them. We set the RPO interval to 5 minutes and the asynchronous delay to 1 minute. The NFS protocol is configured to replicate the modified data between local and remote File Systems.

In one experiment, we started creating files of size 1 MB on the local Primary File System for every 0.1 seconds. We noted the RPO start time on the local Primary File System and the RPO start time on the remote Recovery File System once RPO peer snapshots were taken. The time delay between these two is considered a delay or lag between RPO peer snapshots. As shown in Fig.3, we can see that with Asynchronous delay, the delay between peer snapshots is more than 25s. We repeated the same tests with our proposed solution by early flushing to flush any pending data to the remote Recovery File System. This time, the delay between peer snapshots on the local Primary File System and remote Recovery File systems is around 5 seconds. We repeated these tests a few times.

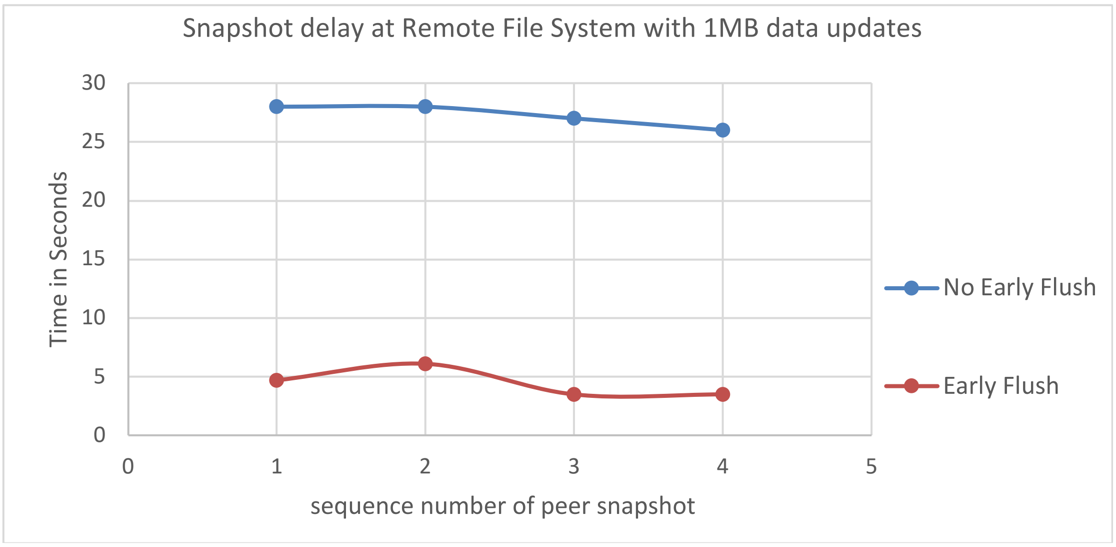

Fig 1. The delay or lag in taking peer RPO snapshots with 1BM data writes.

In the second experiment, we repeated the same experiment by creating files of size 10 MB on the local Primary File System every 0.1 seconds. We noted the RPO start time on the local Primary File System and the RPO start time on the remote Recovery File System. As shown in Fig.4, we can see that with an asynchronous delay, the delay between peer snapshots is more than 18 seconds. With our proposed solution, where early flush is used for flushing any pending data to the remote File System, the delay between peer snapshots is around 3 seconds.

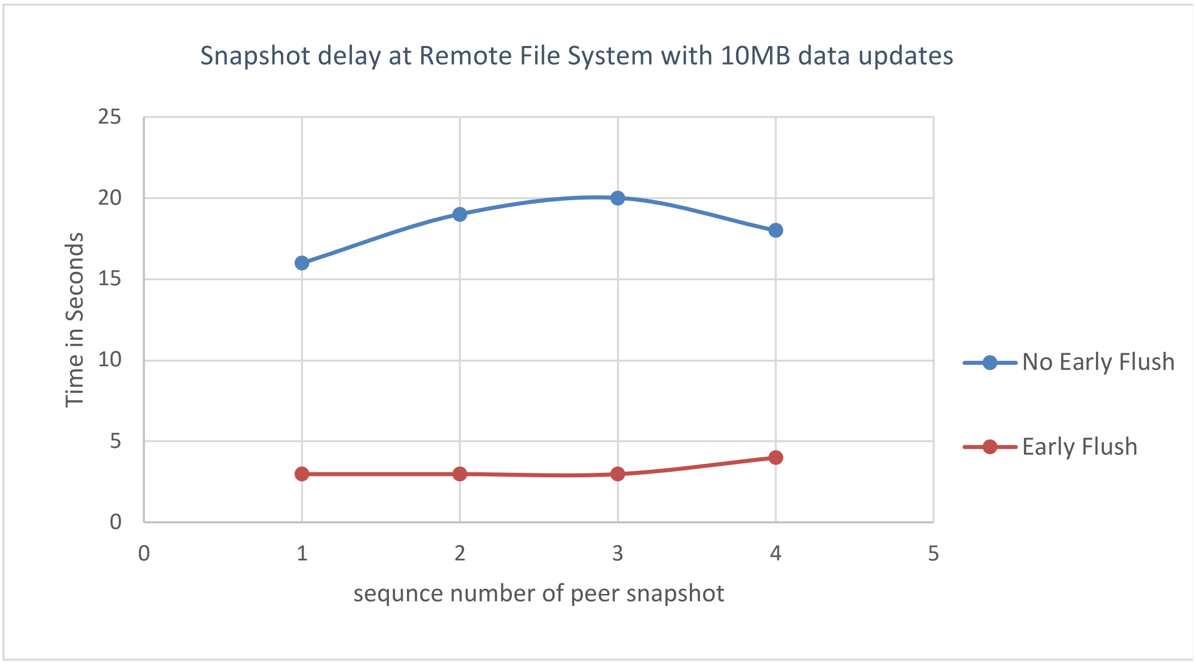

Fig 2. The delay or lag in taking peer RPO snapshots with 10BM data writes.

6. Conclusion

We observed that without our proposed solution, the peer snapshots at the remote Recovery File System are taken after flushing the pending data into the remote Recovery File System. That caused some significant delay in taking peer snapshots at the remote Recovery File System, causing RPO miss, which might cause consequential data loss if a disaster happens on the local Primary File System after taking the RPO snapshot on the local Primary File System but before taking the corresponding RPO snapshot on the remote Recovery File System. That is a violation of RPO. With our proposed solution, the RPO snapshot at the remote Recovery File System occurred close to the RPO snapshot at the local Primary File System. However, there is still a few seconds of delay between the snapshot taken at the local Primary File System and the snapshot taken at the remote Recovery File System. That is due to ignoring the time required for replicating meta-data operations into the remote Recovery File System. Because of that, there could be a delay in starting the flushing of data updates or writing operations to the remote Recovery File System.

References

[1] Umesh Deshpande, Nick Linck and Sangeetha Seshadri. 2021. Self-service Data Protection for Stateful Containers. In 13th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage '21), July 27–28, 2021, Virtual, USA. ACM, New York, NY, USA, six pages.

[2] J. Mendoca, R.Lima, E. Queiroz, E. Andrade and D. S. Kim, "Evaluation of a Backup-as-a-Service Environment for Disaster Recovery," 2019 IEEE Symposium on Computers and Communications (ISCC), Barcelona, Spain, 2019, pp. 1-6, doi: 10.1109/ISCC47284.2019.8969658.

[3] Chao Wang, Zhanhuai Li, and Kun Ren. 2010. ARPRG: An asynchronous replication protocol with RPO guarantee. International Conference on Computer Engineering and Technology 1 (2010), V1–611–V1–615.

[4] W. Xiao, Q. Yang, J. Ren, C. Xie and H. Li, "Design and Analysis of Block-Level Snapshots for Data Protection and Recovery," in IEEE Transactions on Computers, vol. 58, no. 12, pp. 1615-1625, Dec. 2009, doi: 10.1109/TC.2009.107.

[5] H. Patterson, S. Manley, M. Federwisch, D. Hitz, S. Kleinman, and S. Owara. SnapMirror: File System Based Asynchronous Mirroring for Disaster Recovery. In *Proceedings of the First USENIX Conference on File and Storage Technologies (FAST 2002)*, pages 117-129, 2002.

[6] Marc Eshel, Roger Haskin, Dean Hildebrand, Manoj Naik and Frank Schmuck. Panache: A Parallel File System Cache for Global File Access. In FAST'10 Proceedings of the 8th USENIX conference on File and Storage technologies

[7] F. Schmuck and R. Haskin. GPFS: A Shared-Disk File System for Large Computing Clusters. In Proc. of the First Conference on File and Storage Technologies 2000

[8] Ann Chervenak, Vivekenand Vellanki, and Zachary Kurmas. Protecting File Systems: A survey of backup techniques. In Proceedings Joint NASA and IEEE Mass Storage Conference, March 1998.

[9] S. Shumway. Issues in Online Backup. In *USENIX Proceedings of the 5th Conference on Large Installation Systems Administration*, pages 81–88, September 1991.

[10] Ananthanarayanan, R., et al. "Panache: a parallel WAN cache for clustered filesystems." ACM SIGOPS Operating Systems Review 2008:48—53.

[11] Manoj P. Naik and Ravindra R. Sure, "SNAPSHOTS AT REAL TIME INTERVALS ON ASYNCHRONOUS DATA REPLICATION SYSTEM," US Patent 9 983 947, May 29, 2018.S