Time Stamp based Multiple Snapshot Management Method for Storage System

Yunsoo Lee1, Dongmin Shin1, Insoo Bae1, Seokil Song1, Seungkook Cheong2

1 Dept. of Computer Engineering, Korea National University of Transportation, Chungbuk, Korea
yunsou.lee@gmail.com, gkdrmf23@gmail.com, eastnation28@gmail.com, sisong@ut.ac.kr
2 Electronics and Telecommunications Research Institute (ETRI), Yuseong, Daejeon, Korea
skjeong@etri.re.kr

Abstract. Snapshot has been key technology for data protection and disaster recovery. To support RPO (Recover Point Objective) and RTO (Recovery Time Objective) for data protection, multiple snapshots should be created in short period. In this paper we propose a multiple snapshots management method which solve the existing methods. In the proposed method, snapshots share common metadata and a snapshot volume. We show the efficiency of our proposed method through simulation.

Keywords: storage system, snapshot, logical volume

1 Introduction

Data protection and disaster recovery have been more important for storage system developers and administrators. Most storage systems use periodic backup and remote replication to protect data stored in storage systems [1]. Generally, snapshots are used in both backup and replication to enhance RPO (recovery point objective) and RTO (Recovery Time Objective).

Snapshot is the instantaneous image of storage system in specific time [2]. Logical volume snapshot is a feature that translates into easier backup management, faster recovery, and reduced exposure to data loss [3]. The snapshot feature is typically provided by storage companies like IBM (Tivoli Storage Manager), HP (Business Copy), EMC (SnapView), NetApp (SnapDrive), and Hitachi (Copy-on-Write Snapshot) [3]. Furthermore, logical volume snapshots make data mining of the data stored in storage devices easier by enabling users to take “snapshots” of the data at certain points in time [3].

In specifically, LVM (Logical Volume Manager) of Linux supports snapshot of logical volume. Snapshot of LVM is based on COW (Copy-on-Write). COW snapshots allocate a small volume with respect to the source volume when the snapshot is created. After the snapshot is created, upon a data block is updated the original data of the block is copied from the source volume to the snapshot volume. After copying, the update operation is performed on the block in the source volume. As a result, the data image at the time of the snapshot is preserved. The combination of the source volume and the snapshot volume presents the point-in-time image of the
data. After the snapshot is created, all subsequent read I/Os are performed on the source volume[1].

There are some problems in LVM snapshot, which reduce the availability and performance of storage systems [4]. First, it keeps the whole mapping information in memory, so it consumes many resources when the snapshot is very large. Second, if a source storage has multiple snapshots, a write operation will cause more than one COW to every snapshot. This feature reduces the storage performance seriously.

To solve those problems, ESnap [4] and ESnap2 [5] have been presented. They proposed a scheduling method to maintain metadata for snapshot to reduce the memory occupation of snapshot. The method stores all metadata on disks initially and whenever required, loads metadata into memory. Also, they proposed “dependent snapshot” to improve write performance. The dependent snapshot method chains all snapshots for the same original volume according their time sequence. The COW is only performed on the recent snapshot. ESnap2 additionally proposed a write enabled snapshot method.

However, ESnap still has some problems. First, it maintains metadata structures for snapshots respectively but only one COW block for a source block. Therefore, in order to read a COW block for a snapshot, all metadata for snapshots must be examined. Also, they do not consider RTO and RPO. To ensure RTO and RPO for storage systems, snapshots should be created in short period. With ESnap it is hard to create and maintain many snapshots according to the above reason.

In this paper, we propose an efficient snapshot management method for multiple snapshots. The proposed method maintains only one metadata structure for multiple snapshots, and snapshots share one COW volume.

## 2 Proposed Snapshot Management Method

![Overall architecture of proposed snapshot management system](image)

The overall architecture of our proposed snapshot management system is shown in Figure 1. To maintain multiple snapshots for a source storage, we create COW
volume, snapshot list and snapshot hash table. The COW volume stores COW blocks which are before image of updated blocks in source storage, in snapshot list, snapshot name and its creation time are recorded, and snapshot hash table is to record the address of COW blocks, COW time and their source block IDs.

In Figure 1 (a), there are 6 blocks in logical volume. The mapping table of logical volume maps logical block number to physical address in logical volume. In this figure, snapshot S1 is created at t1, and an entry which consists of snapshot name S1 and its creation time t1 is recorded in snapshot list. Then, a write request to update block 0 is given at t2. The before image (A) of block 0 is copied to COW volume, and an entry which consists of the logical block number, the physical address of the copied block in COW volume and COW time t2 is inserted into snapshot hash table. After that, the block 0 of logical volume is updated as A'.

In Figure 1 (b), snapshot S2 is created at time t3, and an entry (S2, t3) are recorded in snapshot list. At t4, block 3 (C) is updated to C', so the block 3 is copied to COW volume and an entry (2, 1, t3) is inserted into snapshot hash table. At this time, if a user want to read snapshot S1, the proposed snapshot management system retrieves blocks from logical block number 0 to 5. In order to read block 0, the proposed system first examines snapshot hash table whether block 0 exists in COW volume. If the block 0 exists in COW volume, the proposed system compares the time t2 with the time t1 of S1. If the COW time of block 0 is more recent and less recent than that of S1, the proposed system reads the block 0 in COW volume.

When the system reads block 3, it also examines snapshot hash table to check the block 3 exists in COW volume. The block 3 exists in COW volume, but the COW time of block 3 is more recent than that of snapshot S2, so we read block 3 from the source logical volume.

### 3 Performance Evaluation

In this paper, we evaluate our proposed snapshot management method through simulation. We assume that the storage size is 200Gbyte, block size of the storage is 2Mbyte, and maximum number of snapshots is 1000. The minimum number of physical IOs for COW snapshot is 3. First IO is to copy a source block to COW volume, second IO is to write the source block, and third IO is to sync metadata such as snapshot hash table. In this paper, we measure the average physical IO per a logical IO after creation a number of snapshots. We create snapshots from 200 to 1000 and we perform logical IOs about 300,000 times.

Figure 2 shows the performance evaluation results. As shown in the figure, the number of average physical IOs about 3. It means that the proposed snapshot management system shows optimal performance.
4. Conclusion

In this paper, we proposed an efficient snapshot management method for logical volume manager that multiple snapshots share common metadata structure and a COW volume. Through simulation, we show that our proposed snapshot management system shows optimal performance. In our future work, we implement the proposed method in Linux and evaluate it.

Acknowledgement. This work was supported by the IT R&D program of MKE/KEIT. (10039153, DRAM-SSD/HDD Hybrid Storage System & Management Solution Research & Development). Also, this research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (NRF-2012R1A1A4A01015615).

References

Time Stamp based Multiple Snapshot Management Method for Storage System