File Version Based Continuous Data Protection on Distributed Object Storage

Xin Yang[1]
College of Electronic Science and Engineering, National University of Denfence Technology
ChangSha, 410000, China
E-mail: `Cyang_92@163.com`

Ning Jing
College of Electronic Science and Engineering, National University of Denfence Technology
Changsha, 410000, China
E-mail: `ningjing@nudt.edu.cn`

Jiangjiang Wu
College of Electronic Science and Engineering, National University of Denfence Technology
Changsha, 410000, China
E-mail: `jiang_wu_820@sina.com`

Jun Li
College of Electronic Science and Engineering, National University of Denfence Technology
Changsha, 410000, China
`E-mail:junli@nudt.edu.cn`

Continuous Data Protection (CDP) can restore data to any point-in-time, but high storage overhead and drastic system performance drop restricts its application. In this paper, we propose a file version based file level CDP system (FV-CDP) by using cheap distributed storage for backup to low down the storage costs and using local object cache and parralel asynchronous object sending to mask network storage latency. It designs special opration log to identify the file system hierarchy at any point-in-time and exploits parallel restoring in filesystem recovery. The experimental results show that parallel asynchronous objects sending makes the FV-CDP system max write ops to get improved by about 3.4 times, and the parallel recovery reduces file system recovery time by up to 57%. Under high frequency file syetem change workload, FV-CDP causes a large storage space overhead.

CENet2017
22-23 July, 2017
Shanghai, China

[1]Speaker

1. Introduction and Related Work

As IT is becoming a data-driven business, the data loss can cause severe disaster, for example, the recent outbreak of the WannaCry viru resulted in a large number of PC loss data and caused a global panic. The traditionally periodic backup is taken for data protection. But for some applications, the Recovery Point Objective (RPO) periodic backup providing is too coarse to endure. For these scenarios, continuous data protection (CDP) can theoretically recover data to any point-in-time, but simultaneously CDP causes high overheads. Distributed storage has become a mainstream solution to large-scale data storage. It provides low-cost storage, high read/write bandwidth, which make it suitable for backup system and provide a new solution to CDP huge space overhead.

CDP system can be classified into block-level, file-level and application-level CDP. Block-level CDP systems work in the block interface layer and is easy for implementation. Previous CDP studies mainly focused on different aspects of block-level CDP. For the block-level CDP system architecture, Laden[1] proposed four different storage controller based CDP architectures. Trap-array [2] proposed a disk array based CDP model. Zhu et al.[3] proposed a iSCSI interface based CDP. In terms of optimizing CDP system overhead, Clotho[4] used a differential backup method to only back up the changed data portion between versions. Morrey III et al. [5] found deduplication to optimize the backup space overhead. Mariner[6] leveraged the track-based logging to minimize the write overhead. Differential backups caused large Recovery Time Objective (RTO). The ST-CDP[7] introduced a snapshot mechanism based on Trap-array, and gave the prediction model of the snapshot frequency influence on storage cost and recovery rate. The block-level CDP system is easy to be implemented, but it ignores the data background, resulting in difficulties in data consistency and usability examination. The file-level CDP system records file updates in file system level, the backup data get better consistence and the recovery process is more concise. In the file-level CDP system research, VMS[8] used a copy-on-write strategy for file version backup. OceanStore[9] used versioning in backup recovery process and introduced cache and replication mechanism. LBFS[10] realized the file version backup in remote area under low-bandwidth network. Ext3cow[11] impemented file version backup in the ext3 system. As it is cheap and scalable, the distributed storage has been gradually favored in data backup field. Wood et al[12] showed that backups placed on cloud storage were cheaper, and Cumulus[13] backed up local data to the cloud and provided backup data management. The cCDP[14] system used the cloud object store to backup the file system change log.

Compared with previous researches, our FV-CDP system is a file version based continuous data protection system (FV-CDP) with distributed object storage as the storage backend. Our FV-CDP implementation focusses on improving the write performance under high network storage latency and exploring parallel data recovery method for system restoring. Thus FV-CDP leverages local cache and asynchronously objects sending to optimize the system writing performance, presents a log-based method to identify filesystem hierarchy at any point-in-time, and exploits parralel restoring in filesystem recovery.

2. System Architecture and Implementation

This chapter introduces the system architecture and implementation in three parts. Section I shows the overview of FV-CDP system achitecture, Section II introduces the FV-CDP backup process and system optimization under high network storage delay and Section III introduces how to restore file system to any poin-in-time.

2.1 System Architecture

As shown in Figure 1, FV-CDP system consists of three parts, the local Linux node Filesystem in Userspace (FUSE) with file operation filtering capability, the back-end storage consisted of the distributed object storage cluster, between the two Swift Gateway is used as proxy sever for object storage management.

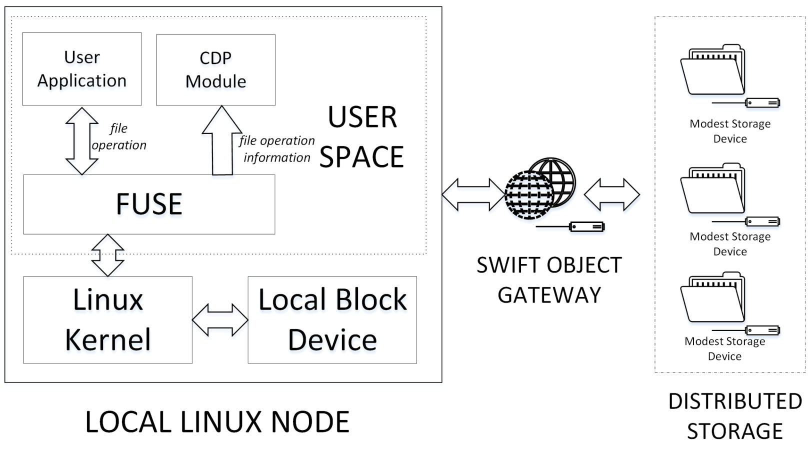


Figure 1: FV-CDP Architecture

FUSE is a software interface for Unix-like computer operating systems that allows users to create file systems in user space. We built a stackble FUSE based on local Linux xfs file system in user space. With FUSE, we added local xfs to file operation filtering function, through it we can get xfs operation information, including the operation type and corresponding data. After filtering the information, FUSE bypasses it to the FV-CDP backup module. After the backup module receives the operation information, it will decide which has to be backed up, and transfer the information into Swift objects and send these objects to the Swift Gateway.

Swift, as the OpenStack Object Store gateway, provides a simple and complete object storage management interface. The Swift Gateway acts as a proxy sever to send the objects to the backend distributed storage cluster. The backend storage uses Ceph object storage cluster.

2.2 FV-CDP Backup Module

As shown in Figure 2**,** when the operation information of the file system is acquired, it is represented by the *operation_info* object, which contains the path name, operation type, timestamp and corresponding data. This information is then sent to the backup module. In FV-CDP backup module, we further convert *operation_info* into two data objects, one is the structured operation log *op_record* object and the other is the unstructured *op_data* object. The *op_data* object is a copy of the file version corresponding to that timepoint, and it is simplely packaged into a Swift object with its name directly encoded by using path and timestamp.

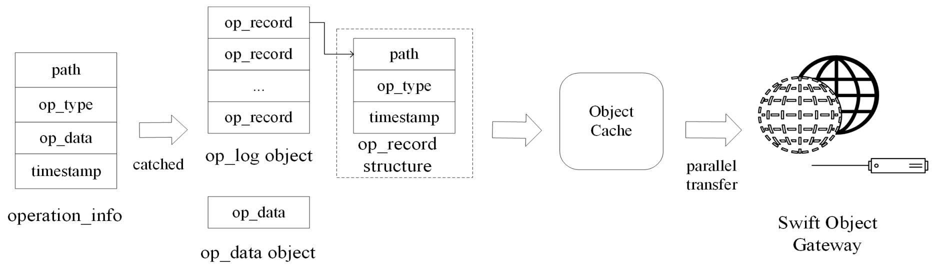


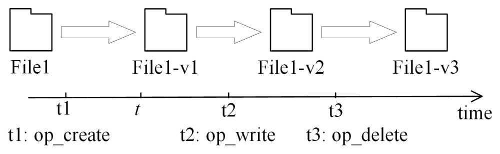
Figure 2: FV-CDP Backup Process

The *op_record* object is used to record operations on FUSE including the filepath, opration type such as create, write, delete and timestamp. *Op_record* is a very small object. If each record is stored as a Swift object to the back-end distributed object storage cluster, the storage delay will greatly affect the system performance. Thus, we aggregate multiple *op_records* to form an *op_log* Swift object. This *op_record* aggregation method reduces the average storage delay for a single *op_record*. In addition, the aggregate o*p_record* makes another convenience. The *op_log* object formed by multiple structured *op_record* objects is essentially a data table (*op_log* table), so that we can retrieve through the *op_log* table in database.

When *op_log* object and *op_data* object are encapsulated, they are sent to the Swift gateway. Simple synchronous network transmission can cause serious storage latency, it will seriously affect the system's write performance; therefore, all objects are firstly cached locally before sending and then parallelly asynchronously sent to the Swift gateway.

2.3 FV-CDP Restoring Implementation

Before restoring entire file system, we firstly look at how to use the file version backup to restore a single file to any point in time. Figure 3 shows the changes of a single file on the time axis. At *t1, t2, t3 and t4*, *file1* is manipulated, and these operations are logged to *op_log* object. If we need to restore *file1* to *t*, we take the instruction in Figure 3.



```
restore file1 to t
if t<t1, file1 not exists;
if t1<t<t2, create file1;
if t2<t<t3, fetch file1-v2 from Swift;
if t>t3, file1 not exists;
```

Figure 3: Restore File1 to Any Point-in-time

In order to restore the file system to any point in time, the file hierarchy at any point-in-time is also needed. The operation log object recorded all changes on the file system, so it can be used to restore the file system hierarchy. Normally, it is not simple to extract the file system hierarchy from the operation log; however, with the way proposed by us, the *op_log* is organized into a data table (*op_log* table). It is easy and efficient to do this task with database. FV-CDP does this task with MySQL. In order to restore the file system to the point in time *t*, it firstly fetched the *op_log* objects logged the operation before *t* or containing *t* from the Swift gateway. These objects are then loaded into the MySQL *op_log_table.* In MySQL server node, procedures in Figure 4 are excuted.

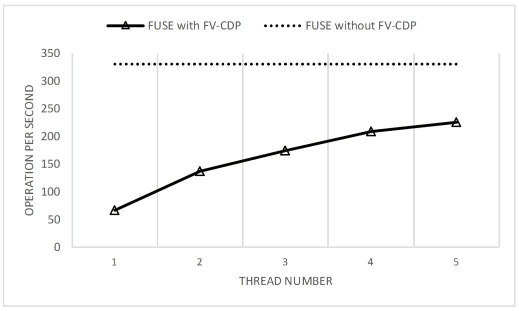
```
MySQL Procedure Description
step 1: create temporary table path_list, sys_ hierarchy ;
step 2: select distinct paths from op_log_table before restore point t ;
step 3: insert paths into temporary table path_list ;
step 4: for each path in path_list
        loop {
              select the latest timestamp that is before t ;
              select op_type for corresponding to (path, timestamp) ;
              insert into sys_hierarchy  values (path, timestamp, op_type) ;
               }
step 5: return table sys_ hierarchy ;
```

Figure 4: Restoring File System Hierarchy

When *sys_ hierarchy* table is obtained, in combination with the method of restoring a single file to *t*, FV-CDP can traverse the file system hierarchy and restore the entire file system to any point-in-time. File version based backup eliminates the interdependence of backed up dataI. In the recovery phase, FV-CDP can take full advantage of the independence between backup and parallelly restore the entire file system.

3. Performance and Evaluation

We firstly introduce the test environment settings, and the back-end storage usese Ceph object storage cluster, storage nodes are built on CentOS7 virtual machine with 2GB RAM, 100GB hard disk and 4 cores CPU. The Swift Object Gateway is also deployed on the CentOS with the same hardware settings. The database FV-CDP uses thesingle node MySQL sever built on virtual CentOS7 machine with 4G RAM, 100G hard disk and 4 cores CPU. Filebench is used as benchmark.



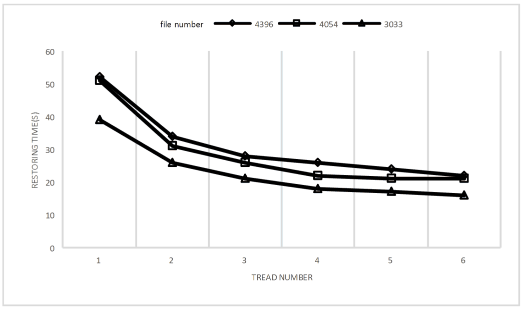


Figure 5: Max Write Perfomance **Figure 6:** Restoring Performance

For the system write performance test, filebench sequential write-file workload is used. The workload is set with 1 thread sequential write on the file set of 10k files and 128k per file size, and the I/O size is set to 64k. The test baseline is FUSE without FV-CDP mechanism. The results is shown in Figure 5. It shows that when FV-CDP mechanism is added, FUSE performance drops drastically from the baseline FUSE 330 write ops to sigle thread FV-CDP 67 write opsbecause under single thread situation, the synchronous backup data sending over network brings huge latency. This latency blocks writing and thus causes write performance drop. To average the network delay, FV-CDP introduces local object cache and uses parallel asynchronous objects sending. It improves the write performance greately. The experimental results show that parallel asynchronous objects sending makes the FV-CDP system max write ops to get improved by about 3.4 times.

In the recovery performance test, the workload is filebench file-sever workload, the worklaod is set 1 thread, filesize 128k, file number 10k, eventgen rate 30. We test the restoring time overhead to the selected points in time. There're 10 minutes since FV-CDP run with the entire filesystem 3033 files, 20 minutes 4054 files and 30 minutes 4936 files. Test result is shown in Figure 6, in which, FV-CDP parallel recovery has speeded up the system restoring. When the thread numbers exceeds a certain value, the added thread no longer decreases the restoring timebecause the time overhead is fixed when MySQL excutes procedurs in Figure 4. The experimental results show that parallel recovery reduces the file system recovery time by up to 57%.

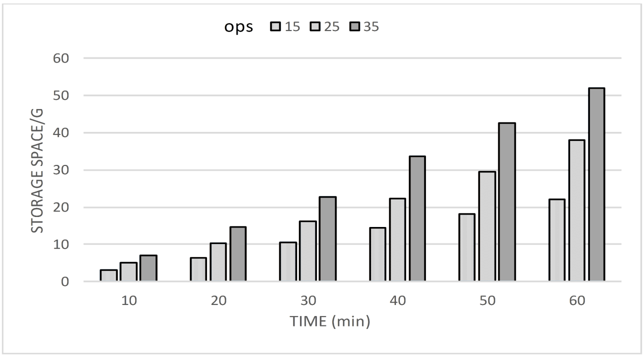


Figure 7: Storage Overhead

In the storage space overhead test, we use filebench file-sever workload, the workload is set under 1 thread, file set is file size 1M, file number 10k. The result is shown in Figure 7, in which, the storage overhead is increasing near linearly over time. 5G size file system under 35 ops workload takes approximate 55G storage space in 60 minutes, indicating that in the scenario of high frequency filesystem changes of FV-CDP has resulted ina larger storage overhead becaues FV-CDP backs up multiple versions of a single file independently. It helps parallel the system recovery, but causes a redundant backup.

4. Conclusion And Futurework

This paper proposes a file version based file-level continuous data protection system: FV-CDP, which uses FUSE to filter the file operation information and backs up the information into the distributed object storage cluster through Swift gateway. FV-CDP sends local cached objects asynchronously to the Swift gateway in parallel to mask the network latency. In the file system restoring phase, we design the method to determine the file system hierachy from operation log. FV-CDP can parallelly restore the entire file system because of independence of the backup data. The experimental results show that parallel asynchronous objects sending makes the FV-CDP system max write ops to get improved by about 3.4 times, and the parallel recovery reduces file system recovery time by up to 57%.

The FV-CDP system can cause significant storage overhead due to independent backup of all versions of the file. Future work will further explore the correlation between file history versions and introduce FV-CDP of deduplication technology to improve the system storage space overhead.

References

[1] LADEN G, TA-SHMA P, YAFFE E, et al. *Architectures for Controller Based CDP.* Proceedings of the FAST, F, 2007 [C].

[2] YANG Q, XIAO W, REN J. *Trap-array: A disk array architecture providing timely recovery to any point-in-time.* Proceedings of the ACM SIGARCH Computer Architecture News, F, 2006 [C]. IEEE Computer Society.

[3] ZHU N, CHIUEH T-C. *Portable and efficient continuous data protection for network file servers.* Proceedings of the Dependable Systems and Networks, 2007 DSN'07 37th Annual IEEE/IFIP International Conference on, F, 2007 [C]. IEEE.

[4] FLOURIS M, BILAS A. *Clotho: Transparent Data Versioning at the Block I/O Level.* Proceedings of the MSST, F, 2004 [C].

[5] MORREY III C B, GRUNWALD D. *Content-based block caching.* Proceedings of the Proceedings of the 23rd IEEE Conference on Mass Storage Systems and Technologies, F, 2006 [C].

[6] LU M, LIN S, CHIUEH T-C. *Efficient logging and replication techniques for comprehensive data protection*. Proceedings of the Mass Storage Systems and Technologies, 2007 MSST 2007 .24th IEEE Conference on, F, 2007 [C]. IEEE.

[7] YANG J, CAO Q, LI X, et al. *ST-CDP: Snapshots in TRAP for continuous data protection* [J]. IEEE Transactions on Computers, 2012, 61(6): 753-66.

[8] MCCOY K. VMS file system internals [M]. Digital Press, 1990.

[9] RHEA S C, EATON P R, GEELS D, et al. *Pond: The OceanStore Prototype.* Proceedings of the FAST, F, 2003 [C].

[10] MUTHITACHAROEN A, CHEN B, MAZIERES D. *A low-bandwidth network file system.* Proceedings of the ACM SIGOPS Operating Systems Review, F, 2001 [C]. ACM.

[11] PETERSON Z, BURNS R. *Ext3cow: A time-shifting file system for regulatory compliance* [J]. ACM Transactions on Storage (TOS), 2005, 1(2): 190-212.

[12] WOOD T, CECCHET E, RAMAKRISHNAN K K, et al. *Disaster Recovery as a Cloud Service: Economic Benefits & Deployment Challenges* [J]. HotCloud, 2010, 10(8-15).

[13] VRABLE M, SAVAGE S, VOELKER G M. *Cumulus: Filesystem backup to the cloud* [J]. ACM Transactions on Storage (TOS), 2009, 5(4): 14.

[14] MANDAGERE N, ROUTRAY R, SONG Y, et al. *Cloud object storage based Continuous Data Protection (cCDP)*. Proceedings of the Networking, Architecture and Storage (NAS), 2015 IEEE International Conference, F, 2015 [C]. IEEE.