IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 18, 2019, accepted October 5, 2019, date of publication October 17, 2019, date of current version October 30, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2948097

Efficient Metadata Management in Block-Level CDP System for Cyber Security

HONGYAN LI[1,2], FENGJUN XIAO[3], AND NAIXUE XIONG[4], (Senior Member, IEEE)

[1]School of Information and Communication Engineering, Hubei University of Economics, Wuhan 430205, China
[2]State Key Laboratory for Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430072, China
[3]Management School, Hangzhou Dianzi University, Hangzhou 310018, China
[4]School of Computer Sciences and Technology, Tianjin University, Tianjin 300000, China

Corresponding author: Fengjun Xiao (bhxfj@126.com)

This work was supported in part by the Natural Science Foundation of China (NSFC) under Grant 61572012, and in part by the Science and Technology Research Project of the Education Department of Hubei Province under Grant D20172203.

ABSTRACT Continuous Data Protection (CDP) is an emerging scheme of data backup and recovery for Cyber Security. Compared with traditional data protection techniques, it can provide much higher reliability and allow the data state to be reverted to any point-in-time in the past. CDP continuously captures and logs every disk update, producing large amount of information, thus desperately requires an efficient mechanism of organizing such information (about the disk I/Os) in order to guarantee the performance to be acceptable. However, there are few works which address the organization of history data to improve the recovery efficiency in case of the disaster occurrences. In this paper, three methods of metadata management for Cyber Security are presented, of which two are simple implementations based on MySQL database (named DIR_MySQL and OPT_MySQL respectively) and the other one is designed with consideration of its characteristics. Experimental results show that META_CDP method is far more efficient than the other two and its performance is reasonably acceptable. Furthermore, we discuss in-depth two recovery algorithms, which are full-recovery and incremental recovery. Users can choose the optimal recovery algorithm based on the required recovery time point.

INDEX TERMS CDP, data backup, recovery time, metadata management, security.

I. INTRODUCTION

The storage system's reliability is very fragile. How to ensure the data reliability is an important problem that the storage system needs to solve for cyber security. With the advent of big data and cloud computing applications, data is increasingly dependent on network and the security threats are becoming more complex and diverse.

Data is more and more important for companies as the level of informatization continues to increase. Data is the foundation of companies' survival and development and data security is even more critical for business. Data loss or leakage will cause huge losses to the companies. Every year, companies make huge investments in data storage and data security. However, storage systems are constantly exposed to a variety of hazards, including hardware damage, natural disasters, power hazards, virus intrusions, operational errors, software defects, etc. How to protect data from various hazards is the primary consideration for system designers and data managers. According to the characteristics of security threats, security threats can be broadly classified into 'hard threats' and 'soft threats. Existing disaster recovery technologies such as data backup and mirroring can protect data against soft or hard threats. And continuous data protection technology can make storage systems recover from soft threats.

A periodic backup is a process in which data in a storage system is collectively transferred to a specified server at regular intervals, such as once a week. Each time a data backup operation is performed, a data image version corresponding to the backup time point is stored on the server side. When a data disaster occurs, the data image closest to the time of the disaster is obtained from the server. However, the periodic backup technology has its own defects. Firstly, backup technology has greater impact on applications. When backing up data, it's necessary to stop running business. And the downtime is related to the amount of data that needs

The associate editor coordinating the review of this manuscript and approving it for publication was Jinming Wen.

to be backed up. The larger the amount of data, the longer the downtime backup window is needed. It's unacceptable in some application environments. Secondly, when disaster occurs it can only provide the data image in time closest to the point of disaster. Data changes that occur between this backup point and the disaster point will not be available. So there is still a risk of data loss and the maximum amount of data loss is the data generated within a backup window. How to properly and effectively store data and implement data protection is a problem that enterprises must face and solve in the information age.

CDP [6] is a new data protection technology for cyber security. It is proposed to solve the shortcomings of traditional backup technologies that require downtime for backup and large data loss during disasters. The Global Network Storage Industry Association defines CDP as: Continuous data protection is a method that captures or tracks changes in data and stores them separately outside of production data to ensure that data can be restored to any point in the past. Continuous data protection is the continuous protection of storage system without specifically shut down the system. It continuously tracks and saves state changes that occur in the storage system and has an image version that stores the state at any point time. Compared with the backup technology, it can restore the storage system state to the point of infinite proximity to the disaster occurrence point, not just a limited number of backup time points, so the data loss is greatly reduced. Therefore, enterprises usually use a combination of backup and continuous data technology to ensure reliable and secure data. Periodic backup data as a data reference version and use continuous data protection technology between two backup points to reduce the amount of data lost. It continuously tracks data changes in real time and logs them to a dedicated storage device or to a dedicated server over network to restore the protected data to any point in time when a failure occurs. CDP is an emerging data backup and recovery method that is more reliable than traditional data protection technologies and provides customers with a point-in-time at any time point. CDP continuously captures and records all disk operations on a protected disk and generating large amount of information. So an effective I/O record (information describing each I/O operation) organization is the key to CDP system performance.

Recovery Point Objective (RPO) and the Recovery Time Objective (RTO) are two concepts closely related to the disaster recovery technology. They are the key indicators for measuring the emergency response capability and data protection of a disaster recovery technology. RPO refers to the point time that can be recovered when disaster. It corresponds to the granularity of data protection applied to the system and reflects the amount of data loss that may be caused by disaster. RTO refers to the time required from the system startup recovery operation to the completion of the recovery operation. It is affected by data protection methods, the amount of data that needs to be recovered and data organization. The ideal RPO for disaster recovery technology is 0, that is when a disaster occurs, the system can quickly recover to available state. In general, RTO for continuous data protection is much smaller than data backup technology because it only needs to recover data modifications occurred since the system was started rather than restore all data sets like backup.

CDP can be divided into application level, file level and block level according to the implementation location. Application-level continuous data protection is often combined with specific application such as a specific database management system. It ensures data protection consistency by tracking application layer changes and utilize the protocols provided by the application system. File-level continuous data protection, combined with specific file system, it tracks data changes by tracking file system calls and can protect data for individual files, file directories or entire file systems. The diversity of file systems limits the widespread use of file-level continuous data protection systems and making it lack compatibility and portability. Block-level continuous data protection captures data changes at the block device level regardless of the upper layer application, which resulting in greater flexibility and portability. According to the protection granularity, the continuous data protection system can be divided into ''accurate'' and ''near'' continuous data protection systems. The '' accurate '' continuous data protection system tracks and preserves every operational change in the system, while the '' near'' continuous data protection system interval tracks the state changes. Continuous data protection systems typically have two implementations: indirection and copy-on-write (COW). Generally, indirection applies to implementations of application-level and file-level continuous data protection systems, while copy-on-write is appropriate for implementations of block-level continuous data protection systems.

CDP has obvious advantages in data protection compared with traditional data protection technology. CDP can greatly improve RPO and RTO. Data protection interval of traditional backup technology is 24 hours usually, so users are at risk of losing data for up to 24 hours. Using snapshot technology can reduce the risk of data loss to a few hours, while the amount of data lost by CDP can be reduced to a few seconds. In fact, traditional data protection technology adopts the management mode of ''single time point'' (Single-Point-in-Time, SPIT) data copy, while continuous data protection can achieve data access to ''Any-Point-In-Time'' (APIT).

The comparison of CDP with backup, snapshot and replication technologies is shown in Figure 1. It can also be seen from Figure 1 that although replication technology can obtain the latest data state by synchronizing with production data, it cannot avoid data loss caused by artificial logic errors or virus attacks. When production data is destroyed due to the above reasons (for example, data is deleted by mistake), replication technology will synchronize the destroyed data state to the backup data storage system which will also destroy the backup data. CDP system can restore the data state to any point in time before data is destroyed.

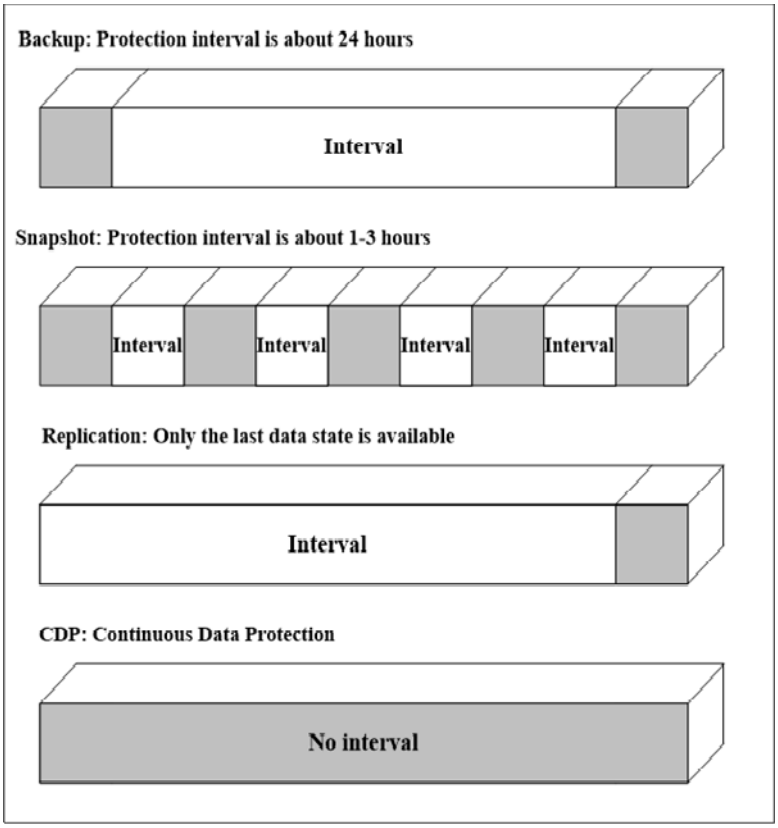

FIGURE 1. Comparisons between CDP and traditional data protection technologies.

In addition, because of the finer granularity of recovery time and recovery objects, the data recovery of continuous data protection is also more flexible. At present, some products and solutions allow end users (not just system administrators) to restore data directly, which greatly facilitates users. According to this feature, some researchers pointed out that CDP technology may even replace version control functions of some systems, so that developers focus on development and design.

The rest of this paper is organized as follows: Related works on the continuous data protection techniques is briefly reviewed in Section 2. Section 3 gives the overall architecture of the Net-Oriented Block-Level CDP System. We elaborate on our design and implementation of metadata management in Section 4 and evaluate our approach via prototype implementation in Section 5. Finally, we conclude the paper in Section 6.

II. RELATED RESEARCH

As mentioned before, the key to implementing continuous data protection techniques is to store historical data and retrieve corresponding historical data upon recovery. For fine-grained continuous data protection systems, the system will produce large amount of historical data. How to manage these historical data is one of the important problems facing continuous data protection systems. Q. Yang et al. proposed a new block-level continuous data protection system TRAP. TRAP stores not only the data stripe upon a write to the array, but also the time-stamped Exclusive-ORs of successive writes to each data block. It is able to recover data very quickly to any point-in-time upon data damage by tracing back the sequence and history of exclusive-ORs resulting from writes. What's more it is space-efficient.

Compared with traditional data protection technology, CDP has many characteristics and advantages [7]. CDP can provide unlimited RPOs and almost zero RTOs. RPO refers to the available recovery time point, which directly affects the amount of data that may be lost when disaster. RTO refers to the time period from the system shutdown and begins to recover until the recovery system is re-usable after disaster occurs. These features and benefits of continuous data protection technology are key to ensuring business continuity and greatly enhance system reliability. According to the implementation hierarchy, CDP can be divided into application level, file level and block level.

CDP technology has attracted a lot of attention in industry and academia in the few years since it advents. Many IT companies such as IBM, Microsoft, EMC and Symantec have developed relatively mature products. Reference [8] proposed four continuous data protection architectures in the storage controller based on the location of the data captured on the write path and analyze the performance and space overhead of each architecture. Reference [9] solved the problem of implementing CDP portability in kernel mode, proposes four

implementations of NFS server user state CDP and compares their performance. Since the CDP system needs to continuously record each data block, the CDP system itself will generate a large amount of data. TRAP [10] solved the problem of storage space in CDP system. It does not store the contents of each updated data block but instead stores non-zero data in the XOR results of the updated content and historical data. Compared to the state-of-the-art continuous data protection technologies, TRAP saves disk storage space by one to two orders of magnitude with a simple and a fast encoding algorithm. However, this method has a longer recovery time in some cases. Literature [11] shortened the retrieval length at the time of recovery by inserting a periodic snapshot in its parity coding chain to reduce the recovery time. Literature [12] reconstructed file versions from disk block versions maintained by a block-level CDP. It reconstructs file versions by following the last modified time of files and directories, a common file metadata supported by almost all modern file systems, and therefore does not require any modification to the host file system that a block-level CDP system protects. ShiftFlash [14] utilized the physical characteristics of SSD itself to implement the CDP function in the SSD device. Experiments show that the CDP overhead and performance are much higher than the TRAP and Ext3cow.

CDP is a method to track variable data and restore it to any previous point in time while ensuring the normal operation of main data. CDP technology can provide block-level, file-level and application-level data protection. Then, how to choose a backup approach suitable for user applications? The following is a brief analysis of three levels of data protection methods.

A. APPLICATION-LEVEL CDP

The main features of application-based CDP system include less data transmission, easy parallel operation or read-write separation. Typical products such as Oracle Golden Gate, Data Guard and so on. It is not easy to achieve full synchronization protection (RPO $= 0$) because transmitted over ethernet, application-level CDP has significant impact on application performance. And there are some unstable factors which need a mechanism to avoid writing conflicts and cannot defend against the damage of the underlying database structure.

By contrast, block-level and file-level CDPs support more data types. At the same time, they can do many-to-one on backup targets. CDP products based on data block protection are mainly for capacity and performance, and the number of protected client systems is unlimited.

B. FILE-LEVEL CDP

File-level CDP transfers variables to backup nodes or devices by monitoring data changes at file system level. Real-time or timed file replication is relatively simple to implement, but due to the transmission based on TCP/IP protocol, limited by network bandwidth and overhead, if synchronization is to be achieved, the performance of the application system will be greatly affected.

C. BLOCK-LEVEL CDP

Block-level CDP does not have these limitations. Its data writing and splitting implementation is located in the disk driver layer which is the under layer of the file system. If it is synchronized protection, it is similar to RAID1 mirror writing that without disk buffers or occupying only a small amount memory. For example, to achieve RPO $= 0$ and RTO $= 0$, block-level CDP is needed. In addition to supporting I/O granularity data rollback, when users deploy Oracle ASM and UNIX/Linux LVM, some systems can also realize automatic takeover of storage failures.

One of the major advantages of block-level CDP is the use of FC or iSCSI storage network protocols. Some CDP products take full account of data consistency, and provide consistent snapshot technology for applications such as Oracle. SQL Server which have strict requirements in this respect. This technology is different from traditional snapshots. It uses a passive mechanism to sense the action of database cache brush disk and tags the I/O records to ensure the effective recovery of data in various applications. Latest data protection system adds file system parsing function to the Web management interface, which can only synchronize the data blocks occupied by the file system on disk, thus combining the advantages of file-level CDP.

In short, CDP implementation based on application system only aims at protected application system, which occupies more resources of application system. Based on file system, FAT tables cannot be copied until they have been written. At the same time, data needs to be read first, which consumes a lot of system resources and increases network bandwidth dramatically. Block-based I/O-level replication technology provides block-level protection from OS interference.

III. NET-ORIENTED BLOCK-LEVEL CDP SYSTEM

The CDP System needs to record the data of each disk write operation and store these I/O information for rollback of the data state to the previous point in time state when recovery is required. The information of each I/O operation includes metadata information describing the I/O attribute and I/O data content. Each metadata information defines the state of a block of data on a disk at a particular time. It includes the disk location (sector number), written content and the size of the I/O operation of an I/O write operation. The set of content states of all the data blocks on the logical volume at a certain moment forms the state of the logical volume at that moment. Therefore, restore the logical volume to the state at a certain moment, it is only necessary to restore all the data block contents on the logical volume to the state at that point in time. The state of the data block is different at different times, and the state of the data block at a certain moment can be obtained by the metadata record corresponding to the data block. Therefore, the management of metadata is crucial to the entire CDP system, which directly affects the correctness

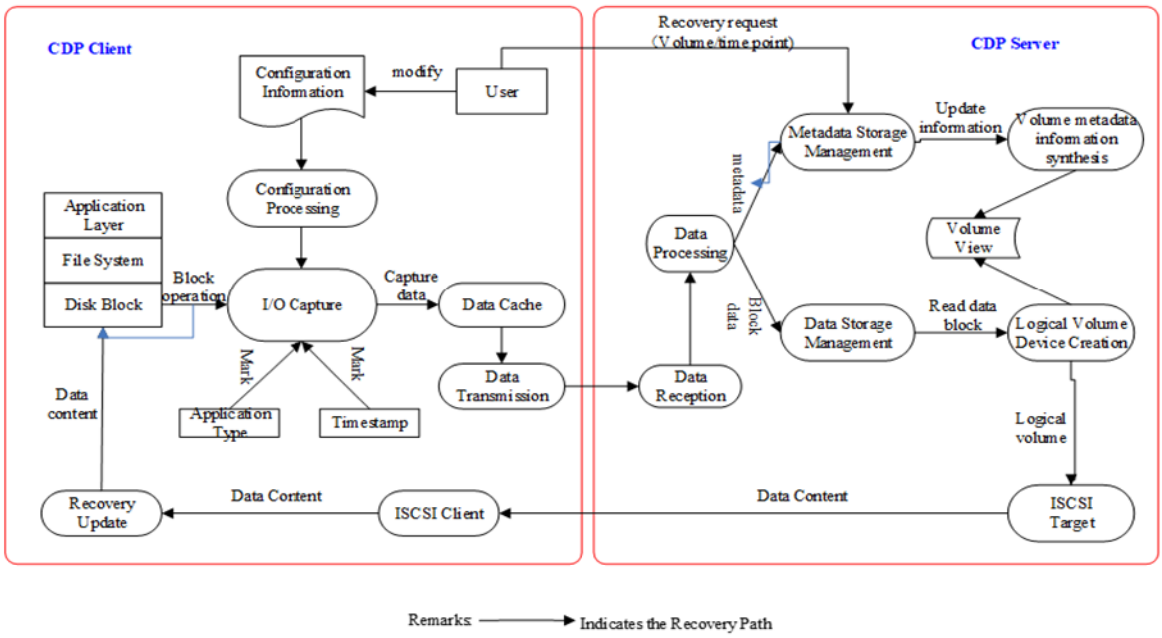

FIGURE 2. CDP system overall architecture.

and efficiency of the system. Therefore, designing an efficient metadata management mechanism and method is one of the keys to implementing CDP systems.

Based on this idea, we have developed a ''network-oriented block-level CDP system'' prototype. The prototype system uses a C/S mode architecture that provides concurrent online continuous data protection for multiple clients. Each client runs a CDP client, configures the logical volume through configuration interface, and the client block-level capture module continuously captures all write I/Os that occur during the protection period of the volume and send the metadata and data content information of each write I/O operation to the CDP server. The CDP server organizes storage metadata and data content information on a user-by-user basis. When the client makes a recovery request, it first retrieves the state of all the data blocks on the volume, then mounts the view of the data volume to the client through the iSCSI mechanism to achieve recovery purposes. The architecture diagram of the prototype system is shown in Figure 2.

IV. METADATA MANAGEMENT DESIGN AND IMPLEMENTATION

CDP system should be able to support multiple rollback recovery operations and it can continue data protection after each rollback recovery operation. In order to ensure that the performance of recovery operations is within an acceptable range, metadata management must provide an efficient retrieval interface which proposed high demands on metadata management design. During the development process of the system we have implemented three metadata management methods, and each management method provides two recovery retrieval algorithms: Full and incremental recovery algorithms. The first two are based on the MySQL database. Another is a management method optimized for the database called OPT_MySQL. And the third management method designed according to the actual characteristics of CDP is called META_CDP.

A. IMPORTANT DATA STRUCTURE

Metadata: Describes the structure of each write I/O attribute information. From the block level perspective, each I/O operation corresponds to a bio operation. It is defined as:

```
typedef char HANDLER[HANDLER_LEN];
typedef struct WOPS_t_str       {
uint64_t seq_num;
char timestamp [TIME_LEN];
uint64_t       src_block;
HANDLER dst_block;
uint32_t size;
} WOPS_str;
```

where timestamp is the time when the I/O operation occurs, its precision is seconds; seq_num is the serial number identifying the I/O operation, because there may be multiple write I/O records for the same data block in one second; src_block is the starting location (source address) of the write I/O operation. dst_block indicates the storage identifier (destination address) of the content of the write I/O on the server storage and the content corresponding to the data block when recover is obtained by the identifier. Size is the amount of data for the write I/O operation.

Branch: A branch corresponds to a recovery operation and the client creates a new branch record for each recovery operation. It is defined as:

```
#define       TIME_LEN 15
typedef struct BRANCH_t_str{
char revert_to_time[TIME_LEN];
```

```
char start_time[TIME_LEN];
char end_time[TIME_LEN];
} BRANCH_t;
```

where revert_to_time represents the RPO of this recovery operation; start_time indicates the start time of the branch; end_time indicates the end time of the branch. Due to the one-dimensional continuity of time, the end time of the previous branch is equal to the start time of the next branch. The RPO of the first branch and the end time of the last branch are set to specific values. The branch concept is the key to supporting multiple rollback operations.

Metadata record result: The state corresponding to the data block at the RPO time after the recovery operation retrieval is completed. It is defined as:

```
typedef      char HANDLER[HANDLER_LEN];
typedef struct {
uint64_t      src_block;
HANDLER dst_block;
uint32_t size;
uint64_t seq_num;
      } META_RES;
```

Among them, the meaning of each member is consistent with the members in the metadata.

B. FULL RECOVERY ALGORITHM

Full recovery firstly restores the data state to the initial state, then retrieve all the write operation records from the start time to the RPO. For records of the same data block, select the record which time point is close to the RPO; For records with the same data block and time point (seconds), select a record with a large serial number. The I/O operations described by the retrieved metadata are then overwritten to the starting state. The algorithm is described as follows:

Let the initial time be t0 and RPO specified by user as t', perform the following steps:

Find the time branch where t' is located

Traverse all branch records and find records for t'<end_time and t'>start_time.

Put all metadata records between [start_time, t') into the result set.

If the start time of the t' branch equals to the preset special value of the first branch record which means that t' is on the start branch, the algorithm ends; Otherwise, assign the revert_to_time value of the current branch to t' and goes to (1) continue to execute the algorithm.

In the retrieved record result set, for multiple records with same source addresses src_block, take the record with the timestamp closest to the recovery time point t'. For records with the same source address src_block and timestamp, select the record with largest sequence number as the final result.

Finally, the I/O operation described in the final result set is overwritten to the initial state to obtain the data state at time t'.

Figure 3 shows the process of changing the protected host data volume (block 1∼block 4) from t0 to t6. For simplicity, we use <timestamp, source address, destination address>

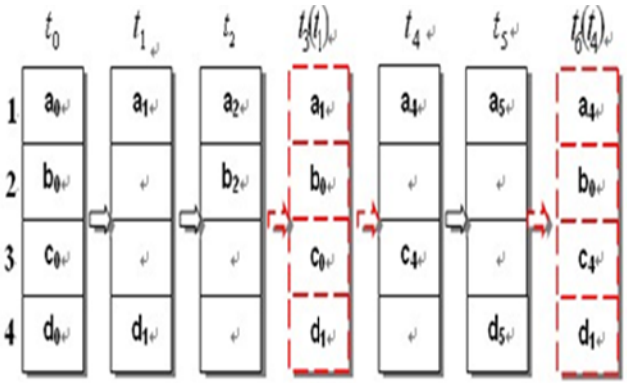

FIGURE 3. Protected data volume change process diagram.

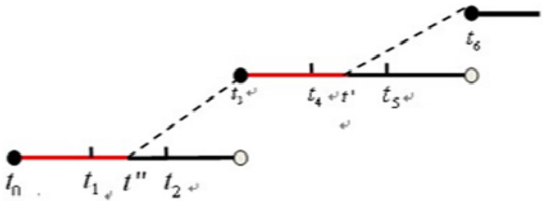

FIGURE 4. Protected data volume recovery branch diagram.

triplet to describe each bio operation (The size of block-level bio operations for a particular file system is same and fixed such as 4K or 1K). The triples corresponding to the bio operation at time t1 are <t1,1, a1> and <t1,4, d1>. Figure 4 corresponds to the recovery operation branch diagram on the data volume. As shown in Figure 4, a new branch is created by performing a recovery operation at times t3 and t6. The t3 and t6 moments are both the end time of the previous branch and the start time of the new branch. Assume the recovery point objective of the first branch (the branch with the starting time t0) is the preset value X. The following describes the process of restoring the protected host data block state from time t6 to time t' (t4<t'<t5) using the full recovery algorithm. According to the steps of full recovery algorithm, do step (1) to find the section where t' is located and get revert_to_time = t'', start_time = t3. Step (2) is executed to extract the metadata <t4,1, a4>, <t4, 3, c4> of the bio operation between [t3, t'). Step (3): Since revert_to_time is not equal to the preset special value X of the starting branch, go to step (1) to find the section where t' is located and get revert_to_time = X, start_time = t0. Extract the metadata <t0,1, a0>, < t0,2, b0>, < t0,3, c0>, <t0,4, d0>, <t1,1, a1> and <t1,4, d1> of the bio operation between [t0, t').

The protected data volumes are sequentially recovered in order of precedence. The recovery process is shown in figure 5. The data state (a4, b0, c4d1) after recovery is the state at time t' of the protected host data block.

① is the corresponding data of recovering <t0,1,a0>, <t0,2,b0>, <t0,3,c0> and <t0,4,d0>

② is the corresponding data of recovering <t1,1,a1>, <t1,4,d1>

③ is the corresponding data of recovering <t4,1,a4>, <t4,3,c4>

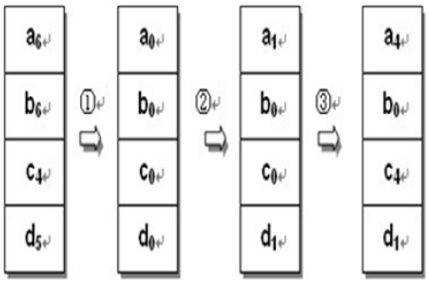

FIGURE 5. Process diagram of full recovery.

C. INCREMENTAL RECOVERY ALGORITHM

The incremental recovery algorithm first retrieves the metadata record set S1 that changes between RPO and current time point, then retrieves the metadata set S2 that varies from start time point to RPO. For each element a ∈ S1 (selecting the starting position member src_block of the record), if it belongs to set S2(a ∈ S2) the record with the largest time point in S2 is written in the final result set S; If a does not belong to S2, the content corresponding to the data block is obtained from the initial state and the result is written to the set S. Finally, I/O operations described by the metadata in set S are overwritten to the initial state. The algorithm is described as follows:

Let the initial time be t0, the user specifies the RPO as t', the recovery request time as Tcurrent to perform the following steps:

Find the time branch where t' is located, get the start time Tr_start and recovery point objective Tr_revert of the branch.

Find the time branch where Tcurrent is located, get the start time Tc_start and recovery point objective Tc_revert of the branch.

If Tc_start > Tr_start, add [Tc_start, Tcurrent) into time set Sinterval. Let Tcurrent = Tc_revert, then go to (2) to continue execution.

If Tc_start < Tr_start, add [Tr_start, t') into time set Sinterval. Let t'= Tr_revert, then go to (1) to continue execution.

If Tc_start = Tr_start and Tcurrent < t', add [Tcurrent,t') into time set Sinterval; If Tc_start = Tr_start but Tcurrent > t', add [t', Tcurrent) into time set Sinterval.

Retrieve all the metadata records which time points in set Sinterval to obtain set S1.

The metadata record set S2 whose start time is t0 and recovery point objective is t' is retrieved using full recovery algorithm.

For each element a (src_block) in S1, if a∈ S2, the record corresponding to src_block in S2 and having the largest time point is added into the result set S; Otherwise, take the content of initial state to construct record and add into result set S.

The time t' volume state is obtained by overwriting the I/O operations described in result set S to the state of current volume.

The final recovery result of the full recovery algorithm and the incremental recovery algorithm is same, but from descriptions of algorithm the full recovery algorithm overwrites the retrieved result set to initial state, but incremental recovery algorithm overwrites the retrieved result set to current state. Therefore, in order to reduce the recovery time, when the requested RPO is closer to starting time point, full recovery algorithm can be selected, but when the requested recovery point objective is closer to current time incremental recovery algorithm can be selected. Here we assume the amount of recovery data is proportional to the length of time. In practical applications, it is easy to determine which recovery algorithm is more appropriate for a certain time point according to characteristics of application. Of course, if the protected volume is physically damaged, only full recovery can be selected.

D. IMPLEMENTATION

In the implementation of prototype system, we have adopted three implementation methods: DIR_MySQL, OPT_MySQL based on the MySQL database and META_CDP. This section describes the specific implementation of each method. In next section, we analyze the actual test results of three implementation methods.

In DIR_MySQL method, a separate metadata record table and branch table are built for each client to store the captured metadata records and recovered branch records. All query and retrieval functions are implemented through SQL statements. For incremental recovery algorithm two create view operations are performed during retrieval process. OPT_MySQL method is optimized on the basis of DIR_MySQL, such as indexing on the built table, analyzing SQL query statements and optimizing. The database-based approach is simple to implement, but cannot achieve satisfactory performance.

Databases are generally designed for general purpose, so their performance is not ideal in certain application environments. The META_CDP method is designed in conjunction with specific application features of CDP metadata management. In META_CDP, metadata records and branch information are sequentially stored in metadata files and index files. Both metadata and branches are generated in chronological order, and their contents remain unchanged during the CDP protection time window. Each metadata record corresponds to a bio operation and branch information corresponds to a recovery operation, so the amount of data of the former will be much larger than the latter. In order to speed up the retrieval of metadata, we create a time-based index on the metadata file. Each index information corresponds to a certain number of metadata records and the index information is also stored in chronological order. Index records are dynamically created during the process of storing metadata. Both metadata records and index records are cached and only written to the file when reach a certain amount in cache. The structure of the metadata file and its index file is shown in Figure 6.

The structure of the index record is:

```
typedef struct {
    char start_time[TIME_LEN];
    off_t file_pos;
    uint32_t count
}
```

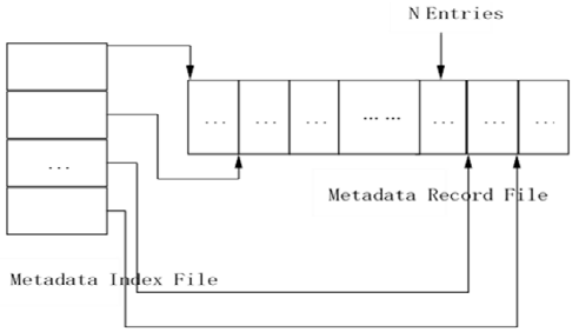

FIGURE 6. Metadata file and index file structure.

Start_time is the time of the first metadata record of the indexed metadata set (the earliest metadata record); File_pos is the position where the first metadata item of the index is stored in the metadata file; count is the number of metadata for the index.

After the required time branch segment is retrieved by full or incremental algorithm, META_CDP algorithm sequentially searches each index record. If the current index record and the next index record are not in any of the retrieved time branches, then override the index entry and check the next index entry; If their time all within the branch time segment, all metadata records of the current index record are added to the result set; If the time of current index record is within a certain time branch and the next index record is not in any time branch, each metadata record indexed by the current index is sequentially checked. If the time is within a certain time branch, this metadata is added to the result set. Repeat this step until all index entries or time branch have been checked. Because each index corresponds to a certain number of metadata records, crossing an index record means that multiple metadata records need not be compared; If both the current index and next index time point are within the time branch, all metadata records of the current index all added to the result set without comparison; So, the actual comparison times of metadata records are greatly reduced, the efficiency is improved.

V. EXPERIMENTAL RESULTS AND ANALYSIS

In this section we analyze the performance of these three management methods and test the efficiency of two different recovery algorithms of META_CDP algorithm.

A. TEST ENVIRONMENT

The server runs RedHat9 (the kernel is 2.6.18), the database is MySQL5.0.77, and the client runs on the virtual FC8 (the kernel is 2.6.23). Create a logical volume of 1G in FC8 system and configure to protect it when the system starts running. During the test, files are continuously written to the logical volume and a recovery request is made to restore the data to a previous state.

B. THE RELATIONSHIP BETWEEN TIME CONSUMED AND THE RECOVERY DATA AMOUNT

In order to test the relationship between the consumed and the recovery data amount for different methods, we tested the time taken given the recovery data amount for different methods. The relationship between the given data amount and the number of metadata records is determined by the file system type on the volume (the file system determines the size of each bio operation). Assuming the total amount of recovered data is M and the block size of the file system is b, then the number of metadata that the retrieval algorithm needs to process is approximately: M/b. Figure 7 shows the relationship between the retrieval time and the amount of recovered data for various recovery algorithms.

As can be seen from the figure 7, DIR_MySQL has the worst performance in full recovery algorithm followed by OPT_MySQL, and META_CDP has the best performance. Optimizing the database can improve performance to a certain extent, but the table linking algorithm inside the database is still complicated and the performance degradation trend is more obvious as the amount of data increases. The retrieval performance of META_CDP is much higher than that of database retrieval and the performance only decreases linearly as the data amount increases. The performance of incremental algorithm is almost worse than the full algorithm, because the incremental retrieval includes the full retrieval part (known by the algorithm description of 2.2.3) and the incremental algorithm using the algorithm of sorting before merging when calculates the intersection of S1 and S2. In incremental algorithm, performance of META_CDP is significantly better than DIR_MySQL and OPT_MySQL because the incremental retrieval process of database needs to create two views and with the data amount increases performance degradation of database retrieval is more dramatic than META_CDP. Although incremental algorithms spend more time on retrieving metadata than full algorithms in general, the data amount needs to be transferred to client for actual recovery after retrieval is much less than that of full recovery. Therefore, from the recovery point of view, the total recovery time is much less and the larger the volume protected the more obvious the advantage of increment.

C. SELECTION OF INCREMENTAL AND FULL RECOVERY

From the algorithm descriptions of incremental recovery and full recovery, it is known that full and incremental recovery are all related to the required recovery point objective. Full recovery selects metadata from the start time point to the recovery time point, while incremental recovery retrieves metadata from the recovery time point to the current time point. Thus, for different recovery point objective, performance of full recovery and incremental recovery is related to the amount of retrieved metadata. In this section we will quantitatively test the recovery efficiencies of the full and incremental recovery algorithms for different recovery time points. We use our own test program to periodically write blocks of 4K size to protected logical volumes for half an hour and record the start and end time of the program run, then select different recovery time point for recovery. These recovery time points are 20%, 40%, 50%, 60% and 80% from the starting time point to the whole period respectively.

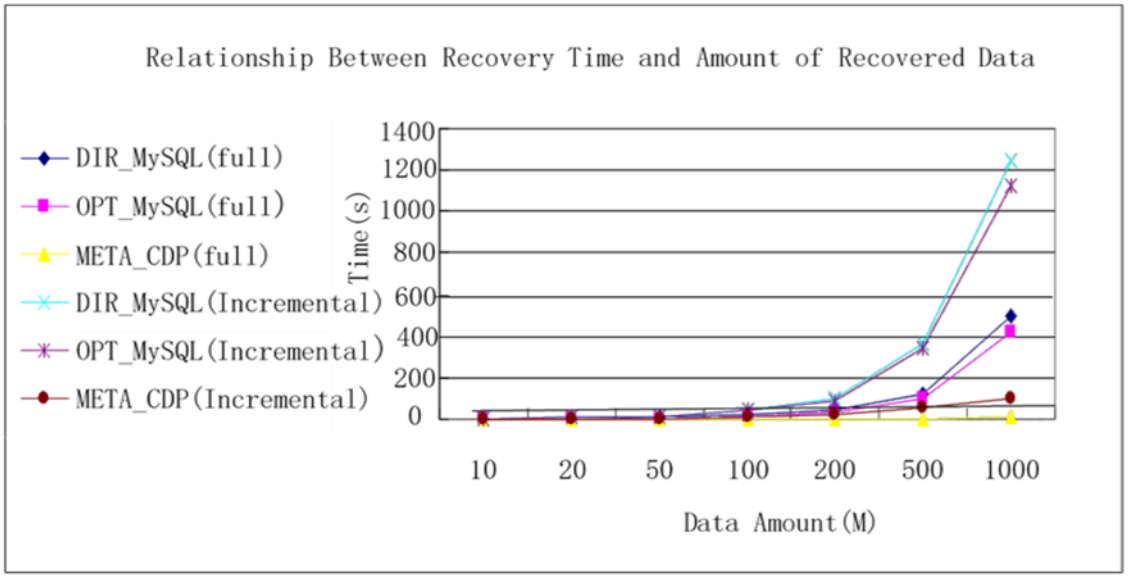

FIGURE 7. Relationship between recovery time and amount of recovered data.

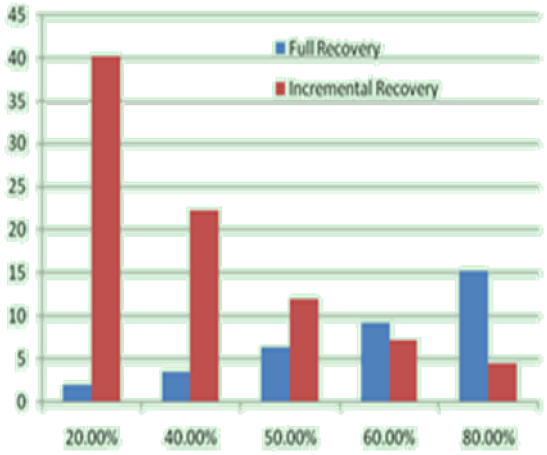

FIGURE 8. Test results of full and incremental recovery.

Figure 8 shows the test results. In this figure the abscissa indicates the time point from the start time point and the ordinate indicates the time taken for recovery in seconds. As can be seen from the figure when the recovery time point is closer to the starting time point, the less time is required for full recovery and the more time is required for incremental recovery; On the contrary, when the recovery time point is closer to the Time Point of Request for Recovery, the less time is required for incremental recovery and the more time is required for full recovery. So, in order to achieve fast recovery, the corresponding recovery algorithm can be selected according to the recovery time point. In general, full recovery can be selected when the recovery time point is close to the start time point and incremental recovery can be selected when the recovery time is close to the current time point.

VI. CONCLUSION

In this paper, we describe the design and implementation of several metadata management methods in block-based CDP systems for cyber security and analyze and compare their performance. From the experiments results, the performance of the general-purpose database is much lower than that of the specific application environment. The performance of metadata management method META_CDP is acceptable in recovery and is one of the keys to the implementation of entire CDP system. In addition, we make a quantitative test and analysis of the two recovery algorithms of META_CDP and give the general principle of choosing algorithms to achieve fast recovery efficiency. That is, when the recovery time point is close to the starting time point, full recovery can be used; When the recovery time point is close to the current time point, incremental recovery can be used. As the importance of data becomes more prominent the potential danger of data increases and the inherent needs of business continuity, CDP and corresponding rapid recovery guarantee mechanisms will play an important role in the field of data storage for cyber security.

REFERENCES

[1] M. Rock and P. Poresky, "Shorten your backup window," *Storage, Special Issue on Managing the Information that Drives the Enterprise*. Sep. 2005, pp. 28–34.

[2] A. L. Chervenak, V. Vellanki, and Z. Kurmas, "Protecting file systems: A survey of backup techniques," in *Proc. Joint NASA IEEE Mass Storage Conf.*, Mar. 1998, pp. 17–31.

[3] G. Duzy, "Match snaps to apps," in *Storage, Special Issue on Managing the Information that Drives the Enterprise*. Sep. 2005, pp. 46–52.

[4] D. Patterson, "A new focus for a new century: Availability and maintainability performance," in *FAST Keynote*. Jan. 2002. [Online]. Available: www.cs.berkeley.edu/patterson/talks/keynote.html

[5] K. Keeton, C. Santos, D. Beyer, J. Chase, and J. Wilkesm, "Designing for disasters," in *Proc. 3rd USENIX Conf. File Storage Technol.*, 2004, p. 5.

[6] J. Damoulakis, "Continuous protection," *Storage*, no. 4, pp. 33–39, Jun. 2004.
[7] X. Yu, Y. Tan, C. Zhang, C. Liang, A. Khaled, J. Zheng, and Q. Zhang, "A high-performance hierarchical snapshot scheme for hybrid storage systems," *Chin. J. Electron.*, vol. 27, no. 1, pp. 76–85, 2018.
[8] G. Laden, P. Ta-Shma, E. Yaffe, M. Factor, and S. Fienblit, "Architectures for controller based CDP," in *Proc. 5th USENIX Conf. File Storage Technol.*, San Jose, CA, USA, 2007, p. 21.
[9] N. Zhu and T.-C. Chiueh, "Portable and efficient continuous data protection for network file servers," in *Proc. 37th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2007, pp. 687–697.
[10] Q. Yang, W. Xiao, and J. Ren, "Trap-array: A disk array architecture providing timely recovery to any point-in-time," in *Proc. 33rd Annu. Int. Symp. Comput. Archit.*, 2006, pp. 289–301.
[11] X. Li, C. Xie, and Q. Yang, "Optimal implementation of continuous data protection (CDP) in Linux Kernel," in *Proc. Int. Conf. Netw. Archit. Storage*, Jun. 2008, pp. 28–35.
[12] M. Lu and T. Chiueh, "File versioning for block-level continuous data protection," in *Proc. 29th IEEE Int. Conf. Distrib. Comput. Syst.*, Jun. 2009, pp. 327–334.
[13] Z. Peterson and R. Burns, "Ext3cow: A time-shifting file system for regulatory compliance," *ACM Trans. Storage*, vol. 1, no. 2, pp. 190–212, 2005.
[14] P. Huang, K. Zhou, and C. Wu, "ShiftFlash: Make flash-based storage more resilient and robust," *Perform. Eval.*, vol. 68, no. 11, pp. 1193–1206, 2011.
[15] X. Yu, Y.-A. Tan, Z. Sun, J. Liu, C. Liang, and Q. Zhan, "A fault-tolerant and energy-efficient continuous data protection system," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 8, pp. 2945–2954, 2018.
[16] X. Yang, N. Jing, J. Wu, and J. Li, "File version based continuous data protection on distributed object," in *Proc. 7th Int. Conf. Comput. Eng. Netw.*, 2017, p. 90.
[17] A. A. Natanzon, "Storage array snapshots for logged access replication in continuous data protection system," Tech. Rep., 2016.
[18] N. Mandagere, R. Routray, Y. Song, and D. Du, "Cloud object storage based Continuous Data Protection(cCDP)," in *Proc. IEEE Int. Conf. Netw. Archit. Storage (NAS)*, Aug. 2015, pp. 244–254.
[19] J. J. Wu, Y. Cheng, S. Z. Mei, J. C. Ren, Z. Y. Wang, J. Zhou, and J. Ma, "SSR-CDPS: A novel continuous data protection system supporting service recovery," *Adv. Mater. Res.*, vol. 601, pp. 318–324, 2012.
[20] J. Yang, Q. Cao, X. Li, C. Xie, and Q. Yang, "ST-CDP: Snapshots in TRAP for continuous data protection," *IEEE Trans. Comput.*, vol. 61, no. 6, pp. 753–766, Jun. 2012.
[21] Z. Guan, J. Li, L. Wu, Y. Zhang, J. Wu, and X. Du, "Achieving efficient and secure data acquisition for cloud-supported Internet of Things in smart grid," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1934–1944, Dec. 2017.
[22] J. Zheng, Y. Tan, Q. Zhang, X. Zhang, L. Zhu, and Q. Zhang, "Cross-cluster asymmetric group key agreement for wireless sensor networks," *Sci. China Inf. Sci.*, vol. 61, no. 4, 2018, Art. no. 048103.
[23] Z. Ruan, X. Sun, and W. Liang, "Securing sensor data storage and query based on k-out-of- n coding," *Int. J. Commun. Syst.*, vol. 26, no. 5, pp. 549–566, 2013.
[24] D.-K. Kang, F. Alhazemi, S.-H. Kim, and C.-H. Youn, "A study of resource management for fault-tolerant and energy efficient cloud datacenter," in *Proc. Int. Conf. Cloud Comput.*, 2015, pp. 22–29.
[25] T. Wang, G. Zhang, X. Yang, and A. Vajdi, "A trusted and energy efficient approach for cluster-based wireless sensor networks," *Int. J. Distrib. Sensor Netw.*, vol. 12, no. 4, 2016, Art. no. 3815834.
[26] H. Peng, S. Si, M. K. Awad, N. Cheng, H. Zhou, X. Shen, and H. Zhao, "Energy-efficient and fault-tolerant evolution models for large-scale wireless sensor networks: A complex networks-based approach," in *Proc. IEEE Global Commun. Conf.*, Dec. 2016, pp. 1–6.
[27] P. Ladd and G. Sarosi, "Methods and apparatus for event logging in an information network," Tech. Rep., 2017.
[28] W. U. Hao, L. Xiaojie, and L. Peng, "TRAP-4 based continuous data protection system," *J. Comput. Appl.*, vol. 34, no. 1, pp. 54–57, 2014.
[29] D. Song, J. Sim, K. Park, and M. Song, "A privacy-preserving continuous location monitoring system for location-based services," *Int. J. Distrib. Sensor Netw.*, vol. 2015, no. 2, pp. 1–10, 2015.
[30] Y. Sheng, D. Wang, J. He, and D. Ju, "TH-CDP: An efficient block level continuous data protection system," in *Proc. IEEE Int. Conf. Netw. Archit. Storage*, Jul. 2009, pp. 395–404.
[31] J. Chen, J. Jia, Y. Deng, X. Wang, and A.-H. Aghvami, "Adaptive compressive sensing and data recovery for periodical monitoring wireless sensor networks," *Sensors*, vol. 18, no. 10, p. 3369, 2018.
[32] S. W. Choi, E. Kwon, Y. Kim, K. Hong, and H. S. Park, "A practical data recovery technique for long-term strain monitoring of mega columns during construction," *Sensors*, vol. 13, no. 8, pp. 10931–10943, 2013.
[33] X. Xu, X. Gao, J. Wan, and N. Xiong, "Trust index based fault tolerant multiple event localization algorithm for WSNs," *Sensors*, vol. 11, no. 7, pp. 6555–6574, 2011.
[34] Y. Zeng, C. J. Sreenan, N. Xiong, L. T. Yang, and J. H. Park, "Connectivity and coverage maintenance in wireless sensor networks," *J. Supercomput.*, vol. 52, no. 1, pp. 23–46, 2010.
[35] Z. Wang, T. Li, N. Xiong, and Y. Pan, "A novel dynamic network data replication scheme based on historical access record and proactive deletion," *J. Supercomput.*, vol. 62, no. 1, pp. 227–250, 2012.
[36] W. Guo, N. Xiong, A. V. Vasilakos, G. Chen, and C. Yu, "Distributed k-connected fault-tolerant topology control algorithms with PSO in future autonomic sensor systems," *Int. J. Sensor Netw.*, vol. 12, no. 1, pp. 53–62, 2012.
[37] W. Guo, N. Xiong, A. V. Vasilakos, G. Chen, and H. Cheng, "Multi-source temporal data aggregation in wireless sensor networks," *Wireless Pers. Commun.*, vol. 56, no. 3, pp. 359–370, 2011.
[38] C. Lin, N. Xiong, J. H. Park, and T.-H. Kim, "Dynamic power management in new architecture of wireless sensor networks," *Int. J. Commun. Syst.*, vol. 22, no. 6, pp. 671–693, Jun. 2009.

HONGYAN LI received the master's degree in computer science from the Central China Normal University, China, in 2005, and the Ph.D. degree in computer architecture from the Huazhong University of Science and Technology, China, in 2016. Since 2017 until now, she has been a Postdoctoral Researcher with the State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University. She is currently an Associate Professor with the Hubei University of Economics. Her research interests include computer vision, big data, and machine learning.

FENGJUN XIAO received the B.S. degree in economics from Beihang University, in 2009, and the master's degree in technology policy, in 2014 under the supervision of Prof. S. Li. He has been a Doctoral degree under the supervision of Prof. C. Li and began his research on the network security and emergency management, since 2015.

NAIXUE XIONG received the two Ph.D. degrees in information science from the Wuhan University and Japan Advanced Institute of Science and Technology, respectively. He is currently a Research Scientist with the Department of Computer Science, Georgia State University, USA. His research interests include communication protocols, network architecture and design, network technologies, and dependable computing, distributed, and parallel systems. Until now, Dr. Xiong published many research articles (including about 35 international journal articles). Some of his works were published or submitted in IEEE or ACM Transactions, JSAC, and IEEE INFOCOM. He has been a Program Chair, a General Chair, a Publicity Chair, a PC Member, and an OC Member of about 53 international conferences, and was invited to serve as a Reviewer for about 33 international journals. He is serving as an Associate Editor, an Editorial Board Member, and a Guest Editor for about nine international journals.