DOI 10.2298/CSIS110413071W

# S-TRAP: Optimization and Evaluation of Timely Recovery to Any Point-in-time (TRAP)

Chao Wang[1], Zhanhuai Li[1], Na Hu[1] and Yanming Nie[1]

[1] School of Computer Science, Northwestern Polytechnical University
710072 Xi'an, China
{wch, huna601, yanming}@mail.nwpu.edu.cn, lizhh@nwpu.edu.cn

**Abstract.** This paper presents S-TRAP, a novel block-level CDP (Continuous Data Protection) recovery mechanism based on TRAP. In accordance with a certain time interval $L$, S-TRAP breaks down the parity chain of TRAP and generates new sub-chains. Besides, S-TRAP introduces previous block cache which can reduce the negative impact on the primary storage system. Both mathematical analysis and experimental evaluation demonstrate that S-TRAP not only has the advantage of high recovery efficiency and reliability, but also further reduces the parity storage usage. Even more important, S-TRAP reduces the negative impacts on primary storage system performance to a large extent.

**Keywords:** data protection, snapshot, CDP, data backup, data recovery.

## 1. Introduction and Related Work

With explosive growth of networked information services and e-commerce, data protection has become one of the major issues for business organizations, government institutions and individuals [17]. Due to the close coupling between information service and the maturity of storage technology, failures such as hardware/software defects, human errors, virus attacks, and power outage can cause data damage or data loss. Recent work [11, 20] has demonstrated that the loss caused by data unavailability or data damage can be up to millions of dollars per hour. In order to protect data from possible failures and to recover data in case of failures, data protection technology is very necessary [26].

In disaster recovery theory, RPO (Recovery Point Objective) and RTO (Recovery Time Objective) [4, 11] are two key criteria to evaluate data protection mechanisms. RPO measures the data loss by time while RTO reflects the time duration spent on data recovery. Depending on the different RPOs, we summarize existing data protection mechanisms in three categories as shown in Fig. 1.

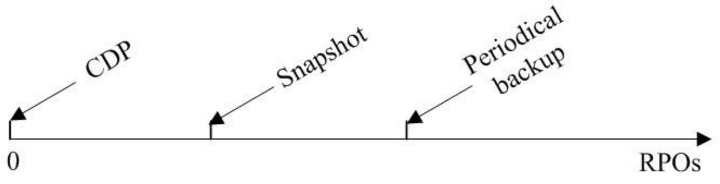


**Fig. 1.** Data Protection mechanisms for different RPOs

**Periodical backup.** Traditional periodical backup [2, 22, 27] has been widely adopted in data protection systems. Typically, backups are done on a daily, weekly or monthly basis. Since the backups can only be done in an offline manner, there may be exist the problems of data unavailability during the backup procedure. Besides, a huge amount of storage has to be occupied to keep the backups, so data compression technologies are often used to save the backup storage usage. As a result, periodical backup consumes too much time and storage space and also degrades system performance.

**Snapshot.** A snapshot is a point-in-time image of a collection of data which allows online backup. Generally, there are two common snapshot mechanisms: COW (Copy-On-Write) [1, 21, 23, 32, 33] and ROW (Redirect-On-Write) [6, 19, 30]. In order to save storage space, COW snapshot obtains and maintains the previous images of a data block upon the first write operation to that data block. Different from COW, ROW snapshot redirects all the write operations to the snapshot storage. Snapshots can be integrated in disk arrays [14], volume managers, file systems [3, 8, 9, 12, 21, 28, 29] and backup systems. Compared with periodical backup, snapshot can be created online, use less storage space, and has less negative impact to application performance. But it still cannot achieve timely recovery to any point in time.

**Continuous data protection (CDP).** CDP can provide timely recovery to any point in time at block level. Traditional CDP mechanism [4, 13, 18] keeps a log of changed data for each data block in timestamp order. Performance overhead and recovery efficiency are the important parameters to measure CDP system, much work [5, 16, 25, 35] has been done. Mariner [16] is a block-level CDP system that is designed to minimize the performance overhead associated with block update logging. Zhu and Chiueh [35] proposed a portable and efficient user-level CDP system, which incurs minimal performance overhead. TH-CDP [25] mainly focuses on general purpose, easy recovery and fast check pointing for fast recovery. Rather than per block basis schemes, GSP_BCDP [5] provides faster recovery by per volume basis.

But on the other hand, the huge amount storage space, which is required to keep log, limits the application and development of CDP. Some research efforts, such as Clotho [7], are made to reduce the storage space usage of traditional CDP to increase adoption of data de-duplication technologies. However, these methods cannot effectively solve log space usage problem of traditional CDP. TRAP [31, 33, 34] is a novel CDP mechanism, by compressing and saving the XOR parity among changed data blocks along the time dimension, it can improve space efficiency significantly. But its recovery efficiency and reliability is low; what's more, it has a great negative impact on storage system performance. ST-CDP [15], a TRAP-based optimal implementation, improves the recovery efficiency and reliability by adding snapshots between parity chains, but still has great impact on storage system performance.

To overcome the shortcomings of existing continuous data protection mechanisms, in this paper we presented a novel block level CDP architecture based on TRAP, named S-TRAP. In accordance with certain time interval $L$, S-TRAP breaks down the parity chain of TRAP and generates new sub-chains. In addition, the Previous Block Cache which contains partial images of data blocks at the time point *T-1* is leveraged to reduce the negative impact on primary storage system. In comparison with ST-CDP, S-TRAP not only has the advantage of fast recovery efficiency and high reliability, but also further reduces the parity storage occupation. Even more important, S-TRAP reduces the negative impact on primary storage system performance to a large extent. Formal analysis and preliminary experiment result show that S-TRAP have low RTO and high reliability, while retaining lower storage overhead and less infection on primary storage system.

The rest of this paper is organized as follows: Section 2 gives a brief overview of the TRAP architecture. Section 3 presents our optimal CDP mechanism S-TRAP in detail. In Section 4, we use mathematical model to guide our design in terms of six aspects, including recovery time, parity storage space usage, reliability, impact on system performance, recovery direction and optimal $L$. Section 5 describes the experiment settings. Numerical result and discussions are presented in Section 6. We conclude the paper in Section 7.

## 2. Brief Overview of TRAP

Instead of keeping all versions of a data block as being modified by write operations, TRAP only keeps a parity log of each write on the parity storage. Fig. 2 shows the basic design of TRAP. Suppose that at time point $t$, a write operation is submitted to block $B$ which updates its content from $B_{t-1}$ to $B_t$. TRAP generates the new parity of block $B$ as follows:

$$P_t = B_{t-1} \oplus B_t \tag{1}$$

where $\oplus$ is a bitwise XOR operator. And then TRAP appends the new parity $P_t$ to the parity chain $(P_1, P_2, \cdots, P_{t-1}, P_t)$.

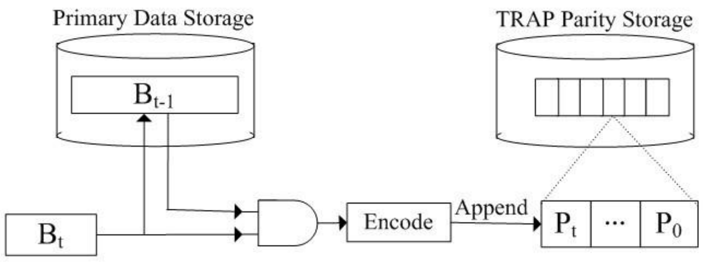


**Fig. 2.** Basic design of TRAP architecture

Extensive experiments [33] have demonstrated a very strong content locality that only 5%-20% of bits inside a data block are actually changed by a write operation. Because of the characteristics of bitwise XOR operation, the parity can reflect the exact changes made by the new write operation at the bit level, i.e., except a very small portion of nonzero bits, most bits in the new parity are zeros. Consequently the parities can be compressed easily with a high compression ratio. Therefore, the storage space required to keep track of write operations can be greatly reduced by orders of magnitude.

TRAP architecture provides a double-way data recovery capability, either forward or backward, referred to as redo and undo, respectively. Consider the parity chain $(P_1, P_2, \cdots, P_{t-1}, P_t)$ corresponding to data block $B$. Suppose we have both the initial and latest images of block $B$, referred to as $B_0$ and $B_t$. If we need to recovery block $B$ to the previous image of time point $r\ (0 < r < t)$, TRAP performs the following process:

$$B_r = B_0 \oplus P_1 \oplus P_2 \oplus \cdots \oplus P_r \tag{2}$$

where $B_r$ denotes the data image of block $B$ at time point $r$.

The above process represents a typical forward/redo recovery. Similarly, to recover from the opposite direction, TRAP performs backward/undo recovery according to the following process:

$$B_r = P_{r+1} \oplus P_{r+2} \oplus \cdots \oplus P_t \oplus B_t \tag{3}$$

**Definition 1** For any recovery time point $r$, the recovery chain of block $B$ is $(P_1, P_2, \cdots, P_{r-1}, P_r)$ with length $r$, or $(P_{r+1}, P_{r+2}, \cdots, P_{t-1}, P_t)$ with length $t-r$.

We can infer from Equation (2) and (3) that any damaged parity in the recovery chain will lead to the failure of the whole recovery process.

TRAP has a negative impact on primary storage system performance. This is because when generating parity, TRAP has to firstly acquire the previous data image from primary storage system which can definitely increases the I/O response time. Although TRAP can benefit from the parity generation component of RAID (RAID 4, 5, etc.), it also has some limitations: (a) only the RAID with XOR parity can be chosen; (b) it requires implementing TRAP in the RAID controller; (c) both primary storage and parity storage have to be under the same RAID controller. In addition, we can also infer from Equation (2) and (3) that the recovery time becomes longer as the parity chain gets longer, and the recovery efficiency decreases as the length of the recovery chain increases. The invalid parity results in the invalidity of the corresponding recovery chains, obviously, the failure probability is larger in a longer recovery chain. Therefore, the reliability of TRAP also decreases with the growth of recovery chain length increases. All these restrictions can limit the application of TRAP.

ST-CDP is an optimal implementation of continuous data protection in Linux kernel based on TRAP. By adding snapshots between parity chains, ST-CDP can improve the recovery efficiency and reliability, but the snapshots increase the parity storage usage as well. Furthermore, ST-CDP also has to obtain the previous data image from primary storage; therefore, high I/O response problem still exists.

## 3. Design of S-TRAP Mechanism

### 3.1. Parity recording

In accordance with a certain time interval $L$, S-TRAP divides the parity chain of TRAP and generates new sub-chains. Consecutive $L$ parities from a sub-chain are illustrated in Fig. 3. Different form ST-CDP, S-TRAP does not add snapshots between parity chains. Instead, we use a special generation method for the FIRST parity of each sub-chain. Consider data block $B$, the first parity of each sub-chain is generated as:

$$P_{kL+1}{}' = B_0 \oplus B_{kL+1} \tag{4}$$

while the generating method of the other parities is consistent with TRAP, as shown in Equation (1).

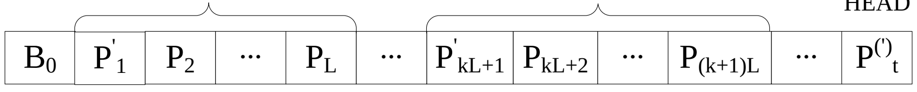


**Fig. 3.** Parity chain of S-TRAP

**Definition 2** For any data block $B$, $\left(P'_{kL+1}, P_{kL+2}, \cdots, P_{(k+1)L}\right)$ is the $k-th$ sub-chain.

Different from TRAP, the idea of our approach breaks down parity chain to sub-chains. Since all parities of write operations are kept on backup storage with the form of SUB-chains in order to provide Timely Recovery to Any Point in time, we name our approach as S-TRAP.

## 3.2. Previous Block Cache

Caching is one of the major technologies used to improve I/O performance in disk array. However, if TRAP wants to takes the advantage of the cache in disk array to reduce the negative impact on primary storage system, it would require: (a) implementing TRAP architecture in the disk array controller; (b) both primary storage and parity storage to be under the same disk array controller. In order to eliminate these restrictions, S-TRAP manages a separate cache to improve its I/O efficiency, which is called Previous Block Cache (PBC).

Suppose that at time point $t$, a write operation is submitted to block $B$, S-TRAP firstly checks in PBC. If there is a cache hit, S-TRAP:

(a) commits the new data image of block $B$ directly to primary storage system,

(b) generates parity by previous data image of block $B$ in Cache, and commits it to parity storage system, and

(c) replaces the new image of block $B$ into Cache according to replacement algorithm;

Otherwise, it:

(a) gets the previous image of block $B$ from primary storage system,

(b) commits the new data image of block $B$ directly to primary storage system,

(c) generates parity by previous data image of block $B$ in Cache, and commits it to parity storage system, and

(d) replaces the new image of block $B$ into Cache according to replacement algorithm.

Upon cache hitting, S-TRAP saves one additional I/O operation. Therefore, PBC can reduce the negative impact on primary storage system.

## 3.3. Data Recovery

Without loss of generality, consider the sub-chain $\left(\mathrm{P'}_{\mathrm{kL+1}}, P_{kL+2}, \cdots, P_{(k+1)L}\right)$ which corresponds to data block $B$. In order to restore it to the previous recovery time point $r = kL + i \; (1 \le i \le L)$, S-TRAP performs the following process:

$$B_r = B_{kL+i} = B_0 \oplus P'_{kL+1} \oplus P_{kL+2} \oplus \cdots \oplus P_{kL+i} \quad (5)$$

Due to the particularity of the first parity in each sub-chain, we do analysis under two conditions:

(a) $r = kL$, recovery time point $r$ is the time point which associates with the first parity in the $k-th$ sub-chain. In this case, S-TRAP only needs to perform $B_r = B_0 \oplus P'_{kL+1}$ to do a forward/redo recovery. The length of recovery chain $\left(P'_{kL+1}\right)$ is 1 and XOR operation only needs to be performed once. So, recovery efficiency is high, and

(b) $r = kL + i,\ 0 < i \le L$, this is the most general case, and recovery time point *r* falls on the time point which associates with other parities of the $k-th$ sub-chain. S-TRAP performs forward/redo recovery based on Equation (5). The length of recovery chain $\left(P'_{kL+1}, P_{kL+2}, P_{kL+3}, \cdots, P_{kL+i}\right)$ is $i$, so XOR operation needs to be performed $i$ times.

## 4. Formal Analysis of S-TRAP

In this session, we mathematically model the S-TRAP mechanisms in detail, including recovery time, parity storage space usage, reliability, and its impacts on system performance and recovery direction. In addition, we calculate the key optimal $L$ value, which has a major impact on parity storage space usage and recovery time. In order to facilitate the description, we define the following notations as shown in Table 1.

**Table 1.** Definition of Symbols

| Symbol | Definition |
|---|---|
| $L$ | The length of the sub-chain |
| $IO_{rate}$ | I/O throughput of storage system |
| $B_{size}$ | Size of the data block (in MB) |
| $C$ | Compression ratio of parity $P$ |
| $C'$ | Compression ratio of the special parity $P'$ |
| $T_{dec}$ | Decoding time of parity |
| $T_{XOR}$ | XOR operation time |

$C'$ is the compression ratio of the special parity. The special parity reflects the exact changes of many write operations. It has less zero bits, therefore $C'$ is smaller than $C$. To simplify the discussion, we assume that $C'$ is a constant. In addition, because of the similarity of two recovery directions of TRAP (i.e., forward/redo and backward/undo), we only select the forward/redo recovery in the following discussion.

### 4.1. Recovery Time

Theorem 1 For any recovery time point r, with using S-TRAP mechanism, the maximum length of recovery chain is $L$, and the expected length of recovery length is $(L+1)/2$.

**Proof.** For any recovery time point $r$, without loss of generality, it certainly falls on the time interval associated with one of the sub-chains $\left(\mathrm{P'}_{\mathrm{kL+1}}, P_{kL+2}, \cdots, P_{(k+1)L}\right)$, and it is uniformly distributed among $kL+1$ and $(k+1)L$.

(a) The worst case is that the recovery time point $r$ is the time point which is associated with the last parity of sub-chain $k$, i.e., $r=(k+1)L$. Therefore, the maximum length of recovery chain is $L$.

(b) Because recovery time $r$ is uniformly distributed on a closed interval $\left[kL+1, (k+1)L\right]$, the expected length of recovery length is:

$$E\left(L_r\right)=\sum_{k=1}^{L} k \times P\left\{x=k\right\}=\frac{L+1}{2} \tag{6}$$

**Theorem 2** For any recovery time point $r$, the recovery time of TRAP is proportional to $r$, and the recovery time of S-TRAP is proportional to $L$.

Proof.

(a) The recovery time of TRAP is:

$$T_{TRAP}=\frac{B_{size}}{IO_{rate}}+\left(T_{dec}+T_{xor}+\frac{B_{size}}{C \cdot IO_{rate}}\right) \cdot r+\frac{B_{size}}{IO_{rate}} \tag{7}$$

where the first item is the time of getting and transferring the initial image $B_0$, the second is the time of decoding, and the last is the time of transferring and recording the final image. It can be deducted from equation (7) that the recovery time of TRAP is proportional to recovery time point $r$.

(b) The recovery time of S-TRAP is:

$$T_{S-TRAP}=\frac{B_{size}}{IO_{rate}}+\left(T_{dec}+\frac{B_{size}}{C' \cdot IO_{rate}}\right)+\left(T_{dec}+T_{xor}+\frac{B_{size}}{C \cdot IO_{rate}}\right) \cdot E\left(L_r\right)+\frac{B_{siz}}{IO_{ra}} \tag{8}$$

where the first item is the time of getting and transferring the initial image $B_0$, the second is the decoding time of the first parity in sub-chain, the third is the decoding time of other parities, and the last is the time of transferring and recording the final image. We substitute Equation (6) to Equation (8), so we can draw a conclusion that the recovery time of S-TRAP is proportional to $L$ and independent of the recovery time point $r$.

In the S-TRAP mechanism, the expected number of parities which is involved in XOR operation is $\left(L+1\right)/2$ ($L$ for the worst case); therefore, the recovery time of S-TRAP always fluctuates within a small range which is defined by $L$ value and Equation (8). As a result, S-TRAP has high and stable recovery efficiency.

The recovery time of both ST-CDP and S-TRAP is independent of RPO, but is dependent on d and L. The recovery time is similar between two mechanisms, which consist of 1) getting and transferring snapshot, and 2) decoding and transferring the final image. When the d value of ST-CDP equals to the L value of S-TRAP, both mechanisms have the same numbers of XOR operations. Thus, the recovery time of two mechanisms is approximately the same.

## 4.2. Parity Storage Space Usage

Theorem 3 Compared with TRAP, the additional parity storage space usage ratio of S-TRAP is inversely proportional to $L$.

**Proof.** Without loss of generality, consider data block $B$ at the latest time point $t$,

The parity storage space of TRAP is:

$$S_{TRAP} = \frac{B_{size} \cdot t}{C} \tag{9}$$

The parity storage space of S-TRAP is:

$$\begin{aligned} S_{S-TRAP} &= B_{size} \cdot \frac{1}{C'} \cdot \left\lceil \frac{t}{L} \right\rceil + B_{size} \cdot \frac{1}{C} \cdot \left( t - \left\lceil \frac{t}{L} \right\rceil \right) \\ &\approx \frac{B_{size}}{C'} \cdot \frac{t}{L} + \frac{\mathrm{B}_{size}}{C} \cdot \left( t - \frac{t}{L} \right) \end{aligned} \tag{10}$$

where the first item is the storage space usage of the first parities in sub-chains, the second item is the storage space usage of the other parities.

Additional storage space usage is:

$$\Delta S = \frac{B_{size} \cdot t}{L} \cdot \left( \frac{1}{C'} - \frac{1}{C} \right) \tag{11}$$

Additional parity storage space usage ratio is:

$$\frac{\Delta S}{S_{TRAP}} = \frac{1}{L} \cdot \left( \frac{C}{C'} - 1 \right) \tag{12}$$

Therefore, it can be drawn from Equation (12) that the additional parity storage space usage ratio of S-TRAP is inversely proportional to $L$.

As mentioned above, S-TRAP neither adds any snapshots between parity chains, nor increases the total count of parities. When the snapshot interval d of ST-CDP equals to the sub-chain length L of S-TRAP, the parity storage usage of S-TRAP is smaller than that of ST-CDP.

**Theorem 4** When d equals to L, the storage usage of S-TRAP is smaller than ST-CDP.

The parity storage space of ST-CDP is:

$$S_{ST-CDP} = B_{size} \cdot \left\lceil \frac{t}{d} \right\rceil + \frac{B_{size}}{C} \cdot t \tag{13}$$

Delta parity storage usage between ST-CDP and S-TRAP is:

$$\Delta S = B_{size} \cdot \left\lceil \frac{t}{d} \right\rceil \cdot \left(1 - \frac{1}{C'}\right) + \frac{B_{size}}{C} \cdot \left\lceil \frac{t}{d} \right\rceil \tag{14}$$

where $C'(C' \geq 1)$ is the compression ratio of special parity, thus, $\Delta S > 0$.

Therefore, the storage usage of S-TRAP is smaller than that of ST-CDP when d equals to L.

## 4.3. Reliability

Due to hardware and/or software failures, disasters or outages, both the initial image and parities are likely to be damaged. It is obvious that invalid parity results in the invalidity of the corresponding recovery chains and the invalid recovery chain can make us unable to recover data to previous time points. Therefore, improving the reliability of recovery chain is extremely important for CDP.

In S-TRAP mechanism, the valid probability of recovery chain only correlates to $L$, and would not decrease as time point grows; while the valid probability of recovery chain of TRAP decreases with the growth of recovery chain length.

**Definition 3** The invalid probability of initial image and parity is $p$.

**Theorem 5** For any recovery time point $r$, with using S-TRAP mechanism, the valid probability of recovery chain is $(1-p)^{L+1}$ in the worst case, and the mathematical expectation of the valid probability of recovery chain is $\frac{1}{pL} \cdot \left[(1-p)^2 - (1-p)^{L+2}\right]$.

**Proof.** For any recovery time point $r$, without loss of generality, it certainly falls on the time interval which is associated with one of the sub-chains $\left(\mathrm{P'}_{\mathrm{kL+1}}, P_{kL+2}, \cdots, P_{(k+1)L}\right)$, and it is uniformly distributed among $kL+1$ and $(k+1)L$.

(a) The worst case is that we have the recovery time point $r = (k+1)L$, and the length of recovery chain reaches maximum value $L$. In order to ensure the validity of the recovery chain, both the initial image and all parities in the sub-chain must be valid. Therefore, the valid probability of recovery chain is minimized in such case:

$$\mathrm{P}_{S-TRAP} = (1-p)^{L+1} \tag{15}$$

(b) Because recovery time $r$ is uniformly distributed on closed interval $\left[kL+1,(k+1)L\right]$, the expected valid probability of recovery chain is:

$$E\left(\mathrm{P}_{S-TRAP}\right) = \frac{1}{L}\cdot\sum_{k=1}^{L}(1-p)^{k+1} = \frac{1}{pL}\cdot\left[(1-p)^{2}-(1-p)^{L+2}\right] \tag{16}$$

**Theorem 6** For any recovery time point $r$, with using TRAP mechanism, the valid probability of recovery chain is decreasing with the increase of time point $r$.

**Proof.** For any recovery time point $r$, with using TRAP mechanism, all the parities from time point $1$ to $r$, and the initial image should be valid to ensure the validity of recovery chain. So, the valid probability of recovery chain is:

$$\mathrm{P}_{TRAP} = (1-p)^{r+1} \tag{17}$$

According to Equation (17), the valid probability of recovery chain for TRAP mechanism is decreasing with the increase of time point $r$.

Based on the above analysis, compared with TRAP mechanism, the valid probability of recovery chain in S-TRAP mechanism is only correlated to $L$, and would not decrease with the growth of time point. Therefore, S-TRAP can provide much higher reliability than TRAP can.

S-TRAP does not insert snapshot between parity chains as ST-CDP, so initial image is required for every recovery operation. Therefore, this causes a single point of failure.

## 4.4. Impact on system performance

For each write operation:

(a) both TRAP and ST-CDP have to get the image of previous time points from primary storage system first, and thereafter the new image can be committed to parity storage; while

(b) S-TRAP firstly checks whether the block cache is hit or not. If there is a hit, it gets the image of previous time point from PBC directly, and then one I/O operation is saved for primary storage system. Cache hit makes committing the new image to primary storage faster, thus reduced the IO response time of primary storage system. If there is a miss, S-TRAP has to get the image from primary storage as well, and then puts the image into PBC according to certain replacement algorithm.

Therefore, for each write operation, both TRAP and ST-CDP need one additional disk I/O on primary storage, while S-TRAP depends on whether cache is hit or not. Increasing the size of PBC can improve the hit ratio, thus, S-TRAP saves more I/O operations for primary storage system accordingly. Meanwhile, S-TRAP has a higher memory footprint than TRAP and ST-CDP do.

## 4.5. Recovery Direction

In S-TRAP mechanism, the special parity in each sub-chain is generated by bitwise XOR operation between initial and the latest image of data block. Due to the special generation method, S-TRAP could only recover in one direction (i.e., forward/redo). Compared with the double-way recovery of TRAP and ST-CDP, S-TRAP cannot recover data when the initial image is damaged. However, except for the initial image stored in primary storage, S-TRAP also keeps an addition backup in parity storage. This lowers the probability of an invalid initial image. Moreover, S-TRAP can also escalate to double-way recovery easily by adding an additional normal parity at the first time point of each sub-chain.

## 4.6. Optimal L value

Recovery time and parity storage space usage are the two key factors to measure the recovery cost of CDP mechanism. Next, we will come out with a definition for the recovery cost function of S-TRAP.

**Definition 4** The recovery cost of S-TRAP is the product of its recovery time and parity storage space usage:

$$\begin{aligned} G(L) &= T_{S-TRAP}(L) \cdot S_{S-TRAP}(L) \\ &= (A_1 + A_2 L) \cdot \left( A_3 + \frac{A_4}{L} \right) \end{aligned} \tag{18}$$

where

$$A_1 = \frac{2 \cdot B_{size}}{IO_{rate}} + \left( T_{dec} + T_{xor} + \frac{B_{size}}{C' \cdot IO_{rate}} \right) - \frac{1}{2} \left( T_{dec} + T_{xor} + \frac{B_{size}}{C \cdot IO_{rate}} \right)$$

$$A_2 = \frac{1}{2} \left( T_{dec} + T_{xor} + \frac{B_{size}}{C \cdot IO_{rate}} \right);$$

$$A_3 = \frac{B_{size} \cdot t}{C};$$

$$A_4 = B_{size} \cdot t \cdot \left( \frac{1}{C'} - \frac{1}{C} \right)$$

The minimum value of $G(L)$ exists when $L = L_0 = \sqrt{\frac{A_1 A_4}{A_2 A_3}}$. Since $L$ is an integer, we choose the integer closest to $L_0$ as optimal $L$ value.

## 5. Experiment Setup

In order to evaluate the S-TRAP mechanism, we designed and implemented a prototyped system. Fig. 4 shows the architecture of this prototype which integrates such three mechanisms, as TRAP, ST-CDP and S-TRAP. The prototype is a block device driver layered below file system, database system or other applications in the Linux kernel and therefore it works transparently to upper-level applications. The prototype captures write requests from upper levels and all the mechanisms keep all parities of each block at any time point according to the description in previous sections; In addition, all the mechanisms use the same open-source compression library, i.e., ***zlib***. Meanwhile, all the mechanisms are implemented independently of RAID controller. According to the discussion of ST-CDP [15], we set the d value of ST-CDP to 85. In addition, we chose five different block sizes: 4KB, 8KB, 16KB, 32KB and 64KB. Experimental configurations are listed in Table 2.

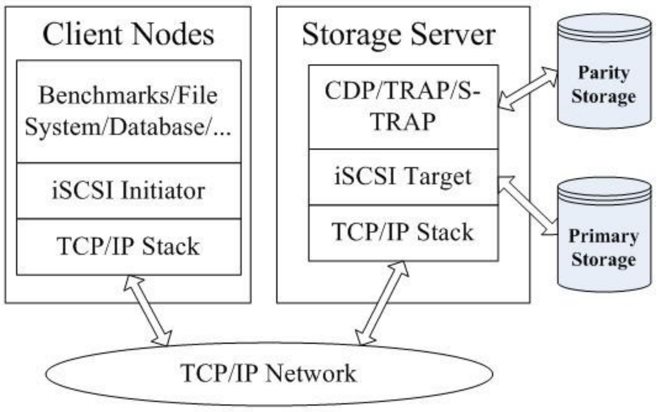


**Fig. 4.** System architecture of prototype

In addition, we have to determine the storage structure of parities on parity storage volume. Generally, there are two types of storage structure, timestamp-ordered and block address-ordered. Keeping parities in timestamp order does not need pre-allocating storage space for each data block. But it has such disadvantages as: (1) need to search previous parity when generating new parity, I/O costly; and (2) need to traverse parity log repeatedly when recovering data, once for each data block, so the recovery algorithm complexity is high. For block address order, it is easy to recover because all the parities of each data block are kept together, so the recovery algorithm complexity is low. But here an efficiency parity space management algorithm is needed. For the purpose of getting an accurate evaluation, we choose block address-ordered to keep parities in the prototype for both TRAP and S-TRAP.

**Table 2.** Experiment Environments

| Configuration | Client Nodes and Storage Server |
|---|---|
| CPU | AMD Opteron 850, 2.4GHz, 64bit |
| Memory | 8GB ECC DDR RAM |
| Disk | 73G SCSI Disk |
| OS | Red Hat Enterprise Server with kernel 2.6.18 |
| Switch | Asus GigaX1124, Gigabit |
| NIC | 1 Gbps PCI Ethernet Adapter |

Appropriate workloads are important for performance studies [10]. We use popular benchmarks in our experiments, which are TPC-C and IoMeter. TPC-C is a well-known benchmark used to model the operation terminal of businesses where real-time transactions are processed. We choose one of the TPC-C implementations developed by the Hammerora Project [24] for Oracle database. According to the TPC-C specification, data tables for five warehouses with 25 users were built in order to issue transactional workloads to Oracle database. IoMeter is a flexible and configurable file system benchmark. In our experiments, we choose the mode of 30% writes and 70% reads for measuring the performance of file system and block device.

## 6. Results and Discussions

### 6.1. Impacts on system performance

In all the three mechanisms, the computation of bitwise XOR and decompression of parities may introduce additional overhead to primary storage system, and such overhead may negatively impact application performance accordingly. For the purpose of quantifying such impact, we measured the computation time of XOR and decompression as shown in Table 3. The block size refers to the decompression unit which is the size of parity before compression. It can be seen that both XOR and decompression computations are only take tens to hundreds of microseconds. Compared with the computation time of disk I/O, these times can be neglected in most cases.

**Table 3.** Measured Computation Time

| Block Size (KB) | XOR Time (ms) | Decompress Time (ms) |
|---|---|---|
| 4 | 0.025132 | 0.070960 |
| 8 | 0.050406 | 0.130094 |
| 16 | 0.101033 | 0.220106 |
| 32 | 0.211721 | 0.374015 |
| 64 | 0.420478 | 0.613474 |

Except for the slight impact caused by computation of bitwise XOR and decompression for parities, parity recording and reading, which needs disk I/O operation, have even more negative impact on primary storage system. In order to see quantitatively how much the impact is, we carried out the following experiments.

In S-TRAP mechanism, PBC is important to mitigate the negative impact on primary storage system. In order to find the suitable size of PBC, we measure the average I/O response time for different PBC size. With block size 4KB and 64KB, we run IoMeter benchmark (70% reads and 30% writes) for one hour on S-TRAP configured with different PBC size. As shown in Fig. 5, the average I/O response time decreases with PBC size increases, and becomes stable after 128MB. So we choose 128MB as the PBC size in the following experiments.

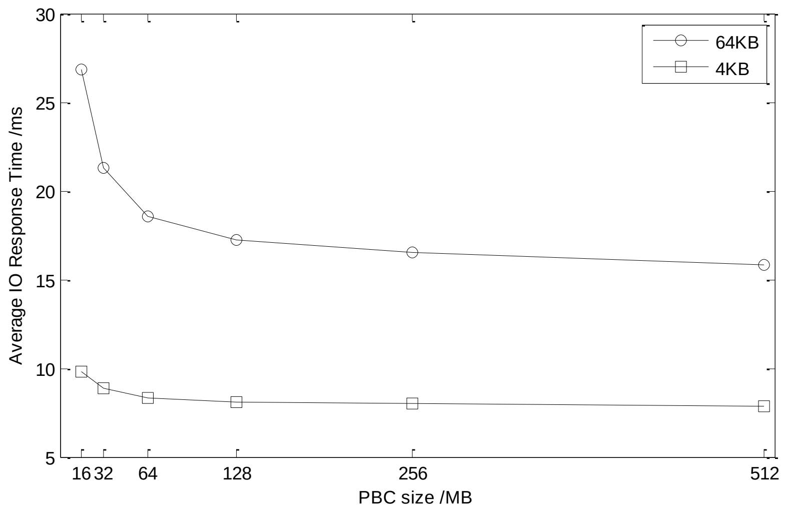


**Fig. 5.** Average IO Response Time over PBC size

With five different block sizes, we run IoMeter benchmark (70% reads and 30% writes) for one hour on TRAP, ST-CDP and S-TRAP, respectively. The results of original primary storage system with no data protection mechanism are used as a base line to show the negative impacts of the three CDP mechanisms. Fig. 6 shows the measured results of average I/O response time for 70% writes and 30% reads of IoMeter benchmark.

As shown in Fig. 6, we observed in our experiments that all the three CDP mechanisms have negative performance impacts; TRAP has the most while S-TRAP the least. For block size 64KB, compared with original storage system, TRAP's average I/O response time is about 66.07% longer, 70.18% and 31.48% for ST-CDP and S-TRAP, respectively. It can be seen that S-TRAP reduces the negative impact on primary storage system about 52.35% compared with TRAP.

The reason is that, with the introduction of Previous Block Cache, the number of additional I/O reduced, and the impact on primary system reduced accordingly. After neglecting the slight impact caused by computation of bitwise XOR and decompress, TRAP and ST-CDP still have one additional I/O to get the image of pervious time point when recording parities. Thus, S-TRAP has less impact.

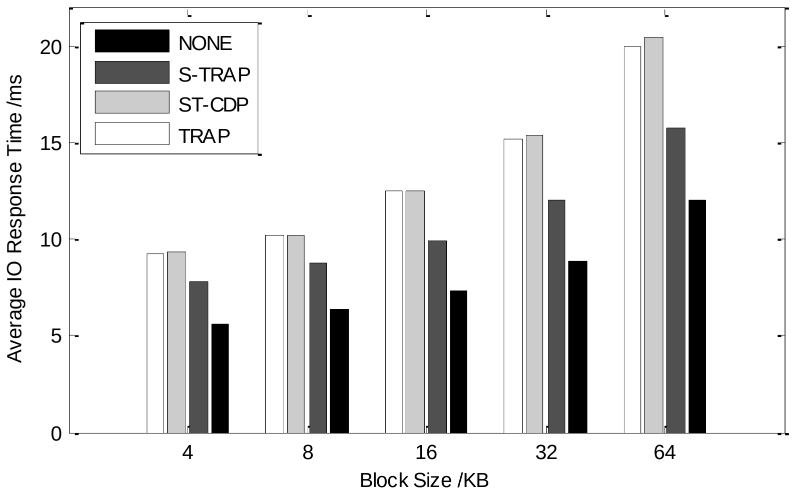


**Fig. 6.** Average I/O response time comparison

### 6.2. Sub-chain Length L

In S-TRAP mechanism, both recovery time and parity storage space usage heavily depend on the length of sub-chain length. According to the definition of recovery cost in Def. 4, through the trade-off between recovery time and parity space usage, recovery cost obtains the minimum when $L = L_0$ . In order to find the appropriate $L_0$, we run TPC-C benchmark for one hour on S-TRAP configured with different sub-chain length $L$ while the block size is 16KB, and then we perform recoveries for different configurations. The measured recovery time and parity storage usage are shown in Figs. 7 and 8. The recovery cost under Def. 4 is shown in Fig 9.

As shown in Figs. 7 and 8, with the increasing of sub-chain length $L$, the parity storage usage decreases while the recovery time grows. In Fig. 9, we observed that the recovery cost obtains the minimum when sub-chain is 90. So we choose 90 as the sub-chain length in the following experiments.

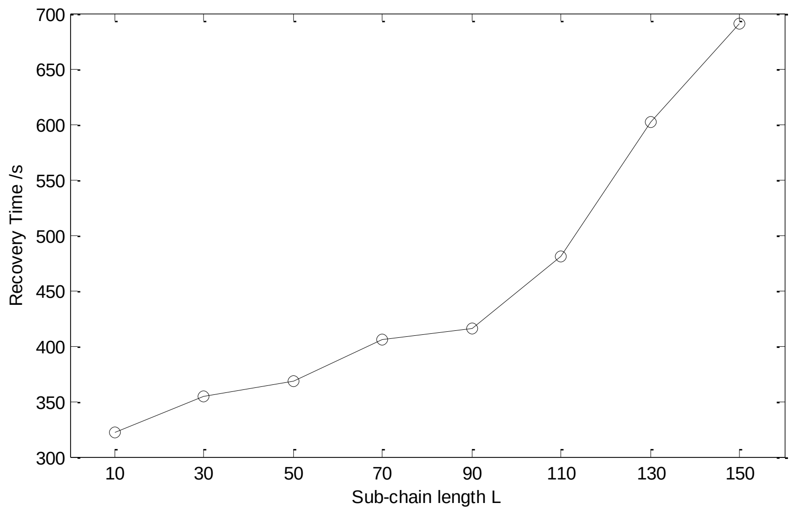


**Fig. 7.** Recovery time of S-TRAP configured with different sub-chain length

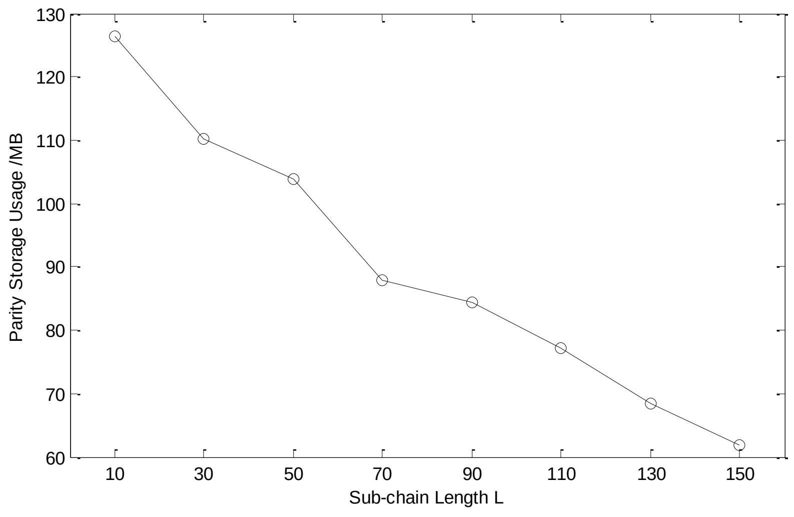


**Fig. 8.** Parity storage usage of S-TRAP configured with different sub-chain length

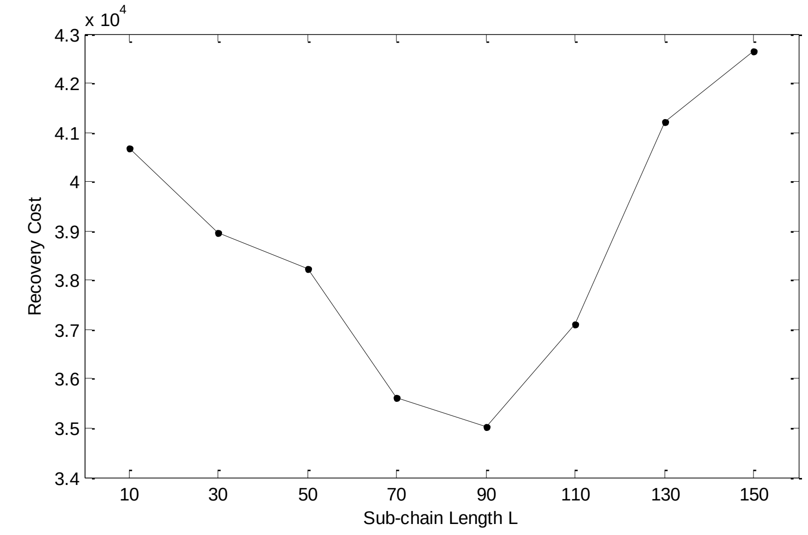


**Fig. 9.** Recovery cost of S-TRAP configured with different sub-chain length

## 6.3. Parity storage space usage

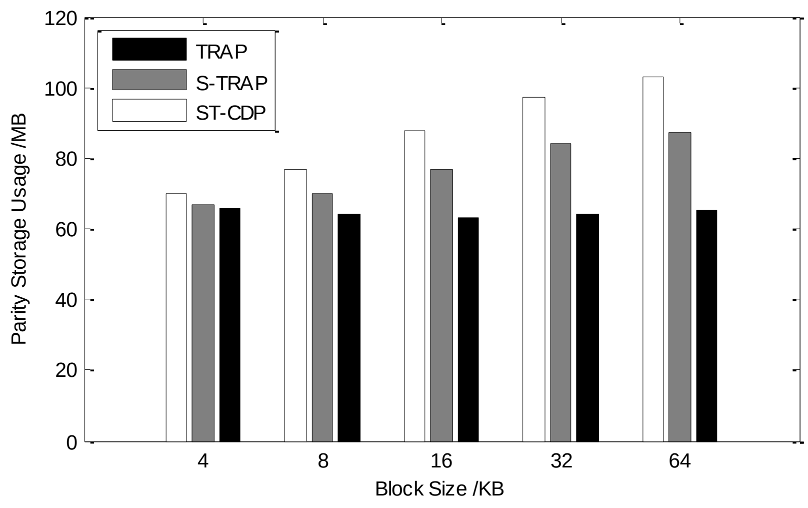
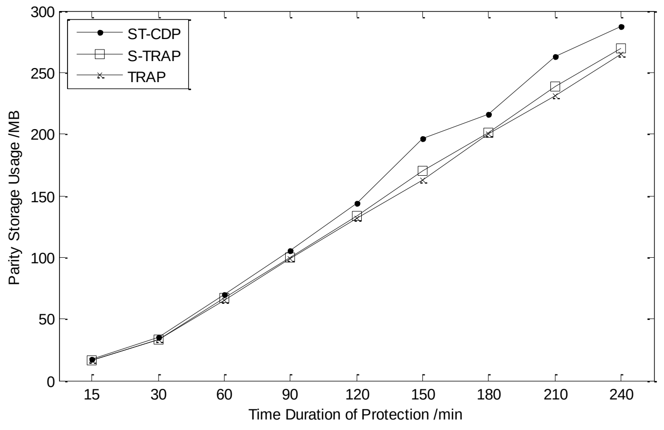


**Fig. 10.** Parity storage space usage comparison

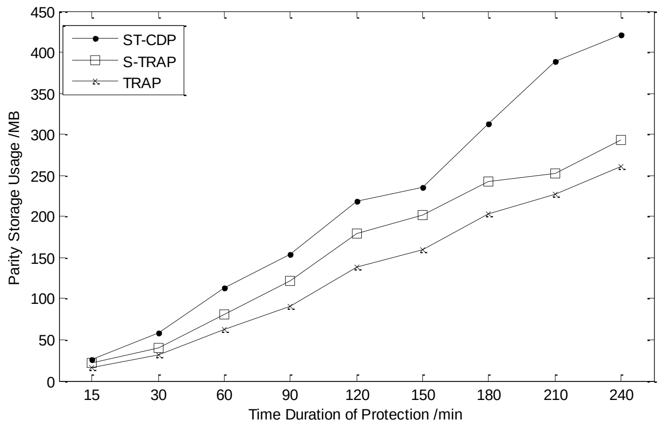
This experiment is to measure the amount of parity storage space usage of different data protection mechanisms. With five different block sizes, we run TPC-C benchmark for one hour on TRAP, ST-CDP and S-TRAP, respectively. Then, we measured the parity storage space usage for different mechanisms. Fig. 10 shows the results of parity storage space usage comparison for TPC-C on the oracle database.

In Fig. 11, we also measure the space usage of TRAP, ST-CDP and S-TRAP for different protecting time durations with block size 4KB and 64KB.

It is shown in Fig. 10 that TRAP requires the least parity storage space while the ST-CDP the most. For example, for the block size of 16KB, the space usage of ST-CDP increases by 22.22% compared with TRAP, while S-TRAP is only 9.52%. We can get similar results from Fig. 11.



(a)



(b)

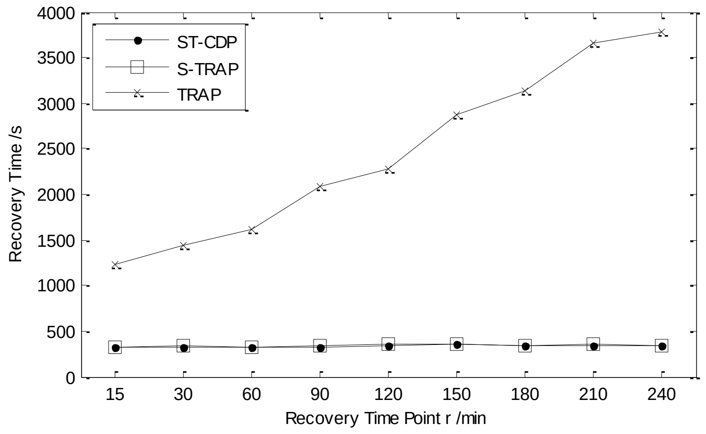
**Fig. 11.** Parity storage space usage over time duration: (a) Block Size=4KB; (b) Block Size=64KB

The reason is that S-TRAP performs the same parity algorithm as TRAP in most cases, only performs the special parity algorithm once for each $L$ time points. It is noteworthy that S-TRAP does not increase the total number of parities. On the other hand, ST-CDP supplely adds a snapshot for each d parities. Compared with parities with high compression ratio, snapshot takes up more storage space.

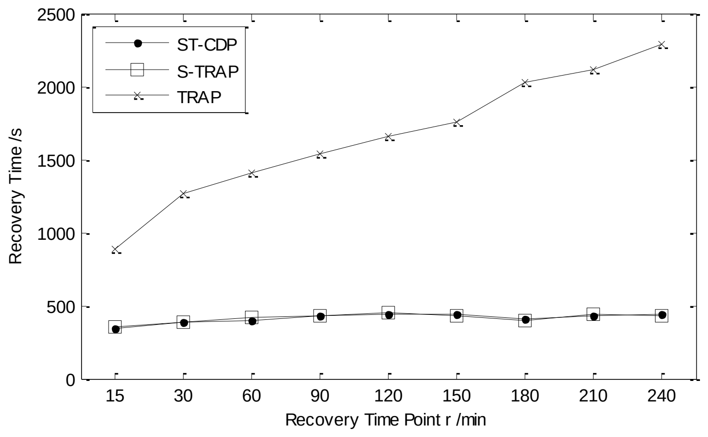
## 6.4. Recovery time

RTO (Recovery Time Objective) is a key indicator to evaluate a data protection mechanism, and it reflects the recovery duration of time after a disaster. In order to compare the recovery time among TRAP, ST-CDP and S-TRAP, we carried out this experiment. With block sizes of 4KB and 64KB, we run TPC-C benchmark for a sufficiently long time, more than five hours, on TRAP, ST-CDP and S-TRAP, separately. As the benchmark runs, the parity storage was filled with sufficient parities. Then, we perform recoveries for different time points in the past. Fig. 12 shows the measured results of recovery time for TRAP, ST-CDP and S-TRAP.

In Fig. 12, TRAP's recovery time increases obviously as RPO increases; while the recovery time of ST-CDP and S-TRAP keeps flat while RPO changes. This also illustrates the related results of Theorem (2). In addition, ST-CDP and S-TRAP perform almost the same in recovery ability. Thus it can be seem that, under the premise of further reducing parity storage usage, S-TRAP still maintains high recovery efficiency. For example, for block size 4KB, TRAP takes 1440 seconds to recover data to 30 minutes ago, about 4.3 times longer than ST-CDP or S-TRAP; while TRAP takes 3790 seconds to recover data to 240 minutes ago, about 11.3 times longer than ST-CDP or S-TRAP. Consequently, ST-CDP and S-TRAP have faster and more stable recovery ability.



(a)



(b)

**Fig. 12.** Recovery time comparison between TRAP, S-TRAP and ST-CDP: (a) Block Size=4KB; (b) Block Size=64KB

## 7. Conclusions

In this paper, we present a novel block-level CDP architecture, referred to as S-TRAP. In accordance with a certain time interval $L$, S-TRAP breaks down the parity chain of TRAP and generates new sub-chains. Furthermore, S-TRAP introduces PBC which can reduce the negative impact on primary storage system. We use a simple mathematical model to guide our design in such six aspects as: recovery time, parity storage space usage, reliability, impact on system performance, recovery direction and optimal $L$. Extensive experiments have been carried out to evaluate S-TRAP. Both mathematical analysis and experimental evaluation have shown that S-TRAP not only has the advantage of high recovery efficiency and reliability, but also further reduces the parity storage usage. Even more important, S-TRAP reduces the negative impacts on primary storage system performance to a large extent.

For future work, we plan to further improve the reliability and recoverability of S-TRAP by ECC (Error Correcting Codes) in sub-chains. We also plan to design parity storage architecture for S-TRAP in order to improve its recording and recovering ability.

**Acknowledgment.** This research is supported by the National High Technology Research and Development Program (863 Program) of China under the Grant No. 2009AA01A404, and the National Natural Science Foundation of China (NSFC) under the Grant No. 61033007.

## References

1. Charles, B.M., Grunwald, D.: Peabody: The time travelling disk. In Proceedings of the 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies. San Diego, CA, USA, 241-253. (2003)
2. Chervenak, A., Vellanki, V., Kurmas, Z.: Protecting File Systems: A Survey of Backup Techniques. In Proceedings of the 6th NASA Goddard Conference on Mass Storage Systems and Technologies / 15th IEEE Symposium on Mass Storage Systems, College Park, MD, USA, 17-31. (1998)
3. Cox, L.P., Murray, C.D., Noble, B.D.: Pastiche: making backup cheap and easy. In Proceedings of the 5th Symposium on Operating Systems Design and Implementation, Boston, MA, USA, 285-298. (2002)
4. Damoulakis, J.: Continuous Protection. Storage, Vol. 3, No. 4, 33-39. (2004)
5. Deng, Y.B., Chen, S.G., Hu, W., Gao, F., Liu, C.Y.: A novel block-level continuous data protection system. In Proceedings of the 4th International Conference on New Trends in Information Science and Service Science, Gyeongju, Korea, 242-247. (2010)
6. Duzy, G.: Match Snaps to Apps. Storage, special issue on managing the information that drives the enterprise, 46-52. (2005)
7. Flouris, M.D., Bilas, A.: Clotho: Transparent Data Versioning at the Block I/O Level. In Proceedings of the 12th NASA Goddard / 21st IEEE Conference on Mass Storage Systems and Technologies, College Park, MD, USA, 101-114. (2004)
8. Gifford, D.K., Needham, R.M., Schroeder, M.D.: The Cedar File System. Communications of the ACM, Vol. 31, No. 3, 188-198. (1998)
9. Howard, J.H., Kazar, M.L., Menees, S.G., Nichols, D.A., Satyanarayanan, M., Sidebotham, R.N., West, M.J.: Scale and performance in a distributed file system. In Proceedings of the 11th ACM Symposium on Operating Systems Principles, New York, NY, USA, 1-2. (1987)
10. Hu, Y.M., Yang, Q.: DCD-disk caching disk: a new approach for boosting I/O performance. In Proceedings of the 23rd Annual International Conference on Computer Architecture, Philadelphia, PA, USA, 169-178. (1996)
11. Keeton, K., Santos, C., Beyer, D., Chase, J., Wilkes, J.: Designing for disasters. In Proceedings of the 3rd Usenix Conference on File and Storage Technologies, San Francisco, CA, USA, 59-72. (2004)
12. Kistler, J.J., Satyanarayanan, M.: Disconnected operation in the Coda File System. ACM Transactions on Computer Systems, Vol. 10, No. 1, 3-25. (1992)
13. Laden, G., Ta-Shma, P., Yaffe, E., Factor, M., Fienblit, S.: Architectures for controller based CDP. In Proceedings of the 5th Usenix Conference on File and Storage Technologies, San Jose, CA, USA, 107-121. (2007)
14. Lee, E.K., Thekkath, C.A.: Petal: distributed virtual disks. In Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems, Cambridge, MA, USA, 84-92. (1996)
15. Li, X., Xie, C.S., Yang, Q.: Optimal Implementation of Continuous Data Protection (CDP) in Linux Kernel. In Proceedings of the IEEE 2008 International Conference on Networking, Architecture, and Storage, Chongqing, China, 28-35. (2008)
16. Lu, M.H., Lin, S.B., Chiueh, T.: Efficient Logging and Replication Techniques for Comprehensive Data Protection. In Proceedings of the 24th IEEE Conference on Mass Storage Systems and Technologies, San Diego, CA, USA, 171-184. (2007)

17. McKnight, J., Asaro, T., Babineau, B.: Digital Archiving: End-User Survey and Market Forecast. The Enterprise Strategy Group (2006). [Online]. Available: http://www.enterprisestrategygroup.com/2006/03/digital-archiving-end-user-survey-market-forecast-2006-2010/ (current December 2011)
18. Morrey, C.B., III, Grunwald, D.: Peabody: the time travelling disk. In Proceedings of the 20th IEEE / 11th NASA Goddard Conference on Mass Storage Systems and Technologies, San Diego, CA, USA, 241-253. (2003)
19. Patterson H., Manley S., Federwisch M., Hitz D., Kleiman S., Owara S.: SnapMirror(R): file system based asynchronous mirroring for disaster recovery. In Proceedings of the 1st Usenix Conference on File and Storge Technologies, Monterey, CA, USA, 117-129. (2002)
20. Patterson, D.: Availability and Maintainability >> Performance: New Focus for a New Century. Keynote of the 1st Usenix Conference on File and Storge Technologies, Monterey, CA, USA (2002). [Online]. Available : http://www.cs.berkeley.edu/~pattrsn/talks/keynote.html (current December 2011)
21. Peterson, Z., Burns, R.: Ext3cow: a time-shifting file system for regulatory compliance. ACM Transactions on Storage, Vol. 1, No. 2, 190-212. (2005)
22. Rock, M., Poresky, P.: Shorten Your Backup Window. Storage, special issue on managing the information that drives the enterprise, 28-34. (2005)
23. Santry, D.S., Feeley, M.J., Hutchinson, N.C., Veitch, A.C., Carton, R.W., Ofir, J.: Deciding when to forget in the Elephant file system. In Proceedings of the 17th ACM symposium on Operating systems principles, New York, NY, USA, 110-123. (1999)
24. Shaw, S.: Hammerora: Load Testing Oracle Databases with Open Source Tools. Hammerora Project (2004). [Online]. Available: http://hammerora.sourceforge.net (current December 2011)
25. Sheng, Y.H., Wang, D.S., He, J.Y., Ju, D.P.: TH-CDP: An Efficient Block Level Continuous Data Protection System. In Proceedings of the 2009 IEEE International Conference on Networking, Architecture, and Storage, Zhangjiajie, China, 395-404. (2009)
26. The 451 Group: Total Recall: Challenges and Opportunities for the Data Protection Industry. The 451 Group (2006). [Online]. Available: http://www.the451group.com/reports/executive_summary.php?id=218 (current December 2011)
27. Wang, D., Xue, W., Shu, J.W., Shen, M.M.: Fault Tolerance with Virtual Disk Replicas in the Mass Storage Network. Journal of Computer Research and Development, Vol. 43, No. 10, 1849-1854. (2006)
28. Xiang, X.J., Shu J.W., Zheng, W.M.: An efficient fine granularity multi-version file system. Journal of Software, Vol. 20, No. 3, 754-765. (2009)
29. Xiang, X.J., Shu, J.W., Xue, W., Zheng, W.M.: Design and implementation of an efficient multi-version file system. In Proceedings of the 2007 International Conference on Networking, Architecture, and Storage, Guilin, China, 277-278. (2007)
30. Xiao, W.J., Liu, Y., Yang, Q., Ren, J., Xie, C.: Implementation and Performance Evaluation of Two Snapshot Methods on iSCSI Target Storages. In Proceedings of the 14th NASA Goddard / 23rd IEEE Conference on Mass Storage Systems and Technologies, College Park, MD, USA, 101-110. (2006)
31. Xiao, W.J., Yang, Q.: Can We Really Recover Data if Storage Subsystem Fails. In Proceedings of the 28th International Conference on Distributed Computing Systems, Beijing, China, 597-604. (2008)

32. Xiao, W.J., Yang, Q., Ren, J., Xie, C.S., Li, H.Y.: Design and Analysis of Block-Level Snapshots for Data Protection and Recovery. IEEE Transactions on Computers, Vol. 58, No. 12, 1615-1625. (2009)
33. Xiao, W.J., Ren, J., Yang, Q.: A Case for Continuous Data Protection at Block Level in Disk Array Storages. IEEE Transactions on Parallel and Distributed Systems, Vol. 20, No. 6, 898-911. (2009)
34. Yang, Q., Xiao, W.J., Ren, J.: TRAP-Array: A Disk Array Architecture Providing Timely Recovery to Any Point-in-time. In Proceedings of the 33rd International Symposium on Computer Architecture, Boston, MA, USA, 289-300. (2006)
35. Zhu, N.N., Chiueh, T.: Portable and Efficient Continuous Data Protection for Network File Servers. In Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Edinburgh, UK, 687-697. (2007)

**Chao Wang** is currently a PhD candidate at the School of Computer Science, Northwestern Polytechnical University, Xi'an, China. His research interests include disaster tolerant, massive storage system and performance evaluation.

**Zhanhuai Li** is a professor at the School of Computer, Northwestern Polytechnical University, Xi'an, China. He is vice chairman of Database Technical Committee, China Computer Federation. He is a senior member of China Computer Federation. He received his Master's and Ph.D. degrees from Northwestern Polytechnical University in 1987 and 1996 respectively. His current research interests include RFID data management and multimedia real-time database.

**Na Hu** is currently a Master candidate at the School of Computer Science, Northwestern Polytechnical University, Xi'an, China. Her research interests include data protection and performance evaluation.

**Yanming Nie** is currently a PhD candidate at the School of Computer Science, Northwestern Polytechnical University, Xi'an, China. His research interests include data stream processing, uncertain data management, RFID data management and software engineering and process.

*Received: April 13, 2011; Accepted: December 12, 2011.*