S2-RAID: A New RAID Architecture for Fast Data Recovery

Jiguang Wan[*], Jibin Wang[*], Qing Yang[+], and Changsheng Xie[*]
[*]Dept. of Computer Science, Huazhong University of Science and Technology, China
[+]Dept. of Electrical, Computer, and Biomedical Engineering
University of Rhode Island, Kingston, RI, 02881, USA
{wanjiguang, wangjibin}@gmail.com, qyang@ele.uri.edu, cs_xie@mail.hust.edu.cn

Abstract--*As disk volume grows rapidly with terabyte disk becoming a norm, RAID reconstruction time in case of a failure takes prohibitively long time. This paper presents a new RAID architecture, S^2-RAID, allowing the disk array to reconstruct very quickly in case of a disk failure. The idea is to form skewed sub RAIDs (S^2-RAID) in the RAID structure so that reconstruction can be done in parallel dramatically speeding up data reconstruction time and hence minimizing the chance of data loss. To make such parallel reconstruction conflict-free, each sub-RAID is formed by selecting one logic partition from each disk group with size being a prime number. We have implemented a prototype S^2-RAID system in Linux operating system for the purpose of evaluating its performance potential. SPC IO traces and standard benchmarks have been used to measure the performance of S^2-RAID as compared to existing baseline software RAID, MD. Experimental results show that our new S^2-RAID speeds up data reconstruction time by a factor of 3 to 6 compared to the traditional RAID. At the same time, S^2-RAID shows similar or better production performance than baseline RAID while online RAID reconstruction is in progress.*

1 INTRODUCTION

RAID is the de facto storage architecture [1] that has been widely used to store petabyte scale data as information keeps growing exponentially. In such large scale storage systems, disk failures will become daily events if not more frequent [2, 3]. Therefore, being able to quickly rebuild disk array in case of failure event has become critical to today's information services that are widespread covering every corner of our society. There are two key issues that make fast reconstruction of RAID upon failure essential. 1) Any additional failure during the reconstruction process will result in data loss. Hence this reconstruction time is often referred to as "window of vulnerability" [2] that should be as small as possible. 2) Data services are either stopped completely for offline RAID reconstruction or productivity is negatively impacted by online RAID reconstruction that interferes with production IOs.

While fast RAID rebuilding is important to minimize the "window of vulnerability", current technology trend adversely affects such reconstruction time. Disk volume continues to grow rapidly with terabytes disks becoming a norm whereas disk bandwidth and access time including seek time and rotation latency improve little. As a result, recovering terabytes of data on a failed disk of the traditional RAID architecture will take prohibitively long time increasing the chance of data loss. Such technology trend is likely to continue in the foreseeable future.

The requirement of fast RAID reconstruction coupled with the technology trend motivates us to seek for a new RAID architecture that allows high speed online RAID reconstruction and has as little negative performance impact as possible. This paper presents a new Skewed Sub-array RAID structure, S^2-RAID for short. The idea is to divide a large disk in the RAID into small partitions that together with partitions on other disks to form sub-arrays. These sub-arrays are skewed among the disks in the RAID in such a way that conflict-free parallelism is achieved during RAID reconstruction when any disk fails. Recovered data that were on the failed disk are stored in parallel on multiple disks consisting of spare disks and available space of good disks. The parallel reading and writing of S^2-RAID can substantially speedup the RAID rebuilding process and at the same time reduces negative performance impact of front end applications while RAID reconstruction is going on in background.

In order to validate our design concept, we have implemented a prototype software S^2-RAID on Linux OS at block device level based on the widely used software RAID, MD (multiple device). The prototype is installed inside an iSCSI target to provide data storage services to iSCSI initiators. Using the S^2-RAID prototype, we have carried out extensive experiments using SPC IO traces and standard IO benchmarks to measure its IO performance, reconstruction time, and online performance impact. Experimental results show that S^2-RAID improves RAID reconstruction speed of the baseline software RAID system by a factor of 3 to 6. The front end application performance while rebuilding RAID online using S^2-RAID stays the same or slightly better than the baseline software RAID depending on workloads.

The paper is organized as follows. Next section presents the design and data layout of S^2-RAID. Section 3 gives the implementation details on Linux system. Experimental settings and workload characteristics are presented in Section 4 followed by numerical results and discussions in Section 5. Related work is discussed in Section 6. Section 7 concludes the paper.

978-1-4244-7153-9/10/$26.00 ©2010 IEEE

2 S²-RAID DESIGN AND DATA LAYOUT

2.1 Sub Array Formation

Consider a traditional RAID5 storage system. When a disk had failed, data in the failed disk would have been rebuilt by reading a data stripe remained in good disks, performing an Exclusive-OR computation, and writing rebuilt data in a spare disk. This process continues until all data chunks in the failed disk are reconstructed. The reconstruction bandwidth is ultimately limited by the single data stream being rebuilt, one data chunk at a time. To rebuild over terabyte of data on a failed disk, it is clearly going to take very long time. Our objective is to speed up this rebuilding process by reading multiple data stripes, performing multiple Exclusive-OR computations, and writing rebuilt data chunks to multiple spare disks all in parallel. The multiple spare disks can be either physical spare disks or spare logic partitions available on good data disks.

In order to achieve our objective of parallel data reconstruction, we would like to be able to read, in parallel and conflict-free, all multiple data stripes that are in remaining good disks and are needed to rebuild multiple data chunks in the failed disk. For this purpose, we divide each one of total R disks in the RAID into K partitions. As a result of this partition, we have $K*R$ logic disk partitions. We then divide R physical disks into groups of size G resulting in $N=\lceil R/G \rceil$ physical disk groups, where G is a prime number. subRAIDs are then formed by picking up one disk partition from each one of N groups giving rise to subRAIDs of size N. Note that $1 < K$, $N <= G$, and G is a prime number. The key is how to form these subRAIDs and position these subRAIDs among the R physical disks in such a way that rebuilding data of any failed physical disk can be done in maximal parallelism. That is, parallel reading of multiple subRAIDs during data reconstruction is conflict-free.

Let notation $S.L$ represent logic disk number L in subRAID number S. Figure 1 shows an example of such subRAIDs formation with $R=9$ and $G=N=3$. This is a S²-RAID5 with each

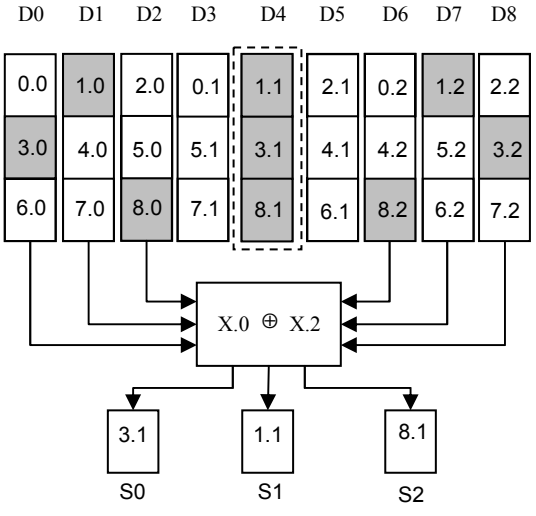

Figure 1. S²-RAID data layout for RAID 5.

subRAID's stripe size being 3 with 2 data chunks and one parity chunk. SubRAID 0 uses physical disks D0, D3, and D6 to store logic disks 0, 1, and 2, respectively. Similarly, subRAID 1 uses physical disks D1, D4, and D7 to store logic disks 0, 1, and 2, respectively, as shown in Figure 1. In this example, we have total of 9 subRAIDs each of which contains 3 logic disks. Such skewed placement of subRAIDs allows maximum parallelism of data reconstruction. Suppose physical disk D4 failed resulting in loss of logic partitions of 1.1, 3.1, and 8.1 as shown in shaded area of Figure 1. To rebuild the lost logic partitions, we would need 1.0 and 1.2 to rebuild 1.1, 3.0 and 3.2 to rebuild 3.1, and 8.0 and 8.2 to rebuild 8.1. The placement of the subRAIDs in Figure 1 assures that all the needed data stripes to rebuild the lost data chunks reside on different physical disks allowing parallel and conflict-free reconstruction. S0, S1, and S2 in Figure 1 are spare disks to store newly rebuilt data.

In general, consider a RAID system with R disks that are divided into N groups of size G each, Let $P_{i,j}$ be a G elements vector representing subRAID numbers of the $(j+1)^{th}$ partition on disks of $(i+1)^{th}$ group in the RAID. For example, in Figure 1, we divide 9 disks into 3 groups with 1st group consisting of D0, D1, and D2; 2nd group consisting of D3, D4, and D5; and 3rd group consisting of D6, D7, and D8. $P_{0,0} = (0,\ 1,\ 2)$ gives subRAID numbers of the first partitions on D0, D1, and D2 of first disk group. $P_{0,1} = (3,\ 4,\ 5)$ gives subRAID numbers of the second partitions of the same group of disks, 1st group. $P_{2,2} = (8,\ 6,\ 7)$ represents subRAID numbers of the third partitions of third disk group containing disks D6, D7, and D8, and so forth.

SubRAIDs are mapped to physical disks in S²-RAID using a mapping table defined by a matrix M, where $M = (m_0\ m_1, \ldots m_{N-1})$. Each sub matrix m_i in M is defined recursively by

$$m_0 = \begin{pmatrix} P_{0,0} \\ P_{0,1} \\ P_{0,2} \\ \ldots \\ P_{0,K-1} \end{pmatrix},$$

$$m_1 = \begin{pmatrix} P_{1,0} \\ P_{1,1} \\ P_{1,2} \\ \ldots \\ P_{1,K-1} \end{pmatrix} = \begin{pmatrix} SH_r^0(P_{0,0}) \\ SH_r^1(P_{0,1}) \\ SH_r^2(P_{0,2}) \\ \ldots \\ SH_r^{K-1}(P_{0,K-1}) \end{pmatrix},$$

$\ldots\ldots,$

$$m_i = \begin{pmatrix} P_{i,0} \\ P_{i,1} \\ P_{i,2} \\ \ldots \\ P_{i,K-1} \end{pmatrix} = \begin{pmatrix} SH_r^0(P_{i-1,0}) \\ SH_r^1(P_{i-1,1}) \\ SH_r^2(P_{i-1,2}) \\ \ldots \\ SH_r^{K-1}(P_{i-1,K-1}) \end{pmatrix},$$

$\ldots\ldots$ and

$$\boldsymbol{m}_{N-1} = \begin{pmatrix} \boldsymbol{SH_r^0(P_{N-2,0})} \\ \boldsymbol{SH_r^1(P_{N-2,1})} \\ \boldsymbol{SH_r^2(P_{N-2,2})} \\ \cdots \\ \boldsymbol{SH_r^{K-1}(P_{N-2,K-1})} \end{pmatrix},$$

where $\boldsymbol{SH_r^b(P_{i,j})}$ is a cyclic shift operator that shifts vector $\boldsymbol{P_{i,j}}$ cyclically to the right by $\boldsymbol{b}$ positions and we denote the shift direction to the right by $\boldsymbol{r}$. For example, $\boldsymbol{SH_r^1(P_{1,0})} = \boldsymbol{SH_r^1}(3,4,5) = (5,3,4)$.

Consider again the example shown in Figure 1. We have

$$\boldsymbol{m_0} = \begin{pmatrix} \boldsymbol{P_{0,0}} \\ \boldsymbol{P_{0,1}} \\ \boldsymbol{P_{0,2}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{0,1,2} \\ \boldsymbol{3,4,5} \\ \boldsymbol{6,7,8} \end{pmatrix}$$

$$\boldsymbol{m_1} = \begin{pmatrix} \boldsymbol{P_{1,0}} \\ \boldsymbol{P_{1,1}} \\ \boldsymbol{P_{1,2}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{SH_r^0(P_{0,0})} \\ \boldsymbol{SH_r^1(P_{0,1})} \\ \boldsymbol{SH_r^2(P_{0,2})} \end{pmatrix} = \begin{pmatrix} \boldsymbol{0,1,2} \\ \boldsymbol{5,3,4} \\ \boldsymbol{7,8,6} \end{pmatrix}, \textbf{ and}$$

$$\boldsymbol{m_2} = \begin{pmatrix} \boldsymbol{P_{2,0}} \\ \boldsymbol{P_{2,1}} \\ \boldsymbol{P_{2,2}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{SH_r^0(P_{1,0})} \\ \boldsymbol{SH_r^1(P_{1,1})} \\ \boldsymbol{SH_r^2(P_{1,2})} \end{pmatrix} = \begin{pmatrix} \boldsymbol{0,1,2} \\ \boldsymbol{4,5,3} \\ \boldsymbol{8,6,7} \end{pmatrix}.$$

The final mapping table is given by

$$\boldsymbol{M} = (\boldsymbol{m_0}, \boldsymbol{m_1}, \boldsymbol{m_2}) = \begin{pmatrix} \boldsymbol{0,1,2,0,1,2,0,1,2} \\ \boldsymbol{3,4,5,5,3,4,4,5,3} \\ \boldsymbol{6,7,8,7,8,6,8,6,7} \end{pmatrix}.$$

The resultant mapping of subRAIDs to disks is shown in Figure 1. Recall the notation of ***S.L*** where ***S*** represents subRAID number whereas ***L*** represents logic disk number. Mapping subRAIDs to disks in this way can achieve maximal parallelism during data reconstruction. The proof of this claim is similar to the concept reported in [4] and omitted here due to page limit.

The design concept presented above can also be applied to other RAID levels. Consider a RAID10, instead of striping data regularly as done in traditional RAID10, S^2-RAID forms subRAIDs using smaller partitions and skew data partitions in such a similar way as RAID5 described above. Figure 2 shows one example of S^2-RAID design for RAID10 with 8 disks and subRAID size of 8. Disks D0, D1, D2, and D3 are data disks and D4, D5, D6, and D7 are mirror disks, respectively. Instead of directly mirroring data stripes as done in traditional RAID10, S^2-RAID skews data partitions in mirror disks by shifting each subsequent partition by one position as shown in the figure. In this way, data reconstruction can be done in parallel in case of a disk failure. As an example, if disk D2 fails as shown in Figure 2 in shaded area, data in D2 had been mirrored across D4 through D7 that can be read out in parallel. The data read from the 4 mirror disks can then be written in parallel to spare disks. Note that the 4 spared disks labeled S0 through S3 shown in Figure 2 can be either physical spare disks if they are available or available disk space on the 8 data/mirror disks.

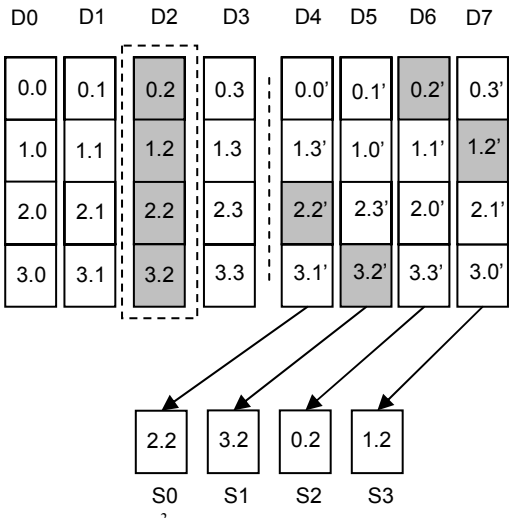

Figure 2. S^2-RAID data layout for RAID 10.

2.2 Mapping of Sub Arrays to LUNs

S^2-RAID divides disks into subRAIDs resulting in smaller stripe sizes that may adversely affect overall IO throughput during normal data services. However, such performance impact can be easily eliminated by proper mapping of LUNS seen by clients to the S^2-RAID storage system. Figure 3 shows an example of mapping subRAID to user LUNs allowing the same level of parallelism as the original RAID without subRAIDs. This mapping is based on the S^2-RAID shown in Figure 1. Suppose each subRAID can hold ***K*** units of data. Users' data will be stored on the first three subRAIDs, SR0, SR1, and SR2, first. We store data units 0, 1, 2 on SR0, SR1, and SR2, respectively. Each of these three data units is composed of two data chunks and one parity chunk to be stored on three separate physical disks (refer to Figure 1). The three data units together form a stripe as seen by storage users across the 3 sub arrays. As a result, the 3 data units are physically stored on 9 disks. This organization is similar to RAID5+RAID0 or RAID50.

After the first three subRAIDs are filled up with ***K*** data units each, we move on to the next three subRAIDs, SR3,

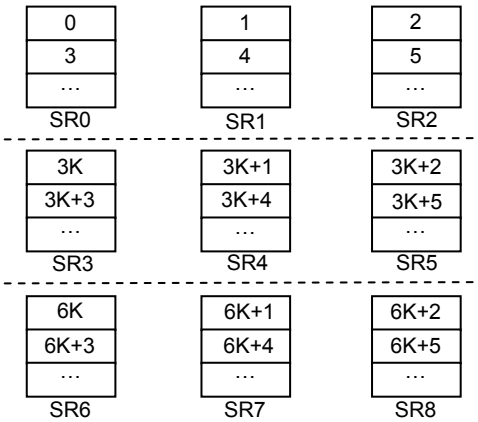

Figure 3. Mapping of Sub array to LUNs.

SR4, and SR5 starting from data unit ***3K*** as shown in Figure 3. The same process repeats until all the three subRAIDs are filled out. After that, we move on to the next three subRAIDs and so forth. This mapping ensures that storage users see the S^2-RAID same as the original RAID in terms of data striping and parallel disk accesses. All the subRAID partitions and data mappings are done transparently to users at lower storage level.

However, there is an additional storage overhead brought by S^2-RAID to store additional parity blocks. For the example in Figure 1, S^2-RAID requires a parity block for every two data chunks because the subRAID size is three. The original RAID, on the other hand, needs one parity block for every 8 data chunks because the stripe size is 8 plus one parity. In this particular example, the additional space is one third of the total space. However, the percentage of space overhead reduces as the RAID size increases. For example, the space overhead of a 36-disk S^2-RAID is one sixth of the total volume. This additional space overhead can be justified as the price per gigabyte of disks decreases rapidly and the importance of data reliability and availability increases.

3 PROTOTYPE IMPLEMENTATION

To demonstrate the efficiency of S^2-RAID architecture, we have implemented a proof-of-concept prototype inside the iSCSI target of the Linux operating system. On top of IET (iscsitarget-0.4.1.7 [5]), we realized the S^2-RAID functionalities. Figure 4 shows the software structure of our prototype implementation. It includes mainly three modules, iSCSI target module, S^2-RAID function module, and Config module.

The iSCSI target module modifies the IET SCSI command handling and disk IO parts. The disk IOs of the IET call upon interfaces of the S^2-RAID module.

The S^2-RAID module realizes the basic functions of RAID10 and RAID5 including RAID rebuilder based on MD. MD itself provides RAID rebuilder that allows parallel reconstruction of multiple RAIDs for a disk failure provided that these RAIDs do not share physical disks. When multiple RAIDs share physical disks, MD's rebuilder reconstruct data sequentially with no parallelism.

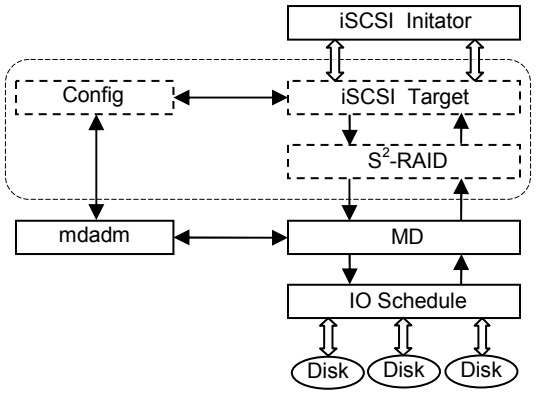

Figure 4. Software structure of S^2-RAID implementation.

S^2-RAID5 employs the special mapping table discussed in Section 2 to ensure sub arrays do not share physical disks. In this way, upon a disk failure, we are able to use MD's rebuilder to reconstruct data in parallel.

For S^2-RAID10 shown in Figure 2, 4 subRAIDs shared physical disks. To allow parallel reconstruction, we built 4 subRAID1 of 2 disks each on top of MD. The S^2-RAID module then forms subRAID10 using the 4 subRAID1. From MD point of view, there are 4 independent RAID1s without sharing physical disks allowing parallel reconstruction in case of failures.

The S^2-RAID module finally maps the multiple subRAIDs to one unified LUN. This LUN presents to the iSCSI target module as one logic disk for read and write IOs.

The Config module provides RAID setup and configuration functions using mdadm commands to realize different S^2-RAID subRAID functions. It also allows users to configure iSCSI target by means of iSCSI configuration functions.

4 EVALUATION METHODOLOGY

This section presents our experimental setting and methodology that we use to study quantitatively the performance of the S^2-RAID prototype.

4.1 Experimental Settings

The prototype S^2-RAID is installed on a storage server that is an iSCSI target. Storage clients are connected to the storage server using the Cisco 3750 Gb Ethernet. The hardware details of the storage server and the disks are listed in Table 1. The hardware details of the application server as the storage client is shown in Table 2.

OS	*Fedora Core 8.0*
disks	*1 Seagate ST3160023AS, 160GB, 7200RPM. 12 Seagate ST3500320AS, 500GB, 7200RPM.*
mainboard	*SUPER X7DVL-I*
CPU	*Intel(R) Xeon(R) CPU 5110 @ 1.60GHz*
NIC	*Intel® PRO/1000*
memory	*1GB DDR2*
HBA	*Highpoint 2240 RAID,*

Table 1. Hardware details of the storage server.

OS	*Fedora Core 8.0*
disks	*Seagate ST3160023AS, 160GB, 7200RPM*
mainboard	*GA-945GCMX-S2*
CPU	*Intel(R) Celeron(R) CPU 2.80GHz*
NIC	*Tigon3*
memory	*512MB DDR2*

Table 2. Hardware details of app server as storage client.

For RAID5, we use 9 disks to construct S^2-RAID as shown in Figure 1. The number of spare disks varies from 1 to 3. The traditional RAID uses MD with 8 data disks and one parity disk

with the same number of spares. For RAID10, we configure S^2-RAID using 8 disks as shown in Figure 2 with the same number of spares. The baseline RAID mirrors data in the same way as traditional RAID10.

In order to run standard benchmarks, we have set up two different types of databases on the servers: Postgres Database 8.1.15 and MySQL 5.0 database installed on Fedora 8. To be able to run real world web applications, we installed Tomcat 5.5 application server for processing web application requests issued by benchmarks. The details of the software packages used are listed in Table 3.

blktrace	*blktrace 1.0*
postmark	*postmark1.5.1*
TPC-C	*TPC-C tpccuva-1.2.3*
postgresql	*postgresql 8.1.15*
gnuplot	*gnuplot 4.2.5*
TPC-W	*TPC-W 1.5*
jdk	*jdk 1.5 .0.06*
tomcat	*tomcat 5.5*
mysql	*mysql 5.0.45*
iscsi initator	*iscsi-initiator-util 6.2.0865*

Table 3. Software settings of the storage server and client.

4.2 Workloads

The workloads that drive our experiments consist of two major parts: SPC IO traces and standard IO benchmarks.

Table 4 shows the statistics of SPC Traces that are used in our performance evaluation. Three different traces are used, Financial-1, Financial-2, and Websearch. On the application server, we replay the IO traces using *btreplay* program of the *blktrace* tool in Linux. As results of the replay, IO requests are generated to the storage server in the form of iSCSI requests. The S^2-RAID inside the iSCSI target handles the iSCSI requests.

Trace File	*Write Ratio*	*Ave Req Size: KB*	*Total Req*
Financial-1	*76.84%*	*3.38*	*5,334,987*
Financial-2	*17.65%*	*2.39*	*3,699,195*
Websearch	*0%*	*15.07*	*4,579,809*

Table 4. SPC trace characteristics.

We choose a set of standard benchmarks that are widely used in the research community and industry. The first benchmark we selected is PostMark [6] that is widely used as file system benchmark tool written by NetApp, Inc. In our experiments, we set PostMark workload to include 20,000 files of size 4KB to 500KB and to perform 100,000 transactions. Read and write buffer sizes are set to 4KB.

We have also chosen two typical benchmarks that run on databases. TPC-C is a well-known benchmark used to model the operational end of businesses where real-time transactions are processed [7]. We set up the Postgres database based on the implementation from TPCC-UVA [8]. 20 warehouses with 10 terminals per warehouse are built on Postgres database with measurement interval of 120 minutes. Details regarding TPC-C workloads specification can be found in [9]. TPC-W, is a transactional web benchmark that models an online bookstore. We use the Java TPC-W implementation of the university of Wisconsin-Madison [10] and build an experimental environment. This implementation uses Tomcat 5.5 as an application server and MySQL 5.0 as a backend database. The configured workload includes 150 emulated browsers (Rbe type 2 shopping mix) and 100,000 items in the item table.

5 Numerical Results and Discussions

Using our prototype implementation and the experimental settings described in the previous sections, we carried out experiments to measure the performance of S^2-RAID as compared to the traditional software RAID, MD. All our experiments assume one spare disk unless otherwise specified. Parallel reconstruction is realized by writing rebuilt data into the spare disk and available partitions on other data disks that are not involved in the reconstruction. In order to finish a large number of experiments in the limited time, we assume the volume of each disk to be 10GB.

Table 5 shows our measurement results using the SPC traces. We measured both front end performance in terms of average user response times while online reconstruction is in progress and the total online reconstruction times for the three traces. It can be seen from the table that S^2-RAID speeds up the online reconstruction times by a factor of 5 to 6 for the three traces. These dramatic speedups can be attributed to the conflict free parallel reconstructions of subRAIDs. Recall Figure 1 where subRAIDs 1, 3, and 8 can be reconstructed in parallel without conflict when physical disk D4 failed. In the traditional RAID, on the other hand, failed disk is rebuilt only one data chunk at a time using remaining chunks of each stripe reside on good disks. Therefore, S^2-RAID can clearly speedup the reconstruction process.

Trace File	*Average User Response Time (ms)*			*Reconstruction Time (in seconds)*		
	MD	*S^2-RAID*	*Speed-up*	*MD*	*S^2-RAID*	*Speed-up*
Fin1	*14.98*	*10.81*	*1.39*	*1799*	*293*	*6.14*
Fin2	*8.51*	*7.87*	*1.08*	*886*	*172*	*5.15*
Web	*13.32*	*11.19*	*1.19*	*4820*	*849*	*5.68*
Off-line				*157*	*51*	*3.08*

Table 5. RAID5 reconstruction performance.

It is interesting to note in Table 5 that S^2-RAID not only speeds up online reconstruction time but also provides better front end performance during reconstruction. We observed 8% to 39% performance improvement in terms of average user response time. Our analysis gives the following three reasons for such improvements. First of all, we noticed that S^2-RAID provides better performance for high write IO ratios. For example, the write ratio of Financial-1 trace is 76.84% and S^2-RAID improves user response times by 39% on average. In S^2-RAID5, the stripe size is 3 requiring 1 read IO and 1 write IO as opposed to 4 IOs in traditional RAID for each chunk write.

The second and more important reason is the number of IOs required to write in the failed disk. S^2-RAID needs 1 read IO to compute parity with the new data to be written and 1 write IO for parity updates while traditional RAID needs 7 read IOs to compute parity before updating new parity. Similar to read IOs, reading data from failed disk requires 2 read IOs to compute damaged data in S^2-RAID whereas traditional RAID needs 8 read IOs to compute damaged data back. As a result of these three reasons, S^2-RAID shows better front end performance than traditional RAID during online RAID rebuild.

From Table 5, we also observed 3.08 times better offline reconstruction performance of S^2-RAID than traditional RAID. The amount of improvement of offline reconstruction is smaller than online reconstruction. This is because MD RAID bears much more burden on disk loads during online reconstruction than S^2-RAID because of more disk operations for user read/write IOs as discussed above. S^2-RAID, on the other hand, reduces the number of disk operations for each read/write IO from users lowering the disk workloads compared to MD RAID. As a result, MD RAID takes much longer time during online reconstruction. For offline reconstruction, the over burden of user IO requests is no longer present giving rise to relatively smaller difference between reconstruction times of MD RAID and S^2-RAID. However, even though all disks in the traditional RAID works harder than in S^2-RAID, the traditional RAID still does not offer better front end performance than S^2-RAID as shown in the table. Therefore, S^2-RAID shows its clear advantage both in terms of RAID rebuild time (online/offline) and performance impacts on front end IO performance during online reconstruction.

Trace File	*Average User Response Time (ms)*			*Reconstruction Time (in seconds)*		
	MD	*S^2-RAID*	*Speed-up*	*MD*	*S^2-RAID*	*Speed-up*
Fin1	*9.05*	*8.80*	*1.03*	*6557*	*1770*	*3.70*
Fin2	*8.25*	*8.24*	*1.00*	*7138*	*2075*	*3.44*
Web	*15.91*	*11.59*	*1.37*	*9080*	*2163*	*4.20*
Off-line				*152*	*44*	*3.45*

Table 6. RAID10 reconstruction performance.

Trace File	*Normal*			*Degraded*		
	MD	*S^2-RAID*	*Speed-up*	*MD*	*S^2-RAID*	*Speed-up*
Fin1	*13.85*	*10.49*	*1.32*	*15.60*	*10.74*	*1.45*
Fin2	*7.66*	*7.61*	*1.01*	*9.13*	*7.86*	*1.16*
Web	*10.45*	*10.47*	*1.00*	*12.94*	*11.40*	*1.14*

Table 7. RAID5 normal and degrade performance (ms)

We have also measured reconstruction performance of S^2-RAID10 with the configuration shown in Figure 2. The measured results are shown in Table 6. It is observed that the online reconstruction performances of S^2-RAID10 are between 3.44 and 4.2 times as good as those of traditional RAID10. Similar to S^2-RAID5, the performance improvement is due to the parallel reconstruction. The front end performance of S^2-RAID10 is about the same as traditional RAID10 for Financial-1 and Financial-2 traces. For web traces, we observed 37% performance improvement because of S^2-RAID10's better parallelism in handling multiple requests.

From the above reconstruction experiments, one can easily see the advantages of S^2-RAID for data reconstruction in case of a failure. In order to see how S^2-RAID performs while no reconstruction is being done as compared to traditional RAID, we have measured average user response times without RAID reconstruction. Table 7 shows such measured results under normal working condition as well as under degraded working mode after one disk failed. For Financial-1 traces, we observed 32% performance gain compared to MD because of high write ratio. For the other two traces, there is hardly any difference in performance. However, when a disk failed, S^2-RAID showed much better performance than MD due to reduced number of disk operations to calculate data stored in failed disk as discussed above. The performance improvement ranges from 14% to 45% as shown in the table. However, such performance gains come at the additional disk space to store more parities because of smaller stripe size of S^2-RAID than that of traditional RAID.

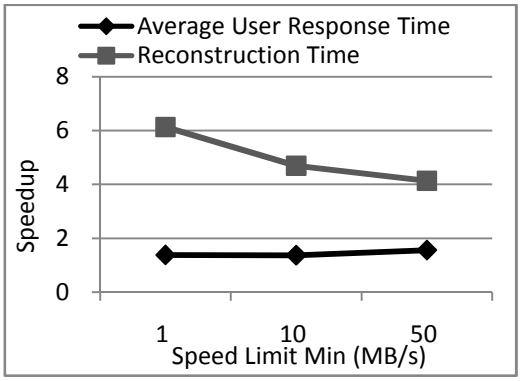

Figure 5.Effects of different rebuild speed thresholds.

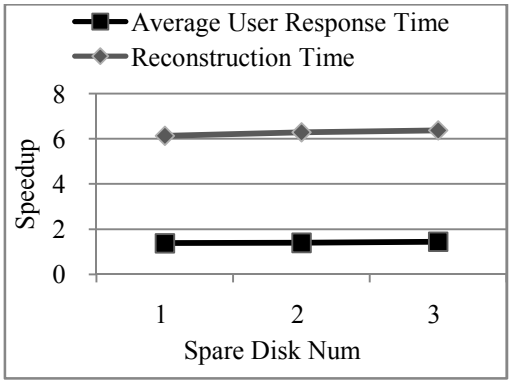

Figure 6. Effects of number of spare disks on performance.

RAID reconstruction speed of MD can be configured by setting the minimum threshold value. The default minimum speed threshold is 1MB/s and the maximum threshold is 200MB/s. The reconstruction threads checks disk loads after rebuilding every 128KB. When the disk load exceeds the predefined value, it is put to sleep. To see how S^2-RAID5

compares with MD with different speed thresholds, we carried out experiments by setting three different thresholds to measure the performances of both S^2-RAID5 and MD RAID5. Figure 5 shows the measured results for three different speed thresholds (speed_limit_min) being 1M/s, 10MB/s, and 50MB/s, respectively, using Financial-1 traces. In general, performance improvement of S^2-RAID5 reduces as speed_limit_min increases. This is because as the reconstruction speed increases, the fixed bus bandwidth and available CPU time for XOR computation limit the parallelism of reconstruction. As a result, the speedup factor of S^2-RAID decreases. However, the minimal speedup is still 4.14. On the other hand, high reconstruction speed also negatively impact front end performance seen by users. When speed_limit_min is set at 50MB/s, S^2-RAID5 provides 57% better performance than MD.

The main reason that S^2-RAID improves reconstruction performance is its parallel reads and writes during reconstruction process. To be able to do parallel data writes while rebuilding RAID, we need more spare disks to reconstruct data. However, more spare disks imply high hardware cost. In order to keep the hardware cost low, one alternative is to use available space on other data disks to act as spare disks. The question is whether using other data disks to carry out data reconstruction will negatively impact reconstruction performance. In order to see this effect, we have measured reconstruction performance using different number of spare disks as shown in Figure 6 using Financial-1 traces. Every time when an additional spare disk is added, reconstruction write IOs on a data disk are reduced and hence reducing the impact on front end performance. However, such negative impact is not as significant as we had expected as shown in Figure 6. The performance degradation is about 2-3% every time a spare disk is taken away. This is because we choose data disks that are not needed to read data for rebuilding and write operations are fairly fast. We noticed that if extra load is put on disks involved in rebuilding data, it will become bottleneck for data reconstruction.

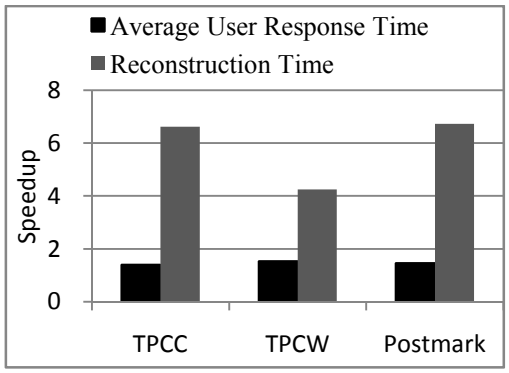

Figure 7. Benchmark performance.

In addition to IO traces, we have also measured reconstruction performance using standard benchmarks as shown in Figure 7. Since all the three benchmarks run above the file system buffer cache before going to the iSCSI target through iSCSI-Initiator, the actual IO loads seen at the RAID system are smaller than traces. To meet the benchmark requirements, each disk is configured to have 100GB space. For TPC-C and PostMark benchmarks, we observed over a factor of 6 speedups for online reconstruction. For TPC-W, the IO workload is much lighter than the other two giving rise to smaller speedup: 4.25. The front end performance in terms of user response time is improved by 40% to 54% due to S^2-RAID.

Since S^2-RAID uses more parity disks than traditional RAID systems, one question arises as to whether such additional parity disks hurt write performance. Further experiments were carried out to measure the write performance of S^2-RAID as compared to traditional RAID systems. Figure 8 shows the IOMeter results of S^2-RAID in terms of I/O rate for both sequential and random write I/Os with different block sizes. As shown in Figure 8, both sequential and random write I/O performances of S^2-RAID are significantly better than those of traditional RAID5. Such good write performance of S^2-RAID can be attributed to the small number of I/Os for each write operations as discussed previously.

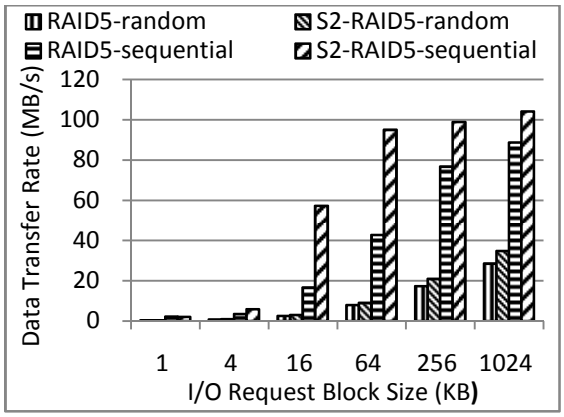

Figure 8. Write performance of S2-RAID as compared with traditional RAID5 measured using IOMeter.

6 Related Work

Data layout: There have been several approaches to improving RAID performance by means of data layouts [11-14]. A Parity Declustering algorithm was proposed by Muntz and Lui [12] to utilize as few disks as possible in data reconstruction so that the rest of the disks can serve foreground requests resulting in improved reconstruction times and user response times. Holland and Gibson [13, 15] implemented a Parity Declustering algorithm to enhance online IO performance and reconstruction performance. S^2-RAID differs from existing parity declustering approaches in forming and organizing subRAIDs, not smaller stripes, that are skewed among disks so that disk accesses are conflict free during RAID reconstructions. In addition, Parity Declustering algorithm just presents the general RAID organizations using the concept "declustering ratio", although it can decrease the user response times during reconstruction, the construction time can not be obviously improved. While such subRAIDs formations allow large sequential reads and writes on disks during reconstruction in additional to parallel and concurrent rebuilding of sub arrays.

There are also existing works on selecting appropriate data organizations stored in RAID systems according to workload characteristics or application scenarios. By applying different strategies adapting to different workloads, IO performance can

be improved such as HP AutoRAID [16], Muti-tier RAID [17], Two-tiered Software Architecture [18], Muti-partition [19] and so on. Cortes and Labarta [20] proposed a data layout algorithm for RAID5 consisting of heterogeneous disks and showed that it provides better throughput than block distribution algorithms.

Tsai [19] presented a new variation of RAID organization, Muti-partition RAID (mP-RAID), to improve storage efficiency and reduce performance degradation when disk failures occur. Their Muti-partition data layout is similar to what we referred to as subRAID. The key difference between mP-RAID and S^2-RAID is that S^2-RAID ensures that any two subRAIDs share no more than one disk. This property is important to allow conflict-free parallel reconstruction of RAID after a failure.

Extensive research has been reported in the literature on data layout of RAID to improve reliability [21-26]. In addition to mirroring structure, different data layouts and coding schemes [27-30] have been presented to tolerate single or more disk failures. In particular, Amer et al. [26] presented a data layout called SSPiRAL to improve data reliability by several orders of magnitude. While sharing some similarity in terms of changing data layouts in RAID, S^2-RAID focuses on fast RAID reconstruction through conflict-free parallel reconstruction of data after failures.

RAID Reconstruction: Realizing the importance of shortening the "window of vulnerability", there is a great deal of research reported in speeding up RAID reconstruction through exploiting workload characteristics [31-35], data or parity reorganization [21, 23] and system structures [36]. Wu et al. [35] proposed a surrogate RAID approach in their WorkOut that saves hot and popular data and rebuilds highly popular data units prior to rebuilding other units when a disk failed. The merit of this approach is that it not only reduces online reconstruction time but also the user average response time. Making use of the file system's semantic knowledge, Sivathanu et al. [36] proposed a live-block recovery approach in their D-GRAID, which rebuilds first the file that are live. The approach of mining popular data in file system, such as Tian et al. [34], attempts to collect hot data information to rebuild.

To balance user response time and disk rebuild time in degraded mode, Hou et al. [37] presented an optimization method which takes disk track as the rebuild unit. Rebuilding one track at a time provides better user response times than rebuilding one cylinder at a time. Bachmat and Schindler [38] presented a new algorithm that has less impact on foreground activity with the priority scheduling. The algorithm also reduces the total reconstruction time.

S^2-RAID is a skewed subRAID architecture that is orthogonal and complementary to above related works. It can be applied on top of above techniques to speed up reconstruction performance further. The key benefit of S^2-RAID is its conflict free parallel reads of good disks and parallel writes of reconstructed data into multiple spare disks.

Another research work that is related to making use of parallel spare disks in S^2-RAID is a technique called distributed sparing by Menon and Mattson [21]. The objective of this work is to utilize otherwise unused spare disks in the system. Spare disk is dispersed to the data disks of the RAID, which can achieve good access performance and also speed up data reconstruction. And S^2-RAID also can take the rebuilt data to the reserved space in the disk which has not been participated in read operations, But the difference is that S^2-RAID makes use of multiple subRAIDs to read in parallel and performs write operations to the spare disks that are not involved in reading during reconstruction, while the method of distributed sparing exists read conflicts when reconstruction is done which does not truly realize parallelism.

7 Conclusions and Future Work

In this paper, we have presented a new disk array architecture, S^2-RAID, for the purpose of fast data reconstruction in case of disk failures. The idea is to partition large disks into smaller logic disks to form sub arrays. The sub arrays are skewed among the physical disks in such a way that reconstruction can be done in parallel and conflict free. By making use of multiple spare disks that consist of either physical spares or available disk space on data disks, S^2-RAID substantially speeds up RAID reconstruction time. A prototype S^2-RAID has been built in the iSCSI target to measure the performance of the RAID architecture. Numerical results using SPC traces and standard benchmarks show that S^2-RAID improves RAID reconstruction time by a factor of 3 to 6. The front end IO performance during online RAID reconstruction is comparable to and better than in some cases the baseline software system.

We are currently working on optimizing foreground performance under different workloads and comparing S^2-RAID performance with other reconstruction techniques discussed in the related work. Another future research is considering multiple disk failures in disk arrays.

Acknowledgment

The authors would like to thank Bo Cheng and Panpan Hu for their help in programming of the prototype, Jianzong Wang for detailed comments, and Minmin Ren for trace analysis. This work is supported by National Natural Science Foundation of China under Grant No. 60933002 and NSFC-60736013, Natural 863 Plan under Grant No. 2009AA01A402, and in part by the National Science Foundation under Grants CCF-0811333 and CPS-0931820. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. We would also like to thank the anonymous reviewers for their valuable insights that have improved the quality of the paper greatly.

References

[1] D. A. Patterson, G. Gibson and R. H. Katz, "A case for redundant arrays of inexpensive disks (RAID)," in *Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, pp. 109-116, Chicago, Illinois, United States, 1988.

[2] Q. Xin, E. L. Miller, T. Schwarz, D. D. E. Long, S. A. Brandt and W. Litwin, "Reliability Mechanisms for Very Large Storage Systems," in *Proceedings of the 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies*, Apr. 2003.

[3] G. Gibson, "Reflections on Failure in Post-Terascale Parallel Computing," in *International Conference on Parallel Processing*, Xi'An, China, Sep. 2007.

[4] Q. Yang and L. W. Yang, "A novel cache design for vector processing," in *Proceedings of the 19th annual international symposium on Computer architecture(ISCA'92)*, pp. 362-371, Queensland, Australia, May 1992.

[5] iSCSI Enterprise Target, 2008, Available: http://sourceforge.net/projects/iscsitarget/files/.

[6] J. Katcher, "Postmark: a new file system benchmark," Network Appliances,Rep. 1997.

[7] S. T. Leutenegger and D. Dias, "A modeling study of the TPC-C benchmark," *SIGMOD Record*, 22(2), pp. 22-31, Jun. 1993.

[8] D. R. Llanos, "TPCC-UVa: an open-source TPC-C implementation for global performance measurement of computer systems," *SIGMOD Record*, 35(4), pp. 6-15, 2006.

[9] Transaction Processing Performance Council, TPC BenchmarkTM C Standard Specification, 2005, Available: http://www.tpc.org/tpcc/.

[10] H. W. Cain, R. Rajwar, M. Marden and M. H. Lipasti, "An architectural evaluation of Java TPC-W," in *Proceedings of the 7th International Symposium on High-Performance Computer Architecture*, pp. 229-240, Monterrey, Mexico, Jan. 2001.

[11] E. K. Lee, "Software and Performance Issues in the Implementation of a RAID," University of California at Berkeley,Rep. 1990.

[12] R. R. Muntz and J. C. S. Lui, "Performance Analysis of Disk Arrays under Failure," in *Proceedings of the 16th International Conference on Very Large Data Bases*, 1990.

[13] M. Holland and G. Gibson, "Parity declustering for continuous operation in redundant disk arrays," in *Proceedings of the 5th international conference on Architectural support for programming languages and operating systems*, Boston, Massachusetts, United States, 1992.

[14] Z. Dimitrijevic, R. Rangaswami and E. Chang, "Preemptive RAID Scheduling," University of California, Santa Barbara,Rep. 2004.

[15] M. Holland, "*On-Line Data Reconstruction In Redundant Disk Arrays*," Ph.D thesis, Carnegie Mellon University, May 1994.

[16] [16] J. Wilkes, R. Golding, C. Staelin and T. Sullivan, "The HP AutoRAID hierarchical storage system," *ACM Transactions on Computer Systems*, 14(1), pp. 108-136, 1996.

[17] N. Muppalaneni and K. Gopinath, "A multi-tier RAID storage system with RAID1 and RAID5," in *Proceedings of the14th International Parallel and Distributed Processing Symposium*, pp. 663-671, Cancun, Mexico, 2000.

[18] B. Salmon, E. Thereska, C. A. N. Soules and G. R. Ganger, "A two-tiered software architecture for automated tuning of disk layouts," in *Proceedings of the 1st Workshop on Algorithms and Architectures for Self-Managing Systems*, pp. 13–18, 2003.

[19] W. J. Tsai and S. Y. Lee, "Multi-partition RAID: a new method for improving perform of disk arrays under failure," *The Computer Journal*, 40(1), pp. 30-42, 1997.

[20] T. Cortes and J. Labarta, "Extending Heterogeneity to RAID level 5," in *Proceedings of the 2001 USENIX Annual Technical Conference*, Boston, MA, Jun. 2001.

[21] J. Menon and D. Mattson, "Distributed sparing in disk arrays," in *Proceedings of the 37th international conference on COMPCON*, pp. 410-421, San Francisco, California, United States, 1992.

[22] M.-Y. Lee and M.-S. Park, "Double parity sparing for performance improvement in disk arrays," in *Proceedings of the 1996 International Conference on Parallel and Distributed Systems*, Jun. 1996.

[23] Q. Xin, E. L. Miller and T. J. E. Schwarz, "Evaluation of Distributed Recovery in Large-Scale Storage Systems," in *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing*, pp. 172-181, Jun. 2004.

[24] D. Stodolsky, G. Gibson and M. Holland, "Parity logging overcoming the small write problem in redundant disk arrays," in *Proceedings of the 20th annual international symposium on Computer architecture*, pp. 64-75, San Diego, California, United States, May 1993.

[25] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong and S. Sankar, "Row-Diagonal Parity for Double Disk Failure Correction," in *Proceedings of the 3th USENIX Conference on File and Storage Technologies*, pp. 1-14, San Francisco, CA, Mar. 2004.

[26] A. Amer, D. D. Long, J.-F. Paris and T. Schwarz, "Increased reliability with SSPiRAL data layouts," in *Proceedings of the IEEE International Symposium onModeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'08)*, Baltimore, MD, USA, Sep. 2008.

[27] M. Blaum, J. Brady, J. Bruck and J. Menon, "EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures," *IEEE Transactions on Computers*, 44(2), pp. 192-202, Feb. 1995.

[28] W. Litwin, R. Moussa and T. Schwarz, "LH*RS---a highly-available scalable distributed data structure," *ACM Transactions on Database Systems*, 30(3), pp. 769-811, Sep. 2005.

[29] G. A. Alvarez, W. A. Burkhard and F. Cristian, "Tolerating multiple failures in RAID architectures with optimal storage and uniform declustering," in *Proceedings of the 24th annual international symposium on Computer architecture*, Denver, Colorado, United States, 1997.

[30] J. L. Hafner, V. Deenadhayalan, K. K. Rao and J. A. Tomlin, "Matrix methods for lost data reconstruction in erasure codes," in *Proceedings of the 4th conference on USENIX Conference on File and Storage Technologies*, San Francisco, CA, Dec. 2005.

[31] J. Y. B. Lee and J. C. S. Lui, "Automatic Recovery from Disk Failure in Continuous-Media Servers," *IEEE Transactions on Parallel and Distributed Systems*, 13(5), pp. 499-515, 2002.

[32] T. Lei, J. Hong, F. Dan, X. Hong Qin and S. Xing, "Implementation and Evaluation of a Popularity-Based Reconstruction Optimization Algorithm in Availability-Oriented Disk Arrays," in *Proceedings of the 24th IEEE Conference on Mass Storage Systems and Technologies*, pp. 233-238, San Diego, CA, Sep. 2007.

[33] M.-S. Chen, D. D. Kandlur and P. S. Yu, "Optimization of the grouped sweeping scheduling (GSS) with heterogeneous multimedia streams," in *Proceedings of the 1th ACM international conference on Multimedia*, pp. 235-242, Anaheim, California, United States, 1993.

[34] L. Tian, D. Feng, H. Jiang, K. Zhou, L. Zeng, J. Chen, Z. Wang and Z. Song, "PRO: a popularity-based multi-threaded reconstruction optimization for RAID-structured storage systems," in *Proceedings of the 5th USENIX conference on File and Storage Technologies*, San Jose, CA, 2007.

[35] S. Wu, H. Jiang, D. Feng, L. Tian and B. Mao, "WorkOut: I/O workload outsourcing for boosting RAID reconstruction performance," in *Proccedings of the 7th conference on File and storage technologies*, pp. 239-252, San Francisco, California, Feb. 2009.

[36] M. Sivathanu, V. Prabhakaran, A. C. Arpaci-Dusseau and R. H. Arpaci-Dusseau, "Improving storage system availability with D-GRAID," *ACM Transactions on Storage*, 1(2), pp. 133-170, 2005.

[37] R. Y. Hou, J. Menon and Y. N. Patt, "Balancing I/O response time and disk rebuild time in a RAID5 disk array," in *Proceedings of the 26th Hawaii International Conference on System Sciences*, pp. 70-79, 1993.

[38] E. Bachmat and J. Schindler, "Analysis of methods for scheduling low priority disk drive tasks," *ACM SIGMETRICS Performance Evaluation Review*, 30(1), pp. 55-65, Jun. 2002.