ERAID: a New Storage Architecture for Mid/Large RAID

Liu Yan[1], Xie Chang-Sheng[2], Li Huai-yang[3], Zhao Zhen[4]

[1,2,3] Key laboratory of Data Storage System, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
[1] to-liuyan@sohu.com, [2] csxie@263.net, [3] lhycly@263.com, [4] zhaozhen@163.net

Abstract. Referring to the concept of “distributed virtual disks” in cluster file system, this paper proposes a new storage architecture for mid/large-scale RAID – ERAID, which consists of several different structure SubRAIDs. Clients data can migrate into optimal SubRAID dynamically according to their access characteristics; the number of SubRAID and the structure of each SubRAID can also be adjusted to the evolution of system load, thus the whole ERAID system can always provide a virtual disk for each client to satisfy its I/O need. Simulation results demonstrate significant I/O performance advantage in ERAID, compared with the traditional mid/large-scale RAID5 or RAID1 of the same storage space.

1 Introduction

RAID improves storage system I/O performance by taking advantage of parallel work of multiple disks, and provides storage system availability by using redundancy scheme [1]. Of various levels of RAID, RAID1 and RAID5 are most popular practically. RAID1 maintains 2 copies of data in different disks, is easier to configure and can provide best I/O performance, while under rather high redundancy overhead. Adopting rotating parity scheme, RAID5 is the least expensive RAID scheme with good large update performance (when writes request spans a integral number of stripe units), but it suffers from poor small write performance, and furthermore, its performance drops sharply when it enters degraded mode with a disk failure.

In recent years, with the explosion of applications that use enormous amounts of data and have high QoS requirements to their storage systems, the market of enterprise storage system is growing gradually [2]. To satisfy the I/O bandwidth and availability requirements of these applications, enterprise storage system typically contain RAID. However, the architecture of current RAID is complex: prior RAID contains several or dozens of disks, while the number of disks in current RAID has been expanded tens (*mid-scale* RAID) or even hundreds (*large-scale* RAID). So the modern enterprise RAID should manage reasonably and take full advantage of its large amounts of disks, to provide its users a storage subsystem with high storage capacity, high performance, high availability. We noticed that modern enterprise

This work was supported by Chinese “973” funds under grants of 2004CD318203, and by NSF funds of China under grants of 60273073.

RAID has certain similarity to cluster file system in the aspect of managing a large number of disks. Apart from adopting data striping technique and redundancy scheme as in traditional RAID, cluster file system, which managing hundreds and thousands of disks distributed across network nodes, has its own characteristic: the technique of *distributed virtual disks* [3]. Owing to this technique, not only the users of cluster file system can build their own virtual disks according to the storage capacity and performance of the physical storage devices, but also the system achieves such great scalability that resources (servers and disks) could be added to the cluster file system automatically.

Using the concept of distributed virtual disk in cluster file system for reference, in this paper we propose a new architecture for *mid/large-scale* RAID – *ERAID*（*Evolving RAID*）. ERAID is implemented at the block device level of storage system. ERAID constructs all array disks into several SubRAIDs, which cooperatively constitute a unique global storage space of ERAID. A SubRAID is constructed by part of array disks, and its configuration is totally the same with traditional RAID. Each SubRAID in ERAID can adopt appropriate data placement scheme and performance optimization policy according to the number, capacity, performance of its disks, and the characteristics of the workload it serves. Thus, through SubRAID of different configuration serving different application, the whole ERAID system exhibits optimized I/O performance to all of its clients.

The paper is structured as follows. Section 2 describes related work on RAID and distributed virtual disks technique in cluster file system. Section 3 describes the ERAID system in detail. Section 4 gives experimental results. Section 5 provides our conclusions and outlines future work.

2 Related Works

Many papers have been published on RAID performance, availability, data placement scheme and recovery policy, since RAID technology was proposed in 1989. [4][5][6]

There’s also a number of works on distributed storage system containing a great amount of disks. CSAR [7] expands the architecture of HP-AutoRAID to distributed RAID. Cluster file systems, such as PVFS（Parallel Virtual File System）[8] and Lustre parallel file system [9] stripe data across multiple servers that can serve file access requests in parallel, allowing high-bandwidth I/O performance and scalability. To protect data against loss from failure of server or disk, RAID-like redundancy scheme are also applied to cluster file system: CEFT-PVFS [10] combines original PVFS striping layout with a doubling of the stripes according to RAID level 10; NetRAID [11] take the striping and redundancy scheme in RAID3 to ensure system I/O performance and fault tolerance; Zebra [12], xFS [13], Swarm [14] all store data using RAID5 redundancy, and use log-structured writes to solve the small-write problem in distributed storage system incurred by the data placement scheme of RAID5. Though consisting of large amount of disks, all these above storage systems are distributed storage systems. There hasn’t any paper special on improving mid/large-scale RAID system performance yet. So, we propose a new architecture for mid/large-scale RAID system – ERAID in this paper.

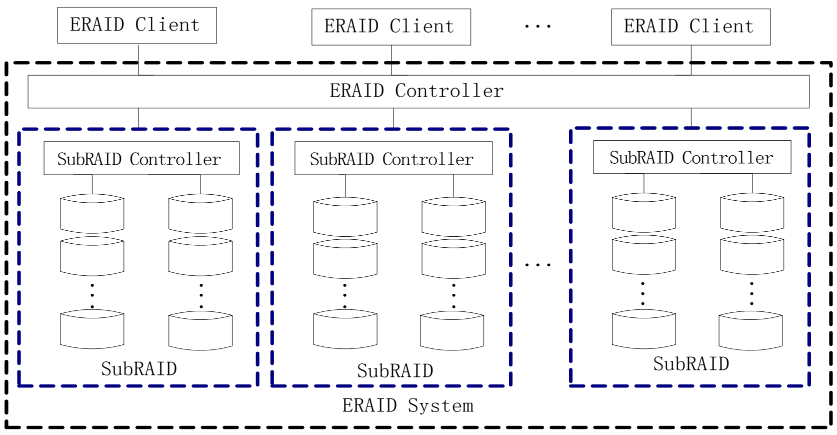


Fig. 1. Diagram of ERAID system architecture

3 The design of ERAID

3.1 the architecture of ERAID

As show in figure 1, ERAID consists of a collection of SubRAIDs that cooperatively implement a single, block-level storage system with a unique global storage space. Each SubRAID could have different data and parity placement scheme that be applied in traditional RAID. Different from the traditional RAID in which all applications access one uniform configuration array, ERAID assigns one *virtual disk* to serve each one of its clients. So clients could view the ERAID system as a collection of virtual disks. A virtual disk provides a logical storage space, which could be the whole or partial physical storage space of a single SubRAID, or is composed of several SubRAIDs' whole or partial physical storage space. Through changing the SubRAID physical storage space that each virtual disk's logical storage space maps over, ERAID system allows data to migrate dynamically among SubRAIDs according to the *evolution* of the workloads without much difficulty, (That's why we name the system ERAID.) as a sequence, to each client, ERAID could always provide a virtual disk to satisfy its I/O requirement. In addition, the configuration of each SubRAID alters dynamically to balance the workload among all SubRAIDs, and to take advantage of the performance of each disk in ERAID system fully. So, the concept of virtual disk, the characteristics of data migration among different-configuration SubRAIDs ensure that the whole ERAID system provides heterogeneous clients and client applications with optimal I/O performance.

Apart form the above-mentioned advantages, the separation of the virtual disks to clients view and the physical storage space endues that (1) new disks could be added into SubRAID without any negative impact on the service to clients, and (2) SubRAID can also be added, deleted, or reconfigured by ERAID system arbitrarily; as a consequence, ERAID system achieves scalability that new disks and other resources such as controller and buses could be added to the system flexibly. In addition, benefiting from the various configurations of SubRAIDs, homogeneous disks could also be organized easily and their performance could be utilized fully. We haven't described the scalability and homogeneity of disks of ERAID in detail, limited to space of the paper.

3.2 Virtual to physical addresses translation

The unique global storage space of ERAID could be viewed as the combination of virtual disks' logical storage space, each of which is mapped dynamically to the whole (or partial) physical storage space of single (or several) SubRAID(s). The translation from virtual disk addresses used by clients into physical disk addresses is processed by ERAID driver, consequently, the logical addresses viewed from clients need no change even though the data are migrating among SubRAIDs.

The rest of this subsection will describe how ERAID translate the virtual addresses used by clients into physical disk addresses in detail. The basic problem is to translate virtual addresses of the form <virtual_disk_id, offset_virtual_disk> to physical addresses of the form <SubRAID_id, disk_id, offset_disk>. This translation must be done accurately and efficiently, for there're many events that alter virtual address translation, such as data migration among SubRAIDs, SubRAID reconfiguration, ERAID system expand, and disk failure in ERAID.

To implement the translation, virtual disks and SubRAIDs are all partitioned into many basic storage units with same size, and we refer to the storage unit in virtual disk and storage unit in SubRAID as *virtual disk block* (VDB) and *VDB_in_SubRAID* respectively. (Without loss of generality, the size of VDB is set to 64KB in this paper). The first basic data structure is *virtual disk mapper* (VDM), used to translate an offset within a virtual disk to its corresponding SubRAID and the VDB_in_SubRAID of that SubRAID. For each virtual disk, ERAID maintains a VDM, and every VDB in that virtual disk has a corresponding entry to record the mapping information: the VDB identifier, SubRAID identifier and VDB_in_SubRAID identifier that the VDB maps to. Another data structure is *SubRAID information table* (SIT), with each entry corresponding to one SubRAID, tracking the information that is similar to the information needed in traditional RAID system to translate logical addresses into physical addresses, including SubRAID_id, SubRAID configuration information (data placement policy, redundancy scheme, stripe unit size, stripe depth), and the physical parameters of the component disks in that SubRAID. Translating a client-supplied logical address <virtual_disk_id， offset_virtual_disk> into a particular disk offset <SubRAID_id， disk_id， offset_disk>occurs in four steps as follows.

(1) The client-supplied logical address is <virtual_disk_id，offset_virtual_disk>, so the given offset, offset_virtual_disk, residents in the VDB whose virtual disk

identifier= virtual_disk_id, and the VDB identifier VDB_id $=\lfloor \text{offset\_virtual\_disk}/64K \rfloor$.

(2) Through referring VDM of the specified virtual disk (virtual disk identifier= virtual_disk_id), the VDB_id is translated to a SubRAID_id and a VDB_in_SubRAID.

(3) The offset within the SubRAID (SubRAID identifier=SubRAID_id) for that client-supplied logical address, designated as offset_SubRAID, equals （VDB_in_SubRAID*64K + offset_virtual_disk%64K）.

(4) Using the information provided by SIT, ERAID translates the offset_SubRAID into a disk identifier (disk_id) and an offset within that disk (offset_disk). This procedure is greatly similar to the procedure of translating a logical address to a physical disk address in traditional RAID.

This multi-layer logical-to-physical address translation method minimizes the amount of information that the system must maintain – when user data in a virtual disk migrate from one SubRAID to another (the basic migration unit in ERAID is VDB/VDB_in_SubRAID), the system just need to update the information in one or several entries of the VDM corresponding to that virtual disk; Furthermore, through updating the SIT, ERAID system could also reflects the configuration alteration of its SubRAIDs without much effort.

3.3 Data migration among SubRAIDs

ERAID system keeps migrating data to the appropriate SubRAID according to the change of workload characteristics, such as request size, read/write ratio. Thus, from the view of a client, the virtual disk that ERAID system assigned to it can always ensure optimal I/O performance although the workload changes dynamically.

System gathers its workload statistics with the help of an *access log* (AL) to which all requests issued to ERAID driver are recorded. Each entry of AL contains four fields with regard to the information of each request: timestamp, virtual disk identifier, start sector, request size (the number of sectors) and read/write flag. As the AL becomes full, ERAID flushes the *access information table* (AIT) of every virtual disk. AIT is another table system maintains for each virtual disk. With each entry of AIT corresponding to each VDB in a virtual disk, tracing the access information to that specified VDB: VDB identifier (VDB_id), the average size of read accesses (average_read_size), the average size of write accesses (average_write_size), the number of read accesses (times_of_read), the number of write accesses (times_of_write), and the read/write ratio of the accesses (r/w_ratio); in this subsection, we use “accesses” to represent “accesses to a specified VDB” for literal concision.

The AL is initially empty, and the value of average_read_size, average_write_size, times_of_read, times_of_write, r/w_ratio in all AITs are uniformly set to 0. With the arrival of requests referencing ERAID system, the requests’ information are recorded in AL. Whenever the AL becomes full, the statistic information of access to each VDB in all AITs are updated according the following formulas.

$$times\_of\_read_{new} = times\_of\_read_{old} + \sum_{i=1}^{t\_read} read\_hit\_VDB_i \tag{1}$$

$$times\_of\_write_{new} = times\_of\_write_{old} + \sum_{j=1}^{t\_write} write\_hit\_VDB_j \tag{2}$$

$$average\_read\_size_{new} = \frac{average\_read\_size_{old} \times times\_of\_read_{old} + \sum_{i=1}^{t\_read} sectors\_in\_VDB\_of\_read_i}{times\_of\_read_{new}} \tag{3}$$

$$average\_write\_size_{new} = \frac{average\_write\_size_{old} \times times\_of\_read_{old} + \sum_{j=1}^{t\_write} sectors\_in\_VDB\_of\_write_j}{times\_of\_write_{new}} \tag{4}$$

$$r/w\_ratio = times\_of\_read_{new} / times\_of\_write_{new} \tag{5}$$

Where average_read_size$_{new}$, average_write_size$_{new}$, times_of_read$_{new}$, times_of_write$_{new}$ and average_read_size$_{old}$, average_write_size$_{old}$, times_of_read$_{old}$, times_of_write$_{old}$ are the average size of read accesses, the average size of write accesses, the number of read accesses, the number of write accesses after and before AIT update respectively. As for read accesses, t_read in formula (1) is the number of read accesses in AL. To a given VDB, when the *i*th read access in AL hits all or part of its sectors (64kB/VDB, 512byte/sector), the $read\_hit\_VDB_i$ of the VDB is set to 1, or else it is set to 0; so it is apparent that $\sum_{i=1}^{t\_read} read\_hit\_VDB_i$ is the number of read accesses in AL that hit the given VDB; in formula (3), $\sum_{i=1}^{t\_read} sectors\_in\_VDB\_of\_read_i$ is the accumulative total number of sectors in the given VDB that be referenced by all read accesses in AL.

To process data migration, another data structure *SubRAID block bitmap* (SBB) is used. Two fields are included in each entry of SBB: the first field records the SubRAID identifier (SubRAID_id); the second field is a bitmap of validate status of VDB_in_SubRAID, with each bit of the bitmap assigned for a VDB_in_SubRAID – if data of the VDB_in_SubRAID is validate, then the corresponding bit is set to 1, and if a bit in bitmap is set to 0, then it means the data in its corresponding VDB_in_SubRAID is invalidate or that VDB_in_SubRAID is unoccupied.

The basic unit of data migration in ERAID is VDB (it is denoted as VDB and VDB_in_SubRAID in virtual disk and in SubRAID respectively). The processing steps of VDB data migration is described as follows:

(1) Check the read/write ratio (r/w_ratio) of each VDB in all AITs in sequence, until all VDBs have been processed.

(2) If the r/w_ratio of a VDB is less than the threshold value (threshold $_{r/w}$) that system set beforehand, which means that the percentage of write accesses to the certain VDB exceeds an appointed value. So ERAID improves the system I/O performance through migrating the VDB data to enhance the VDB’s write performance.
(3) If the RAID level of SubRAID in which the above VDB reside is level1, there’s no need for VDB to migrate, the system continue to process the next VDB, and turn to step (1). But if the RAID level is level5, the VDB data should migrate to a SubRAID whose RAID level is level1, thus the key issue becomes how to choose the target SubRAID for the migrating VDB.
(4) Sort all level1 SubRAIDs in SIT by their stripe size (stripe_size=stripe_unit_size*stripe_depth) in descending order. Then scan these level1 SubRAIDs sequentially, to find a SubRAID whose stripe size is most closet to that of the SubRAID in which the migrating VDB reside in as the target SubRAID.
(5) Migrate the migrating VDB to an available VDB of the target SubRAID. System continues to process the next VDB, turn to step (1).
(6) If the r/w_ratio of a VDB is less than threshold $_{r/w}$, then the objective of migration becomes improving the VDB’s read performance. If the RAID level of SubRAID in which the above VDB reside is level5, there’s no need for VDB to migrate, and system continues to process the next VDB, turn to step (1). If the RAID level is level5, then
(7) Sort level5 SubRAIDs in SIT by stripe size similar to step (4), and choose the SubRAID with stripe size closest to that of the migrating VDB-reside SubRAID as the target SubRAID.
(8) Turn to step (5).

4 Simulation and results analysis

4.1 simulations

We use a trace driven simulation to investigate the effectiveness of the performance of the ERAID system. The simulator is disksim3.0 [15], a widely used, powerful storage subsystem simulator.

Table 1. Structural parameters of SubRAIDs in ERAID

SubRAID ID.	RAID level	No. Of disks	Stripe unit size（KB）
SubRAID0	Level1	4	16
SubRAID1	Level1	6	32
SubRAID2	Level5	4	8
SubRAID3	Level5	5	16
SubRAID4	Level5	6	16
SubRAID5	Level5	6	32
SubRAID6	Level5	8	8
SubRAID7	Level5	9	16

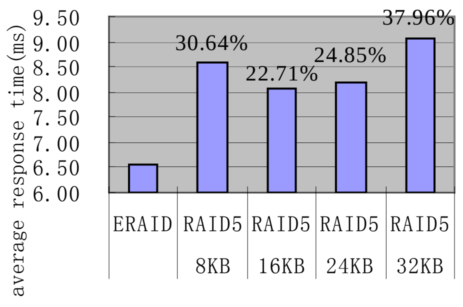
Table 2. Characteristic parameters of each workload in trace

Workload ID.	No. Of requests	read /write ratio	Average request size（KB）	Request size distribution
Workload 0	12800	0.43/0.57	8.90	(0.0，32.0)
Workload 1	12800	0.30/0.70	14.31	(0.0，96.0)
Workload 2	12400	0.66/0.34	8.31	(0.0，24.0)
Workload 3	12300	0.75/0.25	12.33	(0.0，64.0)
Workload 4	12000	0.70/0.30	21.29	(0.0，80.0)
Workload 5	12800	0.75/0.25	9.57	(0.0，80.0)
Workload 6	12400	0.66/0.34	12.49	(0.0，56.0)
Workload 7	12400	0.75/0.25	13.87	(0.0，128.0)

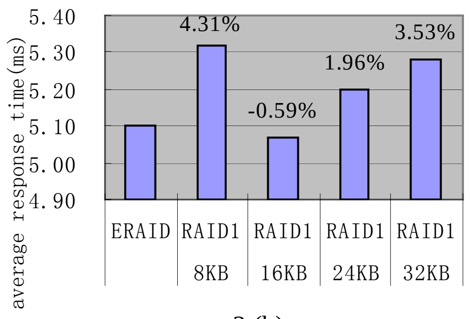
We simulated a mid-scale RAID storage system with 48 disks. The system controller has a 256MB of NVRAM whose data transfer rate is 62 MB/s. The system backplane bus connects 8 SCSI controllers each manage 6 HP Cheetah73 [16] disks of which rotation speed is 10000 RPM, formatted capacity is 73.4GB, and average data transfer rate is 31MB/s. Our objective is to compare the performance of mid-scale RAID applying ERAID architecture and those applying traditional RAID architectures. So we simulated 3 RAID systems, designated as ERAID, RAID5 and RAID1 respectively. System 1(ERAID) is ERAID system consisting of 8 SubRAIDs with their configuration parameters listed in table 1. System 2(RAID5) and system 3(RAID1) are traditional RAID systems, in which all disks are partitioned into *n* independent, disjoint units called LUNs (logical units), specifically, every *n* disks connected to the same controller constitute a LUN that could be viewed as a “large disk”; System 2 is a traditional RAID5 system with data striping among *48/n* LUNs, and system 3 is a traditional RAID1 system taking *48/n* LUNs as *48/n* disks.

The trace used in our simulation was composed of 7 workloads generated by the synthetic workload generator of the disksim3.0 simulator. Each synthetic workload represents one kind of client application, and its statistic characteristics are described in detail in table 2. For each workload, the number of requests distributes with its request size with an exponential distribution *(x, y)*, where *x* and *y* specify the base and mean value of exponential distribution respectively [15]. The request arrival interval is determined by exponential distribution (0.0, 10.0).

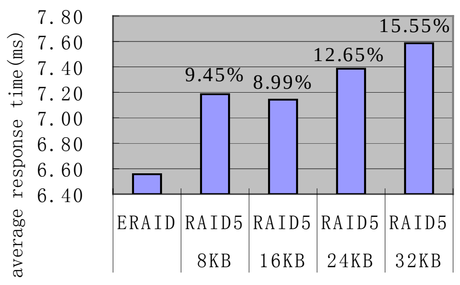
Under the above trace, we implemented 3 sets of simulations: the first simulation set is used to compare the I/O performance of ERAID with that of RAID5 and RAID1 system with $n = 4$, and with stripe unit size ranging from 8KB to 32KB. To eliminate the inaccuracy of the comparison results induced by unequal storage capacity of these storage systems, when compare the performance of ERAID and that of RAID5, we utilize 80% of storage capacity of each SubRAID in ERAID, while limit the utilization percentage of RAID5 system storage capacity to 70.5%; When compare the performance of ERAID and RAID1, the storage capacity utilization percentage of each SubRAID in ERAID and that of RAID1 system are limited to 51.9% and 80% respectively. Parameters of another two simulation sets are same with the first one, except that *n* is set to be *8* and *12* respectively.



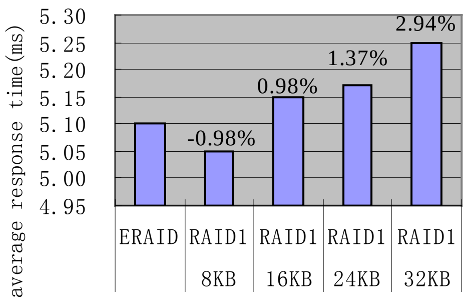
2 (a)



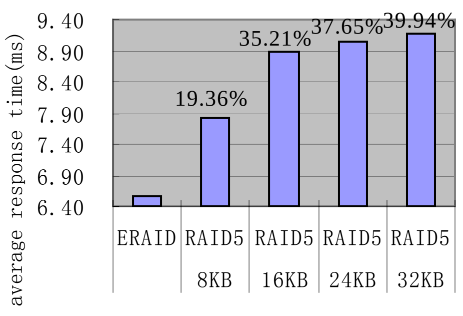
2 (b)



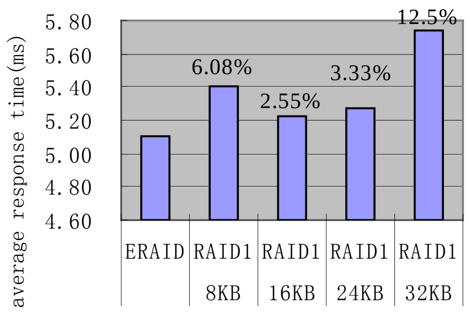
3 (a)



3 (b)



4 (a)



4 (b)

Fig. 2, 3, 4. Performance comparison of ERAID and traditional RAID5, RAID1 systems. Figure 2/3/4 (a) Mean request response time of ERAID (storage capacity utilization ratio of each SubRAID is 80%) and RAID5 (storage capacity utilization percentage is 70.5%, stripe unit size is 8KB, 16KB, 24KB and 32KB; *n*=4/8/12). Figure 2/3/4 (b) Mean request response time of ERAID system (storage capacity utilization percentage of each SubRAID is 51.9%) and RAID5 (storage capacity utilization percentage is 80%; stripe unit size ranges form 8KB to 32KB; *n*=4/ 8/12)

4.2 Simulation results and analysis

The comparison results of I/O performance of ERAID system with that of RAID5 system and of RAID1 system are shown in figure 2, 3, and 4. For the first simulation set, the absolute performance (in term of average response time) of ERAID system and that of RAID5 system recorded in figure 2(a), while figure2 (b) record the average response time of ERAID system and that of RAID1 system In addition, the relative performance improvements of ERAID over traditional RAID5 and over RAID1 system are also shown in figure2 (a) and figure2 (b). Figure3 and 4 records the results of the second, the third and the fourth simulation set, respectively.

A very similar behavior in figure2, 3, and 4 can be observed, so we only discuss figure 3 in detail. The explanation of figure 2 and 4 are also the same. We noticed from figure3 (a) that compared with RAID5 system, although ERAID system reduced the storage capacity by 11.9%, it improved average response time by 9.0% (compared with RAID5 with 16KB stripe unit) -15.5% (compared with RAID5 with 16KB stripe unit). Figure3 (b) showed no obvious performance difference between ERAID and RAID1 system — the average response time of ERAID system is only 1.0% higher than that of RAID1 with 8KB stripe unit, and only 1.0%-2.9% lower than that of RAID1 with stripe unit size ranging from 16KB to 32KB. But ERAID system is still worthwhile, for it gains the advantage over RAID1 by 35.1% in point of storage capacity. So we could drawn from the simulation results that: (1) compared with traditional RAID5 system, ERAID system we proposed in this paper has drastically enhanced I/O performance, through its level1-SubRAIDs servicing the small write requests to solve the “small write” problem of traditional RAID5. Meanwhile, the existence of level5-SubRAIDs minimizes the redundancy overhead of the whole storage system. (2) ERAID system has quite similar performance with traditional RAID1 system, while the former system has an obvious advantage on available storage capacity over the latter (RAID1 system’s available storage capacity is always 50% of all its physical disks’ space).

5 Conclusions

We propose a new-architecture storage system for mid/large scale RAID – ERAID. ERAID consists of a collection of different-configuration SubRAIDs that cooperatively implement a single, block-level storage system with a unique global storage space. ERAID assigns one virtual disk to serve each client. Clients Data could migrate dynamically to appropriate SubRAIDs according to the evolution of the workloads. Consequently, to each client, ERAID could always provide a virtual disk qualified to satisfy its I/O requirement. Our simulation results demonstrate that (1) at little acceptable overhead of storage space, ERAID system can improve I/O performance significantly compared with traditional RAID5 system. (2) With nearly the same I/O performance, ERAID system has much more storage capacity than that of the traditional RAID1 system.

References

1. Patterson, D. A., Chen, P., Gibson, G., Katz, R. H.: Introduction to Redundant Array of Independent Disk. 34th IEEE Computer Society International Conference, Intellectual Leverage, Digest of Papers (1989) 112 – 117
2. Elizabeth, V., Arif, M., Xu, J. Z., Qiu, X.Z.: An Integrated performance model of disk arrays. Proceedings of the 11th IEEE/ACM International symposium on Modeling, Analysis and Simulation of Computer Telecommunications System (MASCOTS'03) (2003)
3. Lee, E. K., Thekkath, C. A.: Petal: Distributed Virtual Disks. Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems, Cambridge, MA (1996) 84-92
4. Gabber, E., Korth, H. F.: Data Logging: A Method for Efficient Data Updates in Constantly Active RAIDs. Proceedings 4th ICDE (1998)
5. Mogi, K., Kitsuregawa, M.: Virtual Striping: A Storage Management Scheme with Dynamic Striping. IEICE Trans. Inf. & SYST., vol. E79-D (1996) 1086-1092
6. Wilkes, J., Golding, R., Staelin, C., Sullivan, T.: The HP AutoRAID hierarchical Storage System. Proceedings of the 15th ACM Symposium on Operating Systems Principles, Copper Mountain (1995) 96-108
7. Manoj, P., Mario, L.: CSAR: Cluster Storage with Adaptive Redundancy. Proceedings of the 2003 International Conference on Parallel Processing (2003)
8. Ligon, W. B., Ross, R. B.: An Overview of the Parallel Virtual File System. Proceedings of the 1999 Extreme Linux Workshop (1999)
9. Braam, P., et. al.: The Lustre Storage Arvhitecture. Cluster File Systems Inc. http://www.lustre.org/docs/lustre.pdf (2004)
10. Zhu, Y.: Design, Implementation, and Performance Evaluation of a Cost-Effective Fault-Tolerant Parallel Virtual File System. Proceedings of Int. Workshop on Storage Network Architecture and Parallel I/Os, IEEE Computer Society (2003) 53-64
11. Sobe, P.: Stable Checkpoing in Distributed Systems without Shared Disks. IPDPS 2003 Proceedings. Workshop on Fault-Tolerant Distributed Parallel and Distributed Systems. IEEE Computer Society (2003)
12. Hartman, J., Ousterhout, J.: The Zebra Striped Network File System. ACM Transactions on Computer System (1995)
13. Anderson, T., Dahlin, M., Neefe, J., Patterson, D., Roselli, D., Young, R.: Serverless Network File System. ACM Transactions on Computer Systems (1996)
14. Hartman, J. H., Murdock, I., Spalink, T.: The Swarm Scalable Storage System. Proceedings of the 19th International Conference on Distributed Computing Systems (1999)
15. Ganger, G. R., Patt, Y. N.: Using system-level models to evaluate I/O subsystems designs. IEEE Transactions on Computers (1998) 667-678.
16. Seagate Corporation. Cheetah 73 Family: ST173404LW/LWV/LC/LCV Product Manual, Vol. 1. http://www.seagate.com/support/disk/mannuals/scsi/83329478f.pdf