

# Performance and Reliability of Secondary Storage Systems

– Invited Paper –

Haruo Yokota  
Department of Computer Science  
Tokyo Institute of Technology  
yokota@cs.titech.ac.jp

## ABSTRACT

Hard disk drives (HDDs) are now capable of storing a large amount of data, and their storage capacities are still rapidly improving. From the performance and reliability point of view, a large capacity HDD, however, has problems. It is, therefore, desirable to improve the performance of a secondary storage system that uses a large number of small HDDs. To maintain a high reliability of such a system, it is essential to keep redundant data in the parallel small HDDs. This paper provides, firstly, an overview of several approaches for constructing reliable parallel disks, as well as their problems. Next, network disk configuration, such as network-attached storage (NAS) and storage area network (SAN) is discussed. Our approach of autonomous disks to provide an efficient secondary storage system is then introduced.

**Keywords:** Hard Disk Drives, Performance, Reliability, RAID, DR-net, SAN, NAS, Autonomous Disks

## 1. INTRODUCTION

Nowadays, the recording density of magnetic hard disk drives (HDDs) is being considerably improved [1]. In the 1980s, the rate of progress was 30 % per year. It changed to 60 % in the early 1990s, and became 100 % around 1998. Currently, it is said to be more than 100 %. Thus, it exceeds the ratio of Moore's Law for semiconductor products, in which the density is quadrupled every three years. In the near future, the recording density will be 100 Gbit/in<sup>2</sup>. It means that we will soon have a 100 GB HDD contained on only one 3.5-inch platter.

From the point of cost, the improvement of recording density is quite good for the many applications that require large amounts of storage, such as large databases, huge data warehouses, and audio-visual systems. Therefore, HDDs occupy a steady position in the secondary storage system.

However, large capacity HDDs also have problems, especially in relation to their performance and reliability. Improving the performance of a secondary storage system, using a large number of small HDDs, is quite desirable. To keep a high reliability of such a system, it is important to maintain redundant data in the parallel small HDDs.

The rest of the paper is organized in the following manner. The next section investigates the problems of a large capacity HDD. Then, in Section 3, an overview is given of parallel disk approaches for improving the performance and reliability of secondary storage systems. Thereafter, our approach of constructing efficient network disks is discussed. Section 4.

## 2. PROBLEMS OF A LARGE HDD

### Problem of Performance

There is an intrinsic limitation in improving the performance of a HDD, because of the physical movement required for accessing its stored data. At first, it has to seek a track storing the objective data. Then it has to wait for rotation until the objective sector comes under its reading head. The search time and rotational-wait time are shortened by technological improvements; the search time becomes less than 1 ms for neighboring tracks, and average rotational-wait time becomes 2 ms with faster rotation. It is difficult to achieve more substantial improvement, so we currently need at least 3 ms to access the required data. This access latency is quite long when compared to the data processing time in a CPU.

Although data transfer rate is dependent on recording density and rotational speed, the data transfer time for small amounts of data is negligible compared with the access latency. For example, the data transfer rate of a current high performance HDD is about 25 MB/s, so that the data transfer time for a sector (256B) is only 0.01 ms, which is within error range of the access latency.

Here, we consider access performance for a capacity unit (GB). If the capacity of a HDD is 10 GB, possible access performance per 1 GB is 33.3 access/s/GB ( $(1/0.003)/10$ ). It becomes 3.3 access/s/GB, if the capacity is increased to 100 GB. Hence, the access performance per capacity unit decreases with the increase of the capacity of the HDD. Is this performance good enough for current sophisticated applications?

When only small capacity HDDs were available, multiple HDDs were used to meet the needed storage capacity. The parallel configuration could hide the access latency and improve performance of the secondary storage. However, the large capacity HDD makes the parallel configuration unnecessary for

satisfying the needed storage capacity. At the same time, the access latency can no longer be hidden.

It would be clearer to consider a multiple-user environment. If the location of data for each user is distributed over each HDD in the parallel configuration, they can simultaneously access their own data. However, a large capacity HDD sequentializes these accesses. Consequently, the tendency of increasing HDD capacity does not correspondingly improve application performance.

### Problem of Reliability

A large capacity HDD also has another problem, related to data reliability.

Mean time to failure (MTTF) is usually used as a criterion for the system reliability. The MTTF of a HDD has been improved by hardware technology, and currently is specified at 1,000,000 hours. This suggests that a HDD probably operates more than 100 years before any failure. However, this is of course a probability, and it is not possible to have zero probability because of the physical movement of a HDD. Therefore, there will be HDD failures with any large batch of products.

The impact of a failure of a HDD is serious because of its characteristics. Hardware failure in a CD or MT device does not cause loss of its stored data, whereas important data stored in a HDD is lost by any type of failure: in its motor, controller, head, media, and so on. Therefore, increasing the capacity of a HDD makes the impact of its failure more serious.

We then consider MTTF per bit. When MTTF of a 10 GB HDD is 1,000,000 hours, its MTTF/bit is about 45 ms. Even when the capacity of a HDD becomes 100 GB, its MTTF will not change. Thus, MTTF/bit for a 100 GB HDD is 4.5 ms. Note that it does not mean that any bit in a particular HDD is lost in 4.5 ms, but that the average age of a bit in a large number of 100 GB HDDs is 4.5 ms, since the influence of a failure in a large capacity HDD is serious.

One of popular methods for protecting data against a HDD failure is its backup to a third storage device such as a MT. However, a large capacity HDD takes considerable time to backup. Using an advanced intelligent tape (AIT) interface for MTs, it takes about two and half hours to fully backup a 100 GB HDD, as the data transfer rate of AIT-3 is 12 MB/s.

## 3. PARALLEL DISKS

From observations of the previous section, we can say that it is better to use a large number of small HDDs to gain the necessary performance. Thus, high performance data processing applications do not require such a large capacity HDD, while a large HDD should still be suitable for audio-visual applications. If the capacity of a HDD is limited, the improvement of recording density reduces the number of tracks in the HDD; that is, it reduces head-search time. This is also desirable for access performance.

However, an increase in the number of HDDs within a secondary storage system lowers reliability of the system. To lift

reliability of a storage system containing multiple HDDs, a redundant array of inexpensive disks (RAIDs), has been proposed [2]. The RAIDs were originally supposed to be constructed from multiple cheap HDDs.

### RAIDs

A RAID system stores data and redundant information with a number of HDDs, and recovers the data in a damaged HDD by using the redundant information. In particular, the RAID level-5 configuration, using parity codes as the redundant information to recover lost data, enables a cost-effective construction with high reliability and good read performance for a small system.

The MTTF of a RAID5 system can be derived by the following formula[2]:

$$MTTF_{RAID5} = \frac{(MTTF_{HDD})^2}{N \times (N - 1) \times MTTR_{HDD}}$$

where  $MTTF_{HDD}$  and  $MTTR_{HDD}$  are the mean time to failure and mean time to repair of a single HDD used in the RAID5, respectively, and  $N$  is the total number of HDDs in the system. The formula indicates that short MTTR of the HDDs gives the system a high reliability.

However, a RAID5 system still has problems with its scalability. The previous formula indicates that the MTTF of RAID5 is inversely proportional to the square of the number of HDDs. Moreover, it is difficult to keep the MTTR short for a large scale RAID system. Therefore, the MTTF of the RAID5 system with a large number of HDDs should be shorter.

Furthermore, the parity calculation technique used in RAID5 makes the throughput for small write operations four times slower than read operations. In the next two sections, we will describe approaches to try to solve these problems.

### Scalability of Parallel Disks

Several problems related to scalability occur for RAIDs. A single controller in a RAID5 system becomes a bottleneck when the system contains a large number of HDDs. Moreover, it is obvious that the controller is heavily used when data stored in damaged disks is reconstructed, and the access speed of RAIDs is particularly reduced in this case.

Reliability of RAIDs with a large number of HDDs is another problem. Since a single fault is assumed in a parity group of a RAID, data is lost when at least two disks in the same parity group are simultaneously damaged. This is not appropriate when the system size becomes large. Reliability of a single controller for the large system is also another problem.

There are several approaches to solve some parts of the problem. RAID level-6 uses Reed-Solomon codes to protect against failure of up to two disks by having two redundant HDDs [3]. Blaum et al. also propose EVEN-ODD methods to treat two-HDD failures [4]. In those methods, however, bottlenecks of the controller and locality of communication among disks are not considered, although both methods increase the load on the controller.

Applying the parity technique of RAID5 to interconnection networks is proposed for solving these problems [5, 6, 7]. We

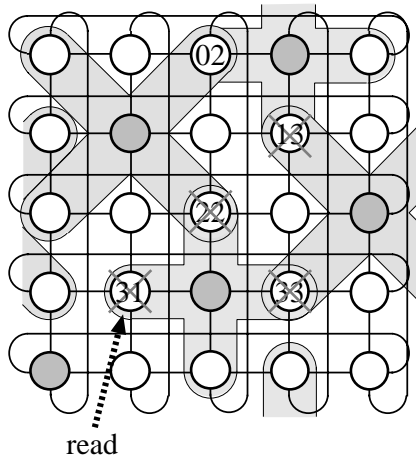


Figure 1: A reconstruction pattern in a  $5 \times 5$  torus DR-net

named them *data-reconstruction networks* or *DR-nets* for short. In the DR-nets, HDDs are connected to each node of an interconnection network, and a parity group is constructed in a sub-network of the interconnection network. Functions for controlling the system are distributed to each node that can behave autonomously. Therefore, there is no central controller in a DR-net.

Since communication of a parity group is kept local within the sub-network for reconstructing data under a HDD failure, performance degradation for the reconstruction in a large network becomes very small. DR-nets also achieve high reliability, capable of handling multiple faults by using two types of parity groups overlapped on the interconnection network. Figure 1 shows an example of recovering four disk failures in a  $5 \times 5$  torus DR-net by combinations of the two types of parity groups. It recovers 100 % data for two-HDD failures and more than 90 % data for four-HDD failures. It can recover, at most, nine failures for 25 HDDs [8].

We have been developing a small experimental system of a  $5 \times 5$  torus DR-net using Transputers [6, 7]. A Transputer, T805, corresponds to each node of the DR-net, and is connected with a small HDD via a SCSI controller. Experimentation results using the system indicate that the DR-net provides good performance under multiple HDD failures [8].

A group in UC Berkeley has proposed a system in which multiple RAID systems with a bus are connected by a network to achieve scalability [9]. However, it differs from applying parity groups to interconnection networks directly and the controller bottleneck still remains. Since it assumes a single fault in a parity group, reliability for a large configuration is not considered.

The DataMesh project [10], that is similar to the experimental system of the DR-nets, used Transputers as controllers of a parallel disk system to increase I/O performance, but they do not provide high reliability. They then proposed TickerTAIP architecture [11] as an enhancement of DataMesh to adopt the RAID parity technique in tolerating a single fault. It distributes

functions for controlling the system like the DR-nets, but cannot tolerate multiple faults.

For scalability, the DR-net removes the controller bottleneck by distributing controlling functions to each autonomous disk node, and provides high availability by overlapping two types of parity groups and distributed autonomous controllers. However, it still has the problem of small write performance inherited from RAID5. In the next section, methods for improving access performance for small write operations, is considered.

## Write Access Performance

Parity calculation techniques introduced to RAID5 require four disk accesses for each write request: read the old data, write the new data, read the old parity, and write the new parity. On the other hand, each read request requires only one disk access in the appropriate HDD. Therefore, the write performance of a RAID5, or other redundant disk arrays like the DR-nets, is poor when compared with its read performance.

There are several approaches for improving small write access performance of a secondary storage system using the RAID5 parity technique. The parity logging [12] and virtual striping [13] methods were proposed to improve write performance, by updating bulk parity. AutoRAID [14] and Hot-Mirroring [15] divide the disk space into two areas: RAID5 and mirroring (RAID1). The RAID1 area is used to store “hot” data that is frequently accessed, while data that has not been accessed for a relatively long time is moved to the RAID5 area to save disk space. Thus, the RAID1 area is used as a caching area. Since control mechanisms for them are rather complicated, it is difficult to make the controller more reliable than the ordinary RAID5 controller.

On the other hand, semiconductor memories are commonly placed between a RAID5 and its host as a disk cache to improve its write performance [16]. As high hit ratios are necessary to make a caching mechanism effective, it is better for the cache to be as large as possible. However, large semiconductor memories make the system significantly more expensive. Moreover, non-volatility is essential for the cache, taking into account the applications of RAID systems, and this adds to the cost of the system.

Hu and Yang proposed a disk caching disk (DCD) architecture to reduce costs by using disks as disk caches instead of using semiconductor memory [17]. They adopted the log-structured file system (LFS) method [18] that converts small random writes into a large sequential write to improve the write performance. The DCD architecture enables a cost-effective system configuration without modifying the RAID and its host, but it still has several problems. Firstly, the LFS approach needs garbage collection phases to reuse disk space. Garbage collection prevents the continuous use of DCD systems, such as 24-hour OLTP applications. Secondly, read operations during the log-style write operations move the position of the disk heads from the tail of the log. They increase both track search time and rotation latency. Thirdly, DCD requires a large amount of non-volatile memory. Information about page locations in DCD must be stored and cannot be lost if the LFS is to be managed. If the location information is lost, later data cannot be recon-

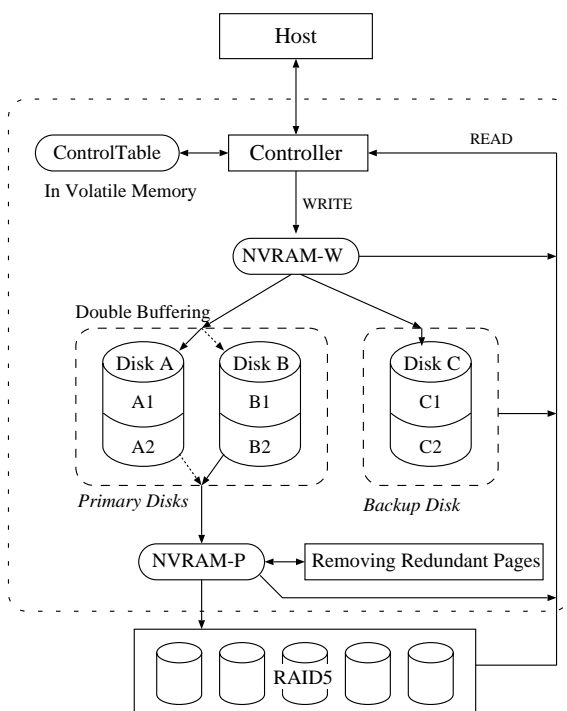


Figure 2: A system architecture for the FBD

structured. Finally, duplication of the DCD to tolerate a failure of the caching disk is impractical, because a tight synchronization would be required to maintain identical states of the duplicated caches.

One of the most important requirements for the mechanism to improve high reliable disk arrays is to maintain reliability while improving its performance. For that purpose, we have proposed *Fault-tolerant Buffering Disks*, or *FBD* for short [19]. It uses 2-4 disks clustered into two groups: primary and backup (Figure 2). Write performance is improved by sequential access for the primary disks without interruption, and by packing to reduce irrelevant disk accesses for the RAID. Buffering instead of LFS caching does not require garbage collection. Thus, FBD can be used for continuous applications without interruption. The backup disks are used to tolerate a disk failure in the FBD and to accept read requests for data stored in the buffering system, so as not to disturb sequential access in the primary disks. The FBD also uses semiconductor memories to balance data transfer speeds and remove redundant pages, but the amount of non-volatile memory is small, and most of the information for controlling the FBD can be located in volatile memories.

We developed an experimental system using an off-the-shelf personal computer and disks, and a commercial RAID5 system. Performance evaluation of the system demonstrates that the FBD considerably improves both throughput and response time [19].

Generally speaking, we can say that the concept of FBD is effective for a small, reliable disk array using the RAID5 par-

ity calculation technique. However, the problem of scalability reappears for the FBD. Of course, we can use multiple FBDs for large systems, but the control mechanism to maintain coherence among them will be complicated and reduce its performance.

Moreover, as already discussed in the previous section, the bit-price ratio of a HDD becomes very low, and the space utilization provided by the RAID5 parity calculation technique becomes less important. In that sense, mirroring or data replication, having no disadvantage in write performance, is then not so expensive.

At the same time, improvement of the local area network (LAN) technology or storage-device interface makes a configuration of network disks more feasible. In the next section, network disks based on these technologies are discussed.

## 4. NETWORK DISKS

### NAS and SAN

Nowadays, *network-attached storage (NAS)* and *storage area network (SAN)* [20] are attracting a great deal of attention as a means of providing scalable secondary storage environments. NASs are directly connected to an IP network as shared multiple hosts connected to a LAN, while a SAN constructs a dedicated network apart from a LAN, using a serial interface of storages such as Fiber Channels. These are approaches to construct storage-centric architecture by commodity components. There is also research for providing a filing system on network disks; Zebra [21] and GFS [22], are examples.

In these configurations, HDDs are passive devices, i.e., all their behaviors are controlled by their hosts via the network. Therefore, communications between HDDs and their hosts are quite frequent, restricting their performance. To make the storage system efficient, the location of data and its duplication is very important. A dedicated host is needed to manage data location and to control its access. The host will create a bottleneck as the system becomes large, as with the controller of a RAID system.

Moreover, reliability of storage depends on the reliability of the host and management of its software. It also depends on the operations for data maintenance. Reliability of software and operations are low in comparison with hardware reliability. This is dependant on the base reliability of HDDs because the system is a series configuration of reliability. Consequently, the system reliability is directly decreased by the component having the least reliability.

### Autonomous Disks

On the other hand, a HDD contains its own controller. If it is sufficiently intelligent to maintain the data, reliability of the storage system is not affected by the host, nor its operation. Moreover, it reduces the count in communicating with its hosts, and provides high scalability and flexibility. For this purpose, we have proposed a concept of *autonomous disks* in the network environment [23].

A set of autonomous disks is configured as a cluster in a network, and data is distributed within the cluster to be accessed

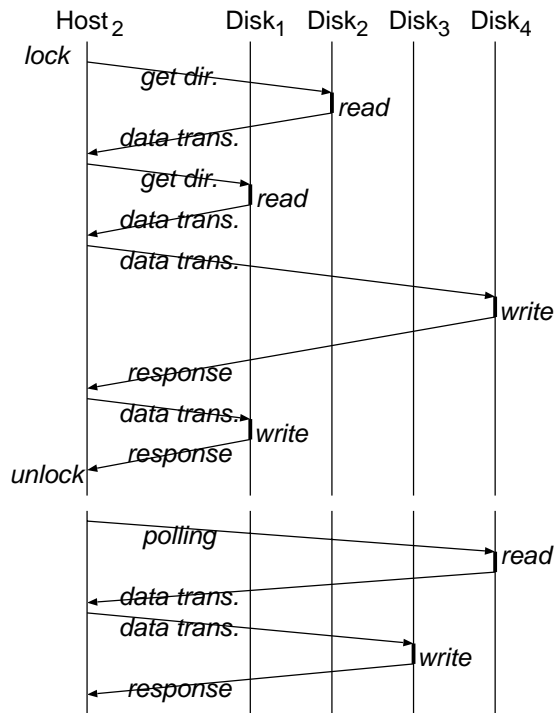


Figure 3: Communication Structure using Ordinary Disks

uniformly. The disks accept simultaneous access from multiple hosts via the network, and handle data distribution and load skews. They are also able to tolerate disk failures and software errors in disk controllers, and can reconfigure the cluster after the damaged disks are repaired. The data distribution, skew handling, and fault tolerance are completely transparent to their hosts. Thus, hosts are not involved in communication among disks to achieve these functions. This is the factor that provides high scalability of the system.

If we consider a situation in which each ordinary HDD has a distributed directory to accept access for a request and to redirect the request to a HDD storing the target data in the cluster, hosts have to traverse the directory and find the target HDD. To make the system reliable, an update log should be stored physically in another HDD, and should asynchronously be applied to backup HDDs. These sequences are also controlled by the hosts. An example of a communication structure for this situation is depicted in Figure 3.

If we use autonomous disks, the whole process is independent of the host except for the first command submission and the receipt of the response. The communication structure of the same situation, using autonomous disks, is depicted in Figure 4. These two figures also demonstrate that the total communication on the network is reduced with autonomous disks.

We propose the use of rules and distributed directories to implement these functions. By adopting rules, it is easy to vary strategies for data distribution and fault tolerance. For example, we can easily change the number of backup disks and log disks.

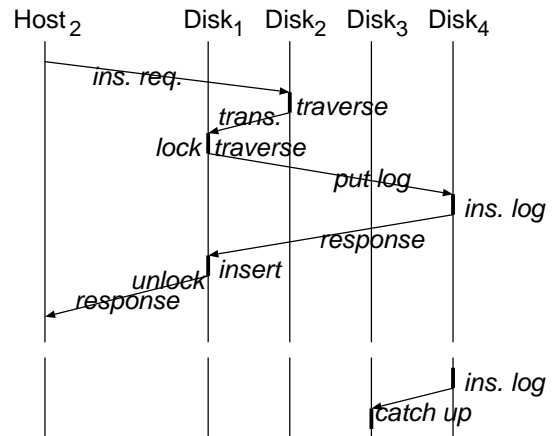


Figure 4: Communication Structure using Autonomous Disks

There are other approaches to utilizing the disk resident controller: the IDISK project at UC Berkeley [24] and the Active Disk projects at Carnegie Mellon [25] and UC Santa Barbara/Maryland [26]. They all have the goal of executing application programs both on HDDs and in their hosts. In other words, they focus on functions and mechanisms for user applications, such as decision support systems, relational database operations, data mining, and image processing. They do not consider management and reliability of data.

## 5. CONCLUDING REMARKS

We considered several aspects of large-scale secondary storage systems. Current high recording density HDDs have problems of performance and reliability. Redundant disk arrays with small size disks such as RAID5 are one solution for the problem. However, the ordinary RAID5 configuration has a weakness with its scalability.

We propose the data-reconstruction networks (DR-nets) architecture for scalable, reliable secondary storage. Autonomy of controllers in the DR-nets is the key concept for achieving scalability. The DR-net also provides high reliability, even for a large configuration, by applying the enhanced parity calculation technique of RAID5. However, the parity calculation technique creates another weakness. It makes performance for small write operations considerably poorer than for its read operations.

We have therefore proposed a fault-tolerant buffering disk (FBD) method for improving the small write performance, while keeping reliability high and the cost of the system low. Although the FBD enables a cost-effective configuration, improving both read and write performance, it again creates the problem of scalability.

On the other hand, disks with high recording densities as first described, have become quite inexpensive. Therefore, we can use a simple replication technique instead of the parity calculation technique, thereby reducing the required storage space. We need not consider the cost of storage, but still must consider the

scalability and management of the storage system.

Recently, network-attached storage (NAS) and storage area network (SAN), have attracted a great deal of attention for storage-centric system configuration. They use off-the-shelf network technology to provide enough scalability of the storage system. In these configurations, however, frequent communication with the hosts is a key performance factor. Furthermore, we have to consider the reliability of the system and flexibility of its management.

To satisfy these requirements, autonomous disks are proposed. They inherit autonomy from the DR-net, but are based on replication instead of parity calculation to maintain reliability, and are more flexible by using rules. We are now implementing an experimental system, using PCs connected by a network, to evaluate the mechanism.

## References

- [1] SRC (Storage Research Consortium in Japan). Annual report of SRC, Nov. 1999.
- [2] D. A. Patterson, G. Gibson, and R. H. Katz. A Case for Redundant Arrays of Inexpensive Disks(RAID). In *Proc. of ACM SIGMOD Conference*, pages 109–116, Jun 1988.
- [3] Peter M. Chen et al. RAID: High-Performance, Reliable Secondary Storage. *ACM Computing Surveys*, 26(2):145 – 185, Jun 1994.
- [4] Mario Blaum, Jim Brady, Jehoshua Bruck, and Jai Menon. EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures. *IEEE Transactions on Computer*, 44(2):192 – 202, Feb 1995.
- [5] Haruo Yokota. DR-nets: Data-Reconstruction Networks for Highly Reliable Parallel-Disk Systems. *ACM Computer Architecture News*, 22(4), September 1994. (Also in Proc. of 2nd Workshop on I/O in Parallel Computer Systems, pp. 105 – 116).
- [6] S. Tomonaga and H. Yokota. An Implementation of a Highly Reliable Parallel-Disk System using Transputers. In *Proc. of the 6th Transputer/Occam Intn'l Conf.* IOS Press, June 1994.
- [7] Haruo Yokota and Seishi Tomonaga. The Performance of a Highly Reliable Parallel Disk System. In A. De Gloria, M. R. Jane, and D. Marini, editors, *Proc. of the World Transputer Congress '94*, pages 147–160. The Transputer Consortium, IOS Press, September 1994.
- [8] Haruo Yokota and Yasuyuki Mimatsu. A Scalable Disk System with Data Reconstruction Functions. In Ravi Jain, John Werth, and James C. Browne, editors, *Input/Output in Parallel and Distributed Computer Systems, Chapter 16*. Kluwer Academic Publishers, June 1996.
- [9] A. L. Drapeau, K. W. Shirriff, and J. H. Hartmann. RAID-II: A High-Bandwidth Network File Server. In *Proc. of the 21st ISCA*, pages 234–244, 1994.
- [10] J. Wilkes. The DataMech research project. In P. Welch et al., editor, *Transputing '91*, pages 547 – 553. IOS Press, 1991.
- [11] P. Cao, S. B. Lim, S. Venkataraman, and J. Wilkes. The TickerTAIP parallel RAID architecture. In *Proc. of the 20th ISCA*, pages 52 – 63, 1993.
- [12] Daniel Stodolsky, Mark Holland, William V. Courtright II, and Garth A. Gibson. Parity-Logging Disk Arrays. *ACM Transactions on Computer Systems*, 12(3):206–235, August 1994.
- [13] Kazuhiko Mogi and Masaru Kitsuregawa. Virtual Striping: A Storage Management Scheme with Dynamic Striping. *IEICE Transactions on Information and Systems*, E79-D(8):1086–1092, 1996.
- [14] John Wikes, Richard Golding, Carl Staelin, and Tim Sullivan. The HP AutoRAID hierarchical storage system. In *Proc. of SIGOPS Conf. '95*, pages 96–108, 1995.
- [15] Kazuhiko Mogi and Masaru Kitsuregawa. Hot mirroring: A method of hiding parity update penalty and degradation during rebuilds for RAID . In *Proc. of SIGMOD Conf. '96*, pages 183–194, 1996.
- [16] Jai Menon. Performance of RIAD5 Disk Arrays with Read and Write Caching. *Distributed and Parallel Databases*, (2):261–293, 1994.
- [17] Yiming Hu and Qing Yang. DCD – Disk Caching Disk: A New Approach for Boosting I/O. In *Proc. of Int. Sympto. on Comp. Arch. '96*, 1996.
- [18] M. Rosenblum and J. Outerhout. The Design and Implementation of a Log-Structured File System. *ACM Transactions on Computer Systems*, 10(1):26–52, 1992.
- [19] Haruo Yokota and Masanori Goto. Fbd: a fault-tolerant buffering disk system for improving write performance of raid5 systems. In *Proc. of PRDC'99*, pages 95–102. IEEE-CS, 1999.
- [20] Barry Phillips. Have Storage Area Networks Come of Age? *IEEE Computer*, 31(7):10–12, 1998.
- [21] John H. Hartman and John K. Ousterhout. The Zebra Striped Network File System. *ACM Trans. on Comp. Sys.*, 13(3):274–310, Aug 1995.
- [22] Kenneth W. Prelan and et al. A 64-bit, Shared Disk File System for Linux. In *Proc. on 7th Mass Storage Systems and Technologies*, March 1999.
- [23] Haruo Yokota. Autonomous Disks for Advanced Database Applications. In *Proc. of International Symposium on Database Applications in Non-Traditional Environments (DANTE'99)*, pages 441–448, Nov. 1999.
- [24] Kimberly Keeton, David A. Patterson, and Joseph M. Hellerstein. A Case for Intelligent Disks (IDISKS). *SIGMOD Record*, 27(3):42–52, Sep. 1998.
- [25] Erik Riedel, Garth Gibson, and Christos Faloutsos. Active Storage for Large-Scale Data Mining and Multimedia. In *Proc. of the 24th VLDB Conf.*, pages 62–73, 1998.
- [26] Anurag Acharya, Mustafa Uysal, and Joel Saltz. Active Disks: Programming Model, Algorithms and Evaluation. In *Proc. of the 8th ASPLOS Conf.*, Oct. 1998.