論文 / 著書情報
Article / Book Information

Title	A Performance Comparison between the DR-net and a Hierarchical RAID System
Author	Yasuyuki Mimatsu, Haruo Yokota
Journal/Book name	Proc. of Pacific Rim International Symposium on Dependable Computing 2000 (PRDC 2000), Vol. , No. , pp. 193-200
Issue date	2000, 12
DOI	http://dx.doi.org/10.1109/PRDC.2000.897302
URL	http://www.ieee.org/index.html
Copyright	(c)2000 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other users, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works for resale or redistribution to servers or lists, or reuse of any copyrighted components of this work in other works.
Note	このファイルは著者（最終）版です。 This file is author (final) version.

A Performance Comparison between the DR-net and a Hierarchical RAID System

Yasuyuki Mimatsu*
Hitachi, Ltd.
Systems Development Laboratory
mimatsu@sdl.hitachi.co.jp

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

Abstract

When the number of disks rises in disk array systems that contain multiple disk drives, system performance is limited by a bottleneck at a centralized controller and/or at a communication path that uses a bus.

We evaluate, through simulation, a scalable architecture called DR-net, in which the controller functions are distributed to all disk drives and each disk has autonomy in processing its tasks. DR-net provided high throughput in proportion to the number of disks. In a conventional system, the influence of the bus setup time and concentration of the parity calculation load caused the throughput to saturate.

We also show that DR-net can take advantage of disk autonomy when reconstructing data stored in failed disks. Our results indicate that the distribution of functions and the autonomy of disk drives enable better scalability and more effective utilization of system resources than with a hierarchical system.

1 Introduction

In recent years, high performance disk systems have been required to narrow a performance gap between processors and secondary storage devices. Also, large capacity is also needed to store very large data sets such as multimedia data or contents in WWW servers, very large database systems, and so on. To provide large capacity and high performance, a scalable storage-system architecture is needed.

A redundant array of inexpensive disks (RAID)[8] consists of many disks to enable high performance and large capacity. Also, a RAID maintains redundant information to ensure high reliability. Many commercial RAID products now are available.

As the number of disks grows, though, a system is expected to concurrently process many requests. This can lead to problems in a conventional level-5 RAID. First, disk accesses are distributed among many disk drives, but the communication between an array controller and the disk drives is concentrated on a bus. If the bus bandwidth is not wide enough, the performance of the whole system will be limited. Second, the probability of multiple disk failures in a RAID5 increases as the number of disks rises. If there are many disks in a system or the reliability of the individual disks is not very high, multiple disks will fail and data will be lost. Third, when many access requests are processed concurrently in an array controller, the system performance may be limited by the CPU performance even if the bus is fast enough.

We have proposed a disk system architecture called DR-net(Data Reconstruction Network)[11][13][16][17]. In DR-net, all disk drives and interfaces to an external environment are connected by an interconnection network. This architecture provides system scalability by eliminating the central communication path typified by the bus in RAID5 or the array controller. The functions of an array controller, e.g., the maintenance of redundant data, data caching, and data reconstruction to recover from disk failures, are distributed to all disk drives. Each disk drive processes these functions, acting as a 'disk node', with its own CPU and memory in a disk controller. DR-net also provides fault tolerance to multiple disk failures.

A great deal of work related to function distribution in a disk system has been reported. In [1][5][10], Active Disk, a successor of Network-attached secure disks (NASD)[6][9], is discussed. This approach aims to shift part of the application programs to disk drives and to reduce the data transfer between an application server and disk drives. IDISK[7] has a general purpose CPU, a memory capacity of tens of megabytes and gigabit communication links in the disk drives. IDISK is designed for decision support systems (DDSs), but a disk array with a number of IDISKs can be made. In such a case, each IDISK will have the functions of a conventional array controller. TickerTAIP/DataMesh[2][12] has multiple controllers which are connected to each other by an internal

*This research was studied when the author was enrolled in Japan Advanced Institute of Science and Technology.

network. In [2], it was shown that the system performance is improved by distribution of the parity calculation.

DR-net differs from these approaches, though, in that it is an architecture where the system scalability, considering not only performance but also reliability, is given a high priority. When the system becomes large and the number of components increased, it is necessary to tolerate multiple failures. DR-net architecture provides tolerance to double disk failures with its parity data management described below. Moreover, its symmetrical and uniform structure provides redundant communication paths between disk drives when communication failures occur.

We expect DR-net to take advantage of function distribution when the number of disks is large because the communication and controller load are not concentrated on specific system components and are distributed over the whole system. We also expect DR-net to use its system resources effectively because each disk node can precisely manage and use its disk and communication links precisely with local information about them, i.e., busy or idle, access schedule, and so on. Furthermore, each disk node processes its tasks more parallelly because the autonomous disk node can process its tasks without waiting for directions from a remote centralized controller.

In this paper, we show the effect of function distribution and disk drive autonomy through simulations. Particularly, we evaluated the read and write access throughput when the system is scaled up with high reliability, in comparison to a conventional disk system that is hierarchically structured with multiple buses.

This paper is organized as follows: in the next section, the structure and behavior of DR-net is briefly described. In section 3, models of disk systems used in the simulation are described. Results of the simulation experiments are discussed in section 4.

2 Overview of DR-net

DR-net consists of disk nodes, interface nodes, and an interconnection network. It eliminates the performance bottleneck that exists in RAID5 by connecting nodes with an interconnection network instead of a bus, and by distributing array controller functions to disk drives. In this section, we describe the structure and behavior of DR-net.

2.1 Disk Nodes

A disk node consists of a disk drive and a disk controller. The controller has its own CPU and memory so that it is not only able to process disk accesses but can also calculate parity to maintain redundant data and communicate with other nodes via communication links. When a disk node receives a request from an interface node or another disk node, it accesses its disk drives to read or write data. If the drive fails, it reconstructs data or updates parity cooperating neighboring nodes through internode communication.

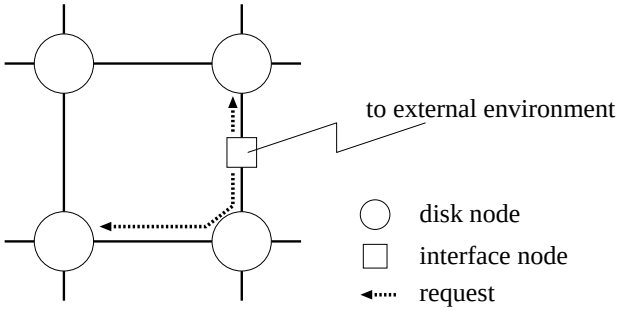

Figure 1. Interface node in DR-net

2.2 Interface Nodes

An interface node is also connected to the network to send access requests from the external environment to disk nodes and return results from the nodes. The number of interface nodes in DR-net is unlimited because they can be placed at any position in the network. For example, if we use a mesh or torus network to connect disk nodes, the interface node can be placed at any link between the disk nodes (Figure 1). Thus, DR-net enables communication throughout the network and provides a wide bandwidth to the external environment through many interface nodes.

2.3 Parity Groups

A parity group is a unit that consists of a number of disk nodes and maintains redundant parity data. In a parity group, one disk node (*a parity node*) contains a parity block in its disk drive and the other nodes contain data blocks (*data nodes*). When parity management is needed in a parity group, data nodes send related data to a parity node and the parity node calculates the parity. The location of a parity node in a parity group can be switched just as the location of a parity block in RAID5 is switched in each parity stripe; thus, parity blocks are distributed to all disks to avoid load concentration during parity updating[11][13].

DR-net can conceptually use any kind of network topology, but a two-dimensional torus network is especially appropriate for mapping compact parity groups and to keep the locality of communication in a parity group. We focus on the 2D-torus network in this paper.

The structure and mapping of parity groups in the torus network are shown in Figures 2 and 3[13]. Each parity group consists of five disk nodes. Groups in Figure 2 are referred to as *FPGs (first parity groups)*, and the groups in Figure 3

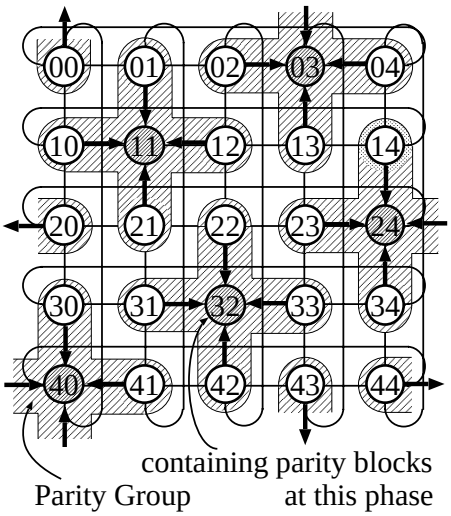

Figure 2. First parity groups (FPGs)

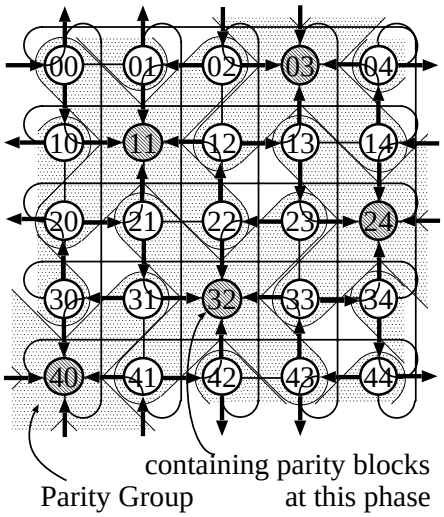

Figure 3. Second parity groups (SPGs)

are referred to as *SPGs (second parity groups)*. Each parity group maintains a parity block so that it can tolerate one disk failure in the group. SPGs are used to provide tolerance to multiple disk failures in an FPG. An SPG consists of five nodes which each belong to different FPGs. DR-net provides higher reliability than RAID6 or a mirroring system with these parity groups[16]. Furthermore, it enables tolerance to any single communication failures because its flat and uniform structure provides alternative communication paths between any two nodes.

When a data block in a disk node is updated, two parity blocks must also be updated: one in an FPG and one in an SPG. A new parity block is generated by a read-modify-write operation. The shapes of the parity groups in Figures 2 and 3 were designed to maintain the locality of the communications.

3 Modeling of Disk Systems

To examine the effects of distributed functions and disk node autonomy, we evaluated DR-net, comparing it to a conventional disk system that has a hierarchical structure with multiple buses, through simulation. In this section, we describe how we modeled the evaluated systems and the parameters that we used in the simulation experiments.

3.1 Hierarchical Disk System

Many disk array systems, e.g., RAID-II[4], use a hierarchical structure to take system scalability into account. Figure 4 depicts the modeling structure of the system we evaluated in our experiments. It consists of multiple RAID subsystems and can be scaled up by increasing the number of subsystems.

3.1.1 Interfaces to an External Environment

An interface receives read and write requests from an external environment, sends them to array controllers via a front-end bus, receives the results, and sends the results back to the external environment.

In our simulations, after an interface sent a disk access request to an array controller, it immediately generated another request and tried to send it also. When an interface tried to send an disk access request to an array controller and the controller had no buffer space to queue the request, the interface was suspended until the request could be queued. It was also suspended if the front-end bus was being used for other communication.

The destination of each disk access request generated by an interface was randomly selected. All accesses were a read or write of a 64-KB block.

3.1.2 Buses

The system has three kinds of bus. The front-end bus is the only bus that connects all interfaces and all array controllers. An intermediate bus connects an array controller to the controllers of back-end buses. A back-end bus connects to a number of disks. When an access request generated by an interface is sent to an array controller via the front-end bus, the array controller sends a request to the bus controller of a back-end bus via an intermediate bus and the bus controller

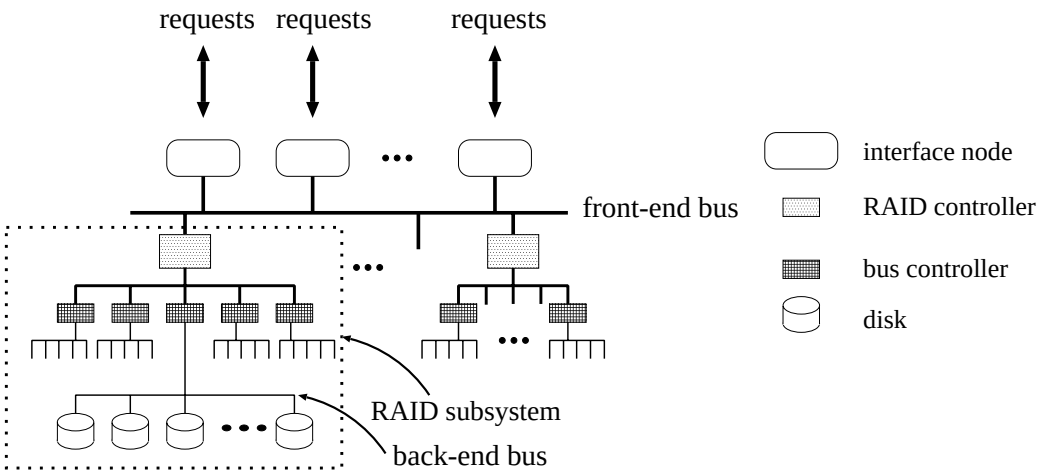

Figure 4. Hierarchical RAID system

sends it to a destination disk via the back-end bus. The result data or status follows the same path in reverse.

A request waiting for release of a bus can immediately acquire the bus as soon as the communication using that bus ends. There is no priority for bus acquisition and a request is randomly selected to acquire a bus if there are more than one request waiting for release of the bus.

3.1.3 Array Controllers

An array controller receives requests from interfaces and sends disk access requests to disks. If it receives multiple requests, it can process them concurrently; that is, if a process of one request is suspended for some reason (e.g., waiting for the release of an intermediate bus), the controller can begin processing another request.

The controller also calculates parity data to maintain redundant data as a precaution against disk failures. The redundant data is maintained as in DR-net; in other words, FPGs and SPGs are organized within a set of disks that are connected by the same intermediate bus via back-end buses and bus controllers. An FPG consists of disks connected to different back-end buses. An SPG consists of disks connected to the same back-end bus.

3.2 DR-net

Each interface node in DR-net is similar to an interface in a hierarchical disk system except that it is connected to a two-dimensional torus network by two links. All interface nodes are located at equal distances from their neighbors to evenly distribute communications over the whole network.

Each disk node is connected by four links and can communicate directions simultaneously. When a disk node tries to send a message to another node and a link which is part of the communication path is being used for other communication, the message cannot pass until the other communication finishes. A disk node can handle multiple requests like an array controller of a hierarchical disk system, but has less memory so the number of requests is much smaller.

3.3 Parameters

Table 1 summarizes the parameters used in our simulations. These parameters were adjusted to be match high-end disk systems.

We used a 1-GB/s bandwidth for the front-end bus because such a bandwidth will be common when PCI-X and 10-Gbps FibreChannel become available in the near future. The bandwidths of the other buses and links correspond to 4-Gbps and 1-Gbps FibreChannel, but the simulation was not based on any specific hardware devices.

The number of requests concurrently buffered and processed in an array controller was 50 times the number in a disk node because an array controller must manage 50 disks.

The ratios define the number of system components. For example, if the number of disks was 100, the number of interfaces, array controllers, and FPGs was 20, 2, and 20, respectively.

4 Results and Discussion

This section presents our simulation results concerning the throughput of read/write access and data reconstruction when disks fail.

4.1 Read/Write performance

Figures 5 and 6 show the read and write throughput when the number of disks rose from 50 to 500. DR-net provided

Table 1. Parameters used in simulations

bandwidth	DR-net link	100 MB/s
	front-end bus	1 GB/s
	intermediate bus	400 MB/s
	back-end bus	100 MB/s
setup time	DR-net link	50 us
	all buses	1 us
XOR performance	disk node	10 MB/s
	array controller	200 MB/s
# of request buffers	disk node	5
	array controller	250
ratio	interfaces/disks	1/5
	array controllers/disks	1/50
	FPGs or SPGs/disks	1/5
disk	unit of disk access	64 KB
	average seek time	8.2 ms
	transfer rate	50 MB/s

high read access throughput in proportion to the number of disks. On the other hand, when the number of disks was greater than 200, the throughput provided by the hierarchical system saturated at about 14,000 I/Os per second (equal to 900 MB/s). The front-end bus was obviously a performance bottleneck because its bandwidth was only 1 GB/s.

The write access throughput was lower than the read access throughput because a write operation needs six disk accesses to update one data and two parity blocks for an FPG and an SPG. Thus, the front-end bus was not overwhelmed and the write throughput increased in proportion to the number of disks in the hierarchical system also.

When the number of disks increases in a hierarchical system, more than a wider front-end bus bandwidth is needed. The bus must also be able to connect a number of array controllers and interfaces whose numbers increase in proportion to the number of disks. If the number of bus ports increases, however, it is likely to become more difficult to keep a wide bandwidth and a short setup time.

Figure 7 shows the influence of setup time in a hierarchical system with 400 disks that uses a high-bandwidth (3 GB/s) front-end bus. When the setup time of the front-end bus was short (1 us), the system throughput was as high as that of DR-net. As the setup time increased, however, throughput decreased. When a setup time increases from 1 us to 5 us, throughput was decreased by 30%.

To keep the bandwidth of front-end buses high and their setup time short, the structure of a hierarchical system can be modified to limit the number of front-end bus ports; that is, the number of array controllers and interfaces. The throughput provided by such systems, each having 400 disks and

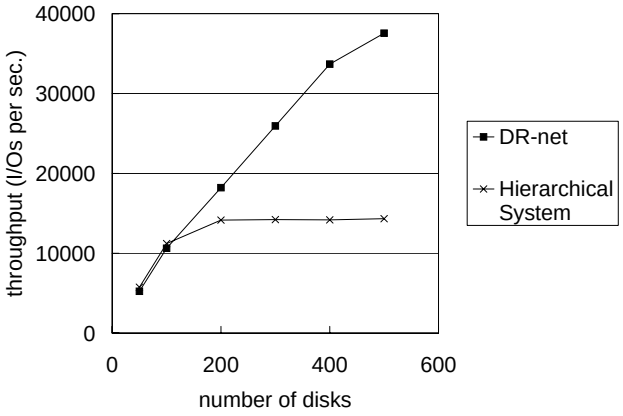

Figure 5. Read Access Throughput

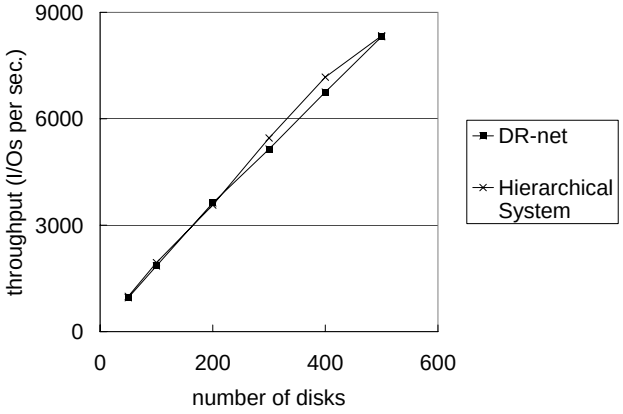

Figure 6. Write Access Throughput

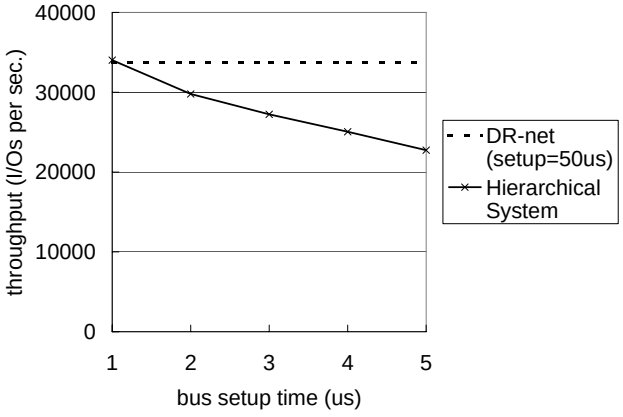

Figure 7. Influence of Setup Time (front-end bus = 3 GB/s)

Table 2. Bus Bandwidths in Modified Hierarchical Systems

40x10	intermediate	1GB/s
	back-end	200MB/s
20x20	intermediate	1GB/s
	back-end	200MB/s
10x40	intermediate	400MB/s
	back-end	100MB/s

only two array controllers and a setup time of 1 us, is shown in Figure 8. In the figure, 10×40 (for example) means that a back-end bus connects 10 disks and an array controller manages 20 (=40/2) back-end buses. The bandwidth of the intermediate and back-end buses also became higher in some systems because their number of ports are increased to compensate for the decreased number of front-end bus ports (Table 2).

By modifying the system structure, as shown in Figure 8, hierarchical systems can provide read throughput nearly as high as that of DR-net. The modification of the structure, however, affects the write throughput because each array controller must manage more disks than in the original structure. Figure 9 shows that each array controller needs much higher parity calculation throughput to provide write throughput as high as that of DR-net. The hierarchical system's throughput was only 70% that of DR-net even when the parity calculation throughput was 500 MB/s. Although CPU power continues to rapidly increase, the throughput of parity calculation depends strongly on memory access speed because a large amount of data must be calculated, and so the whole data cannot be stored in the CPU cache.

Thus, the above results show that DR-net, which has a flat uniform structure, provides better scalability than conventional hierarchical disk systems. In terms of cost, the elimination of expensive components (e.g., high-end CPUs and state-of-the-art communication technology) makes DR-net more cost-effective than the conventional systems. Furthermore, in order to provide tolerance to any single failure, the front-end bus and intermediate buses must be duplicated because if one of them fails, there is no alternative communication path and the accessibility to some data is lost.

4.2 Data Reconstruction

In this section, the data reconstruction throughput is shown as an example of the advantages of disk node autonomy.

When one or more disks fail, they are replaced by new disks and the data stored in the failed disks are reconstructed using redundant data. The data reconstruction throughput

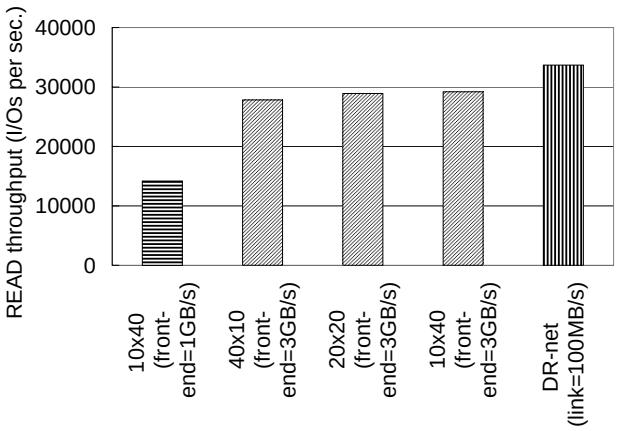

Figure 8. Read Access Throughput of Systems with a Limited Number of Front-end Bus Ports.

is important to system reliability because the probability of data loss increases if the reconstruction process takes a long time[8][3].

In a hierarchical system, when a failed disk is replaced, an array controller reads data and parity from disks in the parity group that the failed disk belonged to, regenerates the data by parity calculation, then stores it on the new disk. In DR-net, a disk node that has a new disk regenerates the data by cooperating with other disk nodes in the same parity group. A parallel parity calculation is done in multiple nodes. The data and parity used for regeneration are sent to and received from disk nodes. The communications are, however, done locally because they are within a parity group.

In the simulation described below, both systems had 50 disks and pre-fetched up to 10 disk blocks. No priority was given to a disk access request related to the reconstruction process.

Figures 10 and 11 show, respectively, the data reconstruction throughput when one disk failed and the interfaces issued read requests, and when two disks failed and the interfaces issued write requests. We used these two simulations in order to examine the influence of controller CPU workload. In the latter case, CPU workload was much higher than the former because controllers had to calculate parity data to process write requests during the reconstruction process for two failed disks.

The difference in the throughput of the two systems was clearly not caused by the parity calculation throughput because XOR performance did not affect the results. Also, the communication bandwidth was equal in both systems because the read and write throughputs were nearly equal when the number of disks was small (Figures 5 and 6).

Thus, the difference must be due to the disk nodes autonomy. Each disk node that is related to the reconstruction process can pre-fetch disk blocks or immediately send data to other nodes when the disk or the link is idle. On the other hand, in a hierarchical system, a disk does not perform a pre-fetch, even if the disk is idle, until a pre-fetch request is received from an array controller. The disk node autonomy allows DR-net to use its resources more effectively than the hierarchical system. Figure 10 shows that DR-net can reconstruct data six times as fast as the hierarchical system. If write requests are processed during reconstruction, though, the reconstruction throughput falls (Figure 11) because write requests need more system resources than read requests.

5 Conclusions

In this paper, we examined the effect of distributed functions and disk drives autonomy in disk systems. We evaluated our disk system architecture, DR-net, which connects autonomous disk drives with an interconnection network, through simulation and compared it to a conventional disk system with a hierarchical structure. System scalability and data reconstruction throughput, which is related to system reliability, were discussed.

DR-net has a flat uniform structure and provided high throughput in proportion to the number of disks. On the other hand, the throughput provided by a hierarchical system saturated because of a performance bottleneck at the front-end bus. If a high-bandwidth front-end bus with a large number of ports was used, the increased setup time decreased the performance of the whole system. An increase in setup time increases from 1 us to 5 us decreased throughput by 30%. Furthermore, if a high-bandwidth front-end bus which has few ports and a short setup time, each array controller will need an extremely high parity calculation throughput to keep the write throughput high. The hierarchical system provided only 70% of the DR-net throughput even when the parity calculation throughput was 500MB/s.

DR-net also provided high data reconstruction throughput. Because each disk node in DR-net can do its task without waiting for directions from an array controller, DR-net effectively utilizes its resources (e.g., disk drives, communication links, and so on.) When one disk fails during read-request processing, DR-net can reconstruct data six times as fast as the hierarchical system.

In this paper, we evaluated the effect of autonomy in terms of data reconstruction throughput. In actual systems, however, we expect all tasks that disk nodes process, e.g., disk access scheduling, to benefit from the autonomy. Furthermore, it is possible to execute other programs in disk nodes of DR-net because their processors are general-purpose. Currently, we are evolving our approach and working on Autonomous Disk[14][15], the successor of DR-net, that

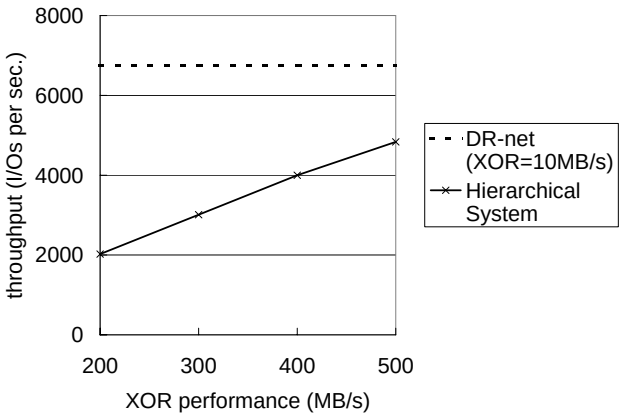

Figure 9. Write Throughput of Modified Hierarchical Systems

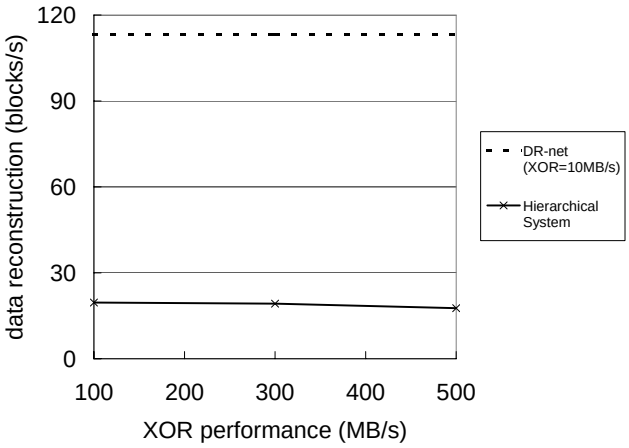

Figure 10. Reconstruction Throughput for One Disk Failure

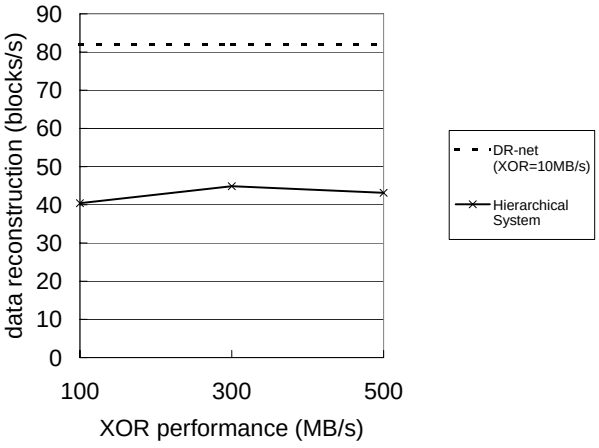

Figure 11. Reconstruction Throughput for Two Disk Failures

puts part of network file system codes into disk nodes and integrate them with maintenance of redundant data in lAN or SAN environment.

References

[1] A. Acharya, M. Uysal, and J. Saltz. Active disks: Programming model, algorithms and evaluation. In *Proc. of Architectural Support for Programming Languages and Operating Systems, October 4-7, 1998, San Jose, CA USA*, pages 81–91, 1998.

[2] 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.

[3] P. M. Chen et al. RAID: High-Performance, Reliable Secondary Storage. *ACM Computing Surveys*, 26(2):145 – 185, Jun 1994.

[4] 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.

[5] G. A. Gibson, D. F. Nagle, K. Amiri, J. Butler, F. W. Chang, H. Gobioff, C. Hardin, E. Riedel, D. Rochberg, and J. Zelenka. A cost-effective, high-bandwidth storage architecture. In *Proc. of Architectural Support for Programming Languages and Operating Systems, October 4-7, 1998, San Jose, CA USA*, pages 92–103, 1998.

[6] G. A. Gibson, D. F. Nagle, K. Amiri, F. W. Chang, E. M. Feinberg, H. Gobioff, C. Lee, B. Ozceri, E. Riedel, D. Rochberg, and J. Zelenka. File server scaling with network-attached secure disks. In *Proc. of the ACM Int'l Conf. on Measurement and Modeling of Computer Systems*, 1997.

[7] K. Keeton, D. A. Patterson, and J. M. Hellerstein. A case for intelligent disks(idisks). In *Proc. of ACM SIGMOD Conference*, pages 42–52, Sep 1998.

[8] 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.

[9] E. Riedel and G. Gibson. Understanding customer dissatisfaction with underutilized distributed file servers. In *Proc. of the 5th NASA Goddard Space Flight Center Conf. on Mass Storage Systems and Technologies*, 1996.

[10] E. Riedel, G. A. Gibson, and C. Faloutsos. Active storage for large-scale data mining and multimedia. In *Proc. of International Conference on VLDB, August 24-27, 1998, New York City, New York, USA*, pages 62–73, 1998.

[11] 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.*, pages 241–254. IOS Press, Jun 1994.

[12] J. Wilkes. The DataMesh research project. In P. W. et al., editor, *Transputing '91*, pages 547 – 553. IOS Press, 1991.

[13] H. Yokota. DR-nets: Data-Reconstruction Networks for Highly Reliable Parallel-Disk Systems. In *Proc. of 2nd Workshop on I/O in Parallel Computer Systems*, pages 105 – 116, Apr 1994. (Also in ACM Computer Architecture News Vol.22, No.4 Sep. 1994).

[14] H. Yokota. Autonomous disks for advanced database applications. In *Proc. of International Symposium on Database Applications in Non-Traditional Environments*, pages 441–448, Nov 1999.

[15] H. Yokota. Performance and reliability of secondary storage systems. In *Proc. of World Multiconference on Systemics, Cybernetics and Informatics, invited paper*, pages 668–673, Jul 2000.

[16] H. Yokota and Y. Mimatsu. A scalable disk system with data reconstruction functions. In R. Jain, J. Werth, and J. C. Browne, editors, *Input/Output in Parallel and Distributed Computer Systems, Chapter 16*. Kluwer Academic Publishers, Jun 1996.

[17] H. Yokota and S. Tomonaga. The Performance of a Highly Reliable Parallel Disk System. In A. D. Gloria, M. R. Jane, and D. Marini, editors, *Proc. of the World Transputer Congress '94*, pages 147–160. The Transputer Consortium, IOS Press, Sep 1994.