Access Load Balancing on the Parallel Storage Systems

Masahisa SHIMIZU*, Yasuhiro OUE*, Kazumasa OHNISHI*, Toru KITAMURA* and Eiji SHIMIZU**

(Recieved September 30, 1996)

Synopsis

A massively parallel computer handles massive amount of data with simultaneous access requests from multiple processors, and therefore the massively parallel computer must have a large capacity secondary storage system of very high concurrency. Such a storage system should consists many disks that are connected in paralle. With such large-scale parallel disk systems, access load balancing is extremely important to enhance the effective operation of all disks. This paper proposes a parallel file access method named DECODE (Dynamic Express Changing Of Data Entry) that implements load balancing among disks by dynamically finding the less busy disks for writing data. The efficiency of this method was verified by preliminary performance evaluation using software simulation under varying access conditions.

Keyword: parallel computer, secondary storage, load balancing, disk, RAID.

1. Introduction

We have been assumed in studies on secondary storage for massively parallel computers involving more than 1,000 processors. They are targeting scalability of up to approximately 1,000 disks.

Load balancing of access requests is extremely important in a parallel disk system of such a large scale, whose scalability is easily impaired by a heavy concentration of data accesses. Striping is often used to achieve load balancing, and there have been some studies done to determine appropriate striping sizes. They claim that an appropriate striping size can be determined by the number of concurrent access requests(simultaneity) that reach an array of disks at a given time.[1)] However, several applications having various access characteristics will run simultaneously on the massively parallel computers that we assume. Therefore, this simultaneity of access and access patterns are likely to change from time to time. This makes it hard to achieve load balancing by only static striping. One of the common methods is a dynamic load balancing in which data in a disk having a large access workload (hot disk) is transferred to another disk of less access workload(cold disk)[2)]. But this method necessarily generates side effect of an additional access for making such data transfers and thus requires the control of timing at which data is transferred.

The DECODE method being proposed in this paper for parallel file access is designed to achieve load balancing by changing the disks into which data is written according to load conditions. [3)] Although this method has no effects in situations requiring only read accesses, it ensures a substantial reduction in overhead in situations containing write accesses because it does not create any additional data access for load balancing. This paper presents the configuration of the parallel secondary storage (Chapter 2), detailed description of load balancing by DECODE (Chapter 3), the method of load information management for load balancing (Chapter 4), the results of preliminary evaluation of DECODE (Chapter 5), related work (Chapter 6), and conclusion (Chapter 7).

---

* Tokyo Information & Communication Research Center, SANYO Electric Co., Ltd.

** Fuculty of Engineering Osaka City University

2. Parallel Disk System

The overall system configuration is introduced below. Several parallel disk system models for connecting processors have been shown as listed below:

- Connect multiple storage disks directly with processors
- Connect multiple storage disks via dedicated processors for I/O processing
- Connect multiple storage disks via a dedicated network for I/O processing etc.

Among them, the third method(the model to connect multiple storage disk via an I/O network independent of the interprocessor network) was chosen for the following reasons [4]:

- All disks can be accessed in the same manner from all processors.
- Two different types of communication(i.e, communication for command executions and for I/O processing), are effectively performed.

The I/O network has communication performance of approximately 20MB/sec per port. The parallel disk system is configured as shown in Figure 1 to match the performance of the disks.

As shown above, the disks are connected to a dedicated I/O network in clusters named PDMs(Parallel Disk Modules), each PDM consisting of around eight disks and a disk node. The disk node is responsible for gathering load information of individual disks, determining target PDM or disk for write access , cache management, disk control, etc. RAIDs are used throughout the system to enhance the system reliability.

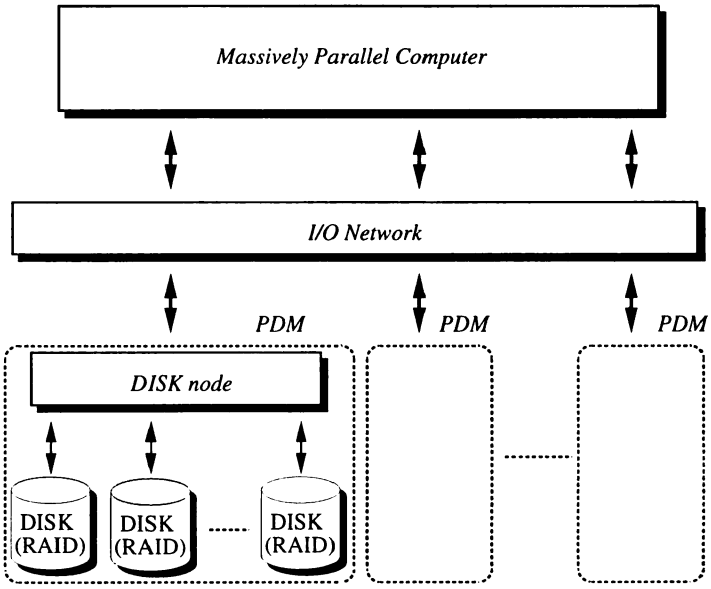

Fig.1 : Configuration of the Parallel Disk System

3. Load Balancing by the DECODE

Disk load balancing is implemented in the DECODE by the disk node (DN) that manages the load information and routes an access request to a disk with a lower workload whenever a write operation occurs. To be more precise, the processor issues a write (data update) request to the DN responsible for the disk that contains the original data. Since the DN has information about load conditions of other PDMs as well, it passes the data update request to another PDM with less workload. In a PDM, the DN intensively manages the workload of all the disk within that module, and selects a disk with a lower workload for writing data to balance the workload.

Ideally, the request should be passed on to the disk with the lowest workload for the sake of load balancing. However, data update causes a constant change in the file data allocation, depending on load conditions of individual disks, which leads to the problem of file fragmentation. Fragmentation is thought to cause a major deterioration of performance especially during sequential read operation from the same file after random

write operation. For this reason, the DN must be designed to select a write data position in disks while achieving "balancing of workload" and "suppression of fragmentation" at the same time.

The system being presented in this paper has free spaces inserted in files for changing a write disk. The write data position within a disk is selected within the same stripe group or its neighborhood as the "original position(OP)" of the data(Figure 2). Such an area is named an "equivalent area", whose size is given to the system as a parameter. In a PDM, the DN, while referencing load conditions of all disks, writes data in the free space of the least busy disk among the disks having a free space within the equivalent area.

Selecting a disk while taking the positions within the disk into consideration tends to minimize the load balancing since data is not necessarily written in the least busy disk. Nevertheless, fragmentation is suppressed drastically compared with a system without any limitation in write positions.

Since data update changes the physical allocation of file data, the data structure (i-node) must be modified each time an update takes place. Modification of the i-node for every data update generates an overhead, but such an overhead due to i-node update can be concealed by writing the i-node at the same time successively next to the data.

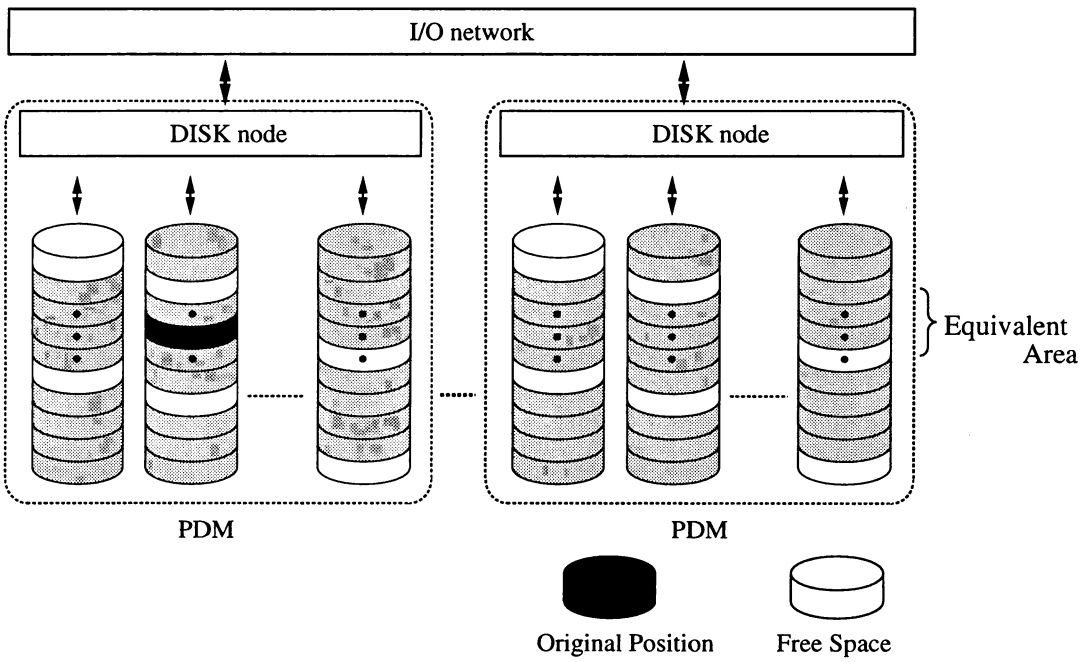

Fig.2 : Selection of Data Write Positions

4. Access Load Management

Managing the access load information of all disks is necessary to balance access workload among them. Besides, since the DECODE balances the workload on the basis of this information, it should use the most recent possible information to produce the best result. This part of the paper introduces the management method of access load conditions of the disks within the parallel disk system. This type of information management is often accomplished either by a centralized management method or a distributed management method.[6)7)] As discussed in Chapter 2, the proposed parallel disk system has a disk node (DN) to control eight disks centrally within the parallel disk module (PDM). The workload of the parallel disk system as a whole, however, is managed by distributed management of the DNs in consideration of the system scalability. For this reason, load conditions are managed in two steps; distributed management among the PDMs and centralized management within each PDM.

In distributed management, exchanging management information among the DNs is a key to successfully balancing access workload in the entire disk system. Consequently, the management information must be

exchanged among the DNs taking the followings into consideration:

- Reduction of communication overhead caused by the exchange of management information.
- Exchange of the most recent possible information.

Frequent exchange of information ensures that the latest possible information is sent, but it increases the communication overhead. In our system, several levels of access workload is assumed and, rather than sending information at regular intervals, each DN sends its load information to other DNs whenever its load level changes. Meanwhile, load information must be broadcast to all DNs in order to inform current conditions of all PDMs. In terms of the scale of the system being proposed in which each DN controls eight disks, this means one DN must send load information to more than 100 DNs because we assume scalability of up to approximately 1,000 disks. This would inevitably lead to an enormous communication overhead. Besides, assuming that all DNs rely on the same information, the PDM with the lowest workload would be bombarded by access requests from many DNs at once, which must be avoided.

In order to prevent this bombarding from occurring, DNs are allocated in several groups, and load information is gathered and access workload requests sent to other DNs within the same group. Grouping of DNs helps to minimize the increase in overhead due to broadcasting of load information. However, a simple grouping of DNs can restrict access workload transfers to occur only within a given group. This prevents load balancing to occur in the system as a whole, and leads to unbalanced loading among different groups. This problem is dissolved by overlapping groups so that each DN is included in plural groups. One way to construct such an overlapping configuration would be to give a serial number to each DN, whereby load information is sent to the DNs whose numbers differ by 1 or n bits from that of the sender(Figure 3).

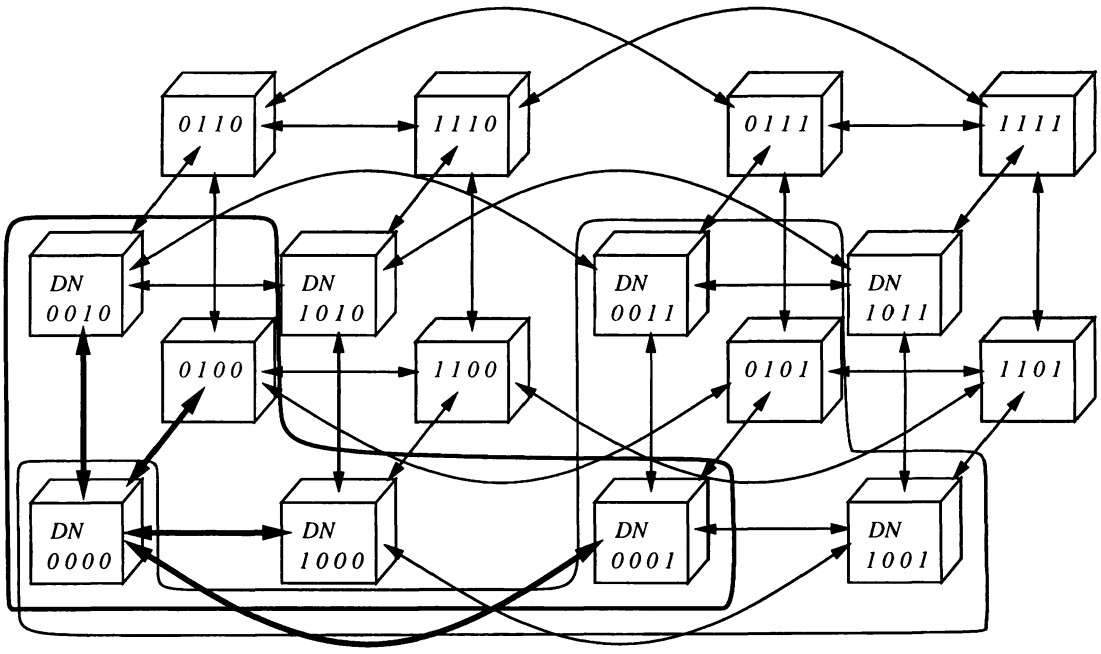

Fig.3 : Transfer of Load Information among DNs

5. Performance Evaluation

To verify the load balancing effect of this basic algorithm of the DECODE, the effect of load balancing within a PDM was evaluated by software simulation.

The simulation was conducted with the access file size of 10MB and data striped over the disks at the time the file was created. The test was designed in such a way that the total amount of workload on the write

target disk was compared with that on the original disk containing the original data. If the former was greater than the half of the latter, in other words, if only a small effect of data transfer was anticipated, data would be written in the original disk without a transfer. The total amount of workload is the sum of the workload on a disk, and the amount of each workload on the disk can be calculated with seek time, waiting time for rotation, and access data size. The performance evaluation test compared the DECODE in which the data position was changed dynamically in the units of stripe size, against the conventional method in which the data position remained unchanged. The average access response times of the two methods were compared for the sake of evaluation.

5.1. Simulation Model

The simulation model was made on the simulator tool SES/workbench that allowed description at a functional level. The simulation was conducted on a single PDM consisting of eight disks as the primary purpose of the test was to verify the effect of the basic algorithm of the DECODE. The simulation model was configured as summarized below.

- Eight disks were connected in parallel and controlled by a disk node.
- Common single disks (having parameters as shown in Table 1) were used to simplify the model although the proposed system would use RAID-3 disk arrays.
- The disk node used the access time, calculated from the size of access requests to the disks, as the workload of each disk.

Table 1 : Disk Parameters

Average seek time	12 [ms]
Rotational speed	5400 [rpm]
Sector size	512 [B]
Sectors/track	100
Tracks/cylinder	25

5.2 Test Conditions

The simulation used two types of application access patterns, presented below, in several combinations.

[Random access]

- Entire file accessed randomly
- Three access sizes:
  - 512B～16KB (1～32 sectors, random)
  - 512B～32KB (1～64 sectors, random)
  - 64KB～128KB (128～256 sectors, random)
- Access frequency: 200KB/s～
- Read/write ratio: 3/1

[Sequential access]

- File accessed sequentially from the top
- Access sizes: 1MB
- Access frequency: 10MB/s
- Read/write ratio: 1/1

5.3. Simulation Cases

The simulation was conducted on three cases to evaluate their access performance.

(a) Two applications having random access patterns were executed simultaneously.

(b) Two applications having sequential access patterns were executed simultaneously.

(c) One application having random access patterns and another application having sequential access patterns were executed simultaneously.

5.4. Evaluation Results

A basic evaluation of the three simulation cases had already led to a conclusion that access performance could be improved substantially in cases (a) and (c) by the DECODE. [5] Based on this past result, case (a) in which two applications generate random accesses were executed simultaneously, was evaluated in more detail by simulation under various conditions. In this paper, not only the influence of the access size but also the influence of the equivalent area size and the free space size are evaluated.

Figures 4 and 5 show the response times of case (a) in the various access frequency under conditions that access sizes are 1-32 sectors and 1-64 sectors. Access frequency in a given time in an application is shown horizontally while the average response time is shown vertically.

The load balancing effect of the DECODE is obvious in write accesses. In the case of an access size of 1-64 sectors, for instance, the graph indicates performance improvement of approximately 46% at access frequency of 800KB/s. In the conventional method, the response time increased sharply with the increase in access frequency. In the DECODE, the response time increased only slowly with the increase in access frequency because of the load balancing effect.

In read accesses, the response time of the DECODE indicated a 2-8% increase over the conventional in lower access frequencies. However, in higher access frequencies, the load balancing effect of the DECODE is obvious in read accesses also. In the case of an access size of 1-64 sectors, for instance, the graph indicates performance improvement of approximately 33% at access frequency of 800KB/s.This was because when the access frequency is low, an amount of workload on the disk is small and the degree of workload imbalance among disks is low as well. Thus, the influence of fragmentation eliminates the effectiveness of load balancing by the DECODE. As a result, the access performance is unchanged or little worse when compared to the performance result derived from the conventional method. When the access frequency is high, the imbalance of the workload become large.

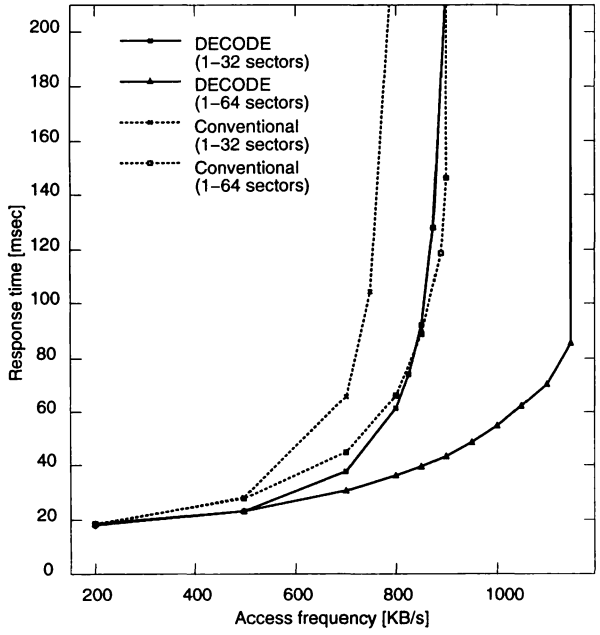

Fig.4 : WRITE Response Time for Access Frequency

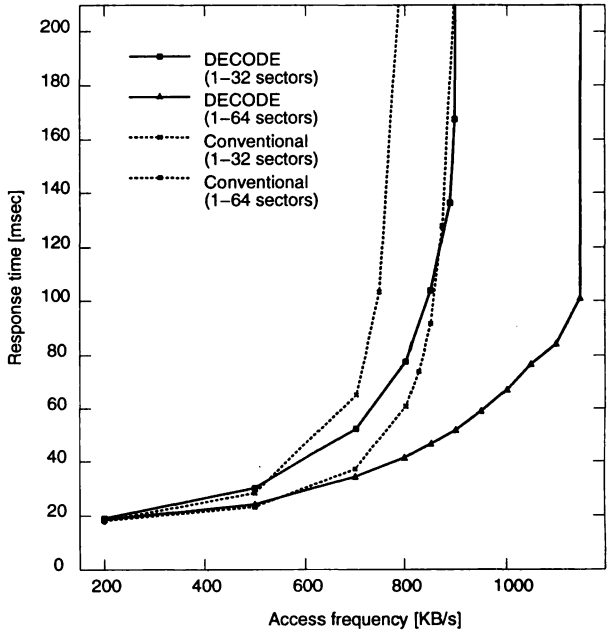

Fig.5 : READ Response Time for Access Frequency

So, the response time is greatly extended when using the conventional method. On the other hand, load balancing by the DECODE has a large positive effect on the response time. When the access frequency is high, movement of the write position is more frequent, and influence of fragmentation becomes larger. But, the DECODE can minimize the influence of fragmentation as described in Chapter 3. Hence, the positive effect of load balancing overcomes the negative influence of fragmentation and therefore the access performance can be improved. In addition, the response time increased more slowly with the increase in access frequency as demonstrated in write accesses.

Next, a similar evaluation was conducted with relatively large access sizes, namely 128-256 sectors. Figure 6 shows the results of evaluation at those access sizes. The DECODE demonstrated improvement in performance at those access sizes also.

At last, the influence of the size of the equivalent area upon performance was examined at the access size of 1-64 sectors. Figure 7 shows the results. As defined earlier, the size of the equivalent area determines the range to which data can be removed from the stripe group of the original data when updating data is written in another disk. This range is limited strongly when the size of the equivalent area is small, thereby suppressing the fragmentation more effectively. But it also reduces the effect of load balancing. Among three equivalent area sizes, 1, 10, and 100 times the stripe size(8 sectors), the best results were achieved when the size of the equivalent area was 10 times that of the stripe size.

Thus, the DECODE proved to be effective in random access applications under varying conditions in terms of the access size and equivalent area size. For applications that generate random access requests, access requests concentrate to some disks in a short term though they are even in a long term. The DECODE method balances the disk workload for the concentrated access requests, thus greatly enhancing the access performance. The above results indicate the effect of this method will be greater for applications with complicated access request patterns.

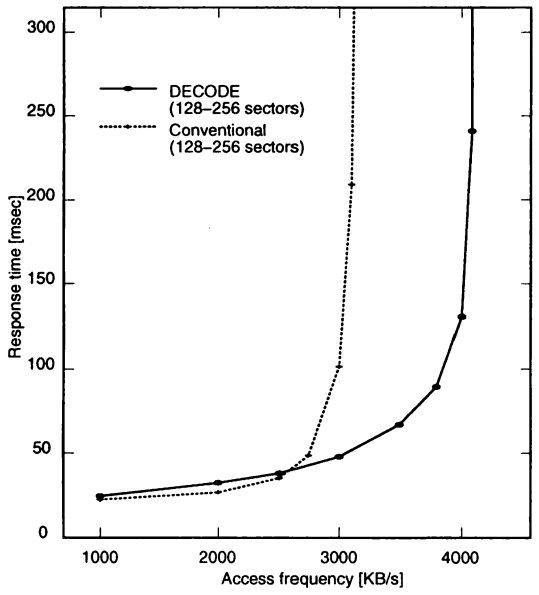

Fig.6 : Response Time for Access Frequency(Access Size 128-256)

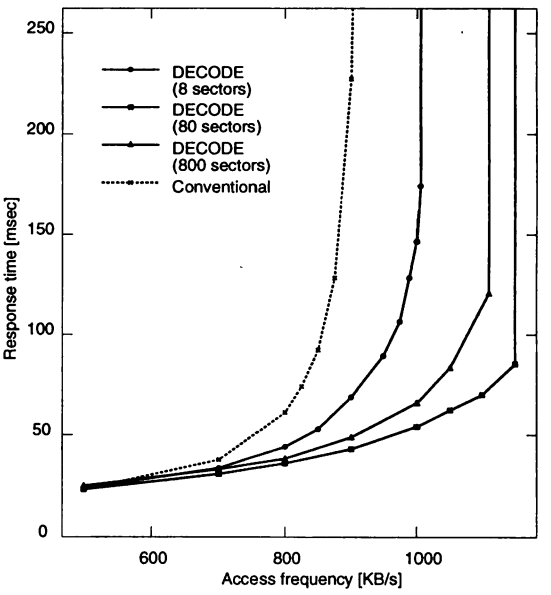

Fig.7 : Influence of Equivalent Area Size

6. Related Work

The DECODE enables high speed access to secondary storage by dynamically selecting a disk to write data according to the workload of each disk. The independence of logical storage blocks from physical storage blocks as adopted in the DECODE has already been advocated with existing methods such as LFS [8)9)] and

FLOATING.[10]

The LFS method manages files as log data. It collects multiple data modifications into a larger amount of data, then writes it into a disk as a unit. This method ensures high access performance by minimizing the head seek time. There is a research that has adopted the LFS method to RAIDs, though the LFS method does not assume parallel accessing.[9] However, the LFS method requires costly garbage collection operations to allocate continuous disk space in which to write a data segment. The DECODE method enables efficient use of disk storages as mentioned above since it uses free space that are allocated in advance for the writing of modification data.

FLOATING is a method for writing data into the closest free block to the current head position instead of the original block. Thus, this method tends to shorten the revolution wait time and enhance the disk access speed. The virtual striping[11] is a method used on RAID disk arrays for separating logical addresses from physical addresses and dynamically assigning parity stripes to reduce the parity modification cost.

The above methods aim to enhance disk access performance with individual disks or RAID disk arrays, but not with parallel disk systems. Also, these methods do not tend to balance workload of parallel disks and do not enhance the overall parallel disk system performance.

So far, various research has been performed in the field of parallel file systems for parallel computers.[12)-16)] In these researches, files are positioned in disks by the striping method which optimizes the number of disks and the striping size for each file. The optimum positioning of individual files by the above method is not always effective where multiple applications are executed simultaneously, as mentioned previously. Dynamic load balancing according to the current workload of each disk is indispensable in taking full advantage of parallel disk systems.

7. Conclusion

A method is proposed here that will allow efficient use of enormous storage resources of parallel computers by balancing access workload on secondary storage. It implements dynamic load balancing by dynamically selecting a disk for writing data on the basis of current load conditions of all available disks, thereby meeting varying access requests more flexibly.

This paper presents the proposed parallel file access method named DECODE. It introduces a concept of equivalent area designed to minimize fragmentation associated with the DECODE, and presents the method of selecting data positions that would suppress fragmentation. This paper also proposes a method of efficient distributed management of load information to implement load balancing. Finally, it reports the result of preliminary evaluation of performance by software simulation regarding load balancing within a PDM. The proposed method proved to be capable of substantially improving access performance through a series of detailed tests under various access conditions.

Future plans include preliminary verification of the parallel disk system as a whole including load balancing among clusters. The proposed system will eventually be built into a massively parallel computer to demonstrate its practical efficiency.

References

1) Chen, D. Patterson., "Maximizing Performance in a Striped Disk Array", *Proc. of Int. Symp. on Computer Architecture*, (1990).

2) Weikumm G., Zabback, P., Scheuermann, P., "Dynamic File Allocation in Disk Arrays", *Proc. of ACM SIGMOD Conference*, (1991).

3) Oue, T. Kitamura, K. Ohnishi, M. Shimizu, "Dynamic Load Balancing of File Access on Parallel Secondary Storage", *IPSJ, SIG NOTES* 95-ARC-112-1, (1995), pp.1--8. (In Japanese)

4) Ohnishi, T. Kitamura, Y. Oue, M. Shimizu, "Network Architecture and Performance Evaluation for a

Dispersive Independent I/O System", *IPSJ, SIG NOTES* 94-ARC-111-6, (1995), pp.41--48. (In Japanese)

5) Oue, T. Kitamura, K. Ohnishi, M. Shimizu, "Parallel File Access for Dynamic Load Balancing on the Massively Parallel Computer", *Proc. of Int. Symp. on Parallel and Distributed Supercomputing*, (1995), pp.179--187.
6) Watanabe, T. Ohta, T. Mizuno, and H. Nakanishi, "Adaptive Load Balancing with Bidirectional Piggybacking", *The Journal of IEICE* Vol.78 D-1 No.3, (1995), pp.302-312. (In Japanese)
7) K.M.Baumgartner and B.W.WAH, "GAMMON: A Load Balancing Strategy for Local Computer Systems with Multiaccess Networks", *IEEE Trans. on Computers* Vol.38 No.8, (1989), pp.1098-1109.
8) Mendel Rosenblum and John K.Ousterhout, "The Design and Implementation of a Log-Structured File System", *Proc.of the 13th ACM Symposium on Operating Systems Principles*, (1991).
9) Margo Seltzer, Keith Bostic, Marshall Kirk McKusick and Carl Staelin, "An Implementation of a Log-Structured File System for UNIX", *1993 Winter USENIX*, (1993), pp.201--220.
10) Jai Menon and Jim Kasson, "Methods for Improved Update Performance of Disk Arrays", *Proc. of 25th Hawaii Int. Conf. on System Science* Vol.I, (1992), pp.74--83.
11) Mogi and M. Kituregawa, "Analysis of data update performance of RAID5 disk arrays with Virtual Striping", *Proc. of the 48th Annual Convention IPS Japan* 3B-7, (1994). (In Japanese)
12) T.W.Crockett, "File Concept for Parallel I/O", *Proc.of Supercomputing'89*, (1989), pp.574--579.
13) E.P.DeBenedicts and S.C.Johnson, "Extending Unix for Scalable Computing", *IEEE Computer,* November 1993, (1993), pp.43--53.
14) P.F.Corbett, D.G.Feitelson, J.P.Prost and S.J.Baylor, "Parallel Access to Files in the Vesta File System", *Supercomputing'93*, (1993), pp.472--481.
15) R.R.Bordawekar, J.M.del Rosario and A.N.Choudhary, "Design and Evaluation of Primitives for Parallel I/O", *Supercomputing '93*, (1993), pp.452--461.
16) J.M.del Rosario and A.N.Choudhary, "High-Performance I/O for Massively Parallel Computers", *IEEE Computer,* March 1994, (1994), pp.59-68.