Data Placement of a RAID Based on Zone Disks

Yin-Fu Huang and Chia-Pin Chen
Institute of Computer Science and Information Engineering
National Yunlin University of Science and Technology
Email: huangyf@el.yuntech.edu.tw

Abstract

Due to the characteristics of MZR disks, such as variable transfer rates and capacities in different zones, the file placement has great effects on the system performance. First, according to the physical zones of zone disks, we can partition a disk array into several regions. For each region, we give a range of maximum number of data striping respectively. Then considering the factors such as file characteristics and popularity levels, the placement constraints for files are presented. According to the constraints and current disk array configuration, a file placement algorithm is proposed to allocate appropriate disk locations for files. Through the simulation, we found that the dropping rate of real-time file requests in our method is explicitly less than those in the other two placement strategies. Furthermore, we also observed that more available disk spaces in high-bandwidth regions, after using our method, make our method more elastic than the PVW placement in the future when placing high QoS files such as MPEG2.

Keywords: zone disk; data placement; disk array; multimedia data

1 Introduction

The developments of the multimedia technology grow rapidly in recent years. Among them, several video compression techniques such as MPEG1, MPEG2, and MPEG4 offer variety quality of services for users. For different video applications, a video server must have huge storage to store these videos and also enable to transfer them in real-time. To achieve the goals mentioned above, the disk zoning technique has been widely used for increasing disk capacity and bandwidth by manufacturers [8]. The basic idea is to partition a disk surface into several regions, termed zones such that outer zones constituted by longer tracks contain more data than inner zones. Then, given a fixed rotation speed, a zone disk may have variable bandwidths, depending on which zone a disk head is presently positioning on. By the way, the total bandwidth and capacity of a system can be increased by using the multi-disk technique [2, 6, 7].

In the past, some researches considered increasing the average transfer rate of a zone disk by rearranging zone layout in a logical manner [9]. Because this manner will prune some zones with a lower transfer rate, it wastes disk spaces. Some studies focused on data layout [1, 3], and they adopted the two-pair way to store files in outer tracks and inner tracks. Its drawback is that it incurs delay when users access files located at zones with a low transfer rate. Some researchers proposed the concepts of popular regions [4, 5, 10] where a disk is divided into two regions, such as hot region and cold region. The files with more popular will be placed in hot region, and the files less accessed will be placed in cold region. In the paper, an efficient file placement algorithm for a multi-zone-recording disk array is proposed. Since each file has its own characteristic, they are classified into four types. Besides, owing to different transfer rates of zones within a disk, we consider placement constraints for different file types when placing files on a disk array. According to the placement constraints and current disk array configuration, appropriate disk locations for files can be allocated in the proposed file placement algorithm.

The remainder of the paper is organized as follows. In Section 2, we describe the system model used in the paper. There some assumptions and file classifications are made. In Section 3, we formalize placement constraints based on file characteristics, and then propose the file placement algorithm. In Section 4, a simulation model is proposed, and several experiments are conducted to compare different placement strategies. Finally, we make conclusions and future research directions in Section 5.

2 System Model

The system model investigated here is illustrated as shown in Fig. 1. A server provides services to the clients that are connected with the server through a high-speed network, such as an ATM switch, a fast Ethernet, or an optical fiber, etc. Within the server system, a disk array with mass storage is connected with a high-speed system bus, such as an SCSI interface. The disk array used here is composed of several zone disks. When a client issues a request for a file access, the server must transfer the file with the service rate demanded by the client during the transmission time.

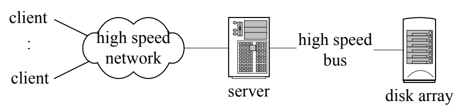


Fig. 1 System model

2.1 The Assumptions

In the model, we have made some assumptions for the server end, client ends, and file allocation, respectively. For the server end, since the problem addressed here is the file placement in a disk array, no cache or even disk scheduling is considered. However the server still has the maximum number of requests served in a service round owing to the bound disk bandwidths. For client ends, buffer availability is not considered here. For file allocation, we assume that a disk array is composed of several MZR (i.e., multi-zone-recording) disks with the following properties. Firstly, the block size within the zone disks is fixed, so the block number of outer tracks is more than that of inner tracks. Secondly, to reduce high latency and save storage space, we would partition the surface of a disk into several regions (i.e., logical zones). In other words, each logical zone in the model is supposed to contain the same number of physical zones. Thirdly, although a file could be striped into separated zone disks, it should be continuously allocated in a single zone disk. Finally the bandwidth of the high-speed bus connected to the disk array is higher than the total bandwidths of zone disks in the disk array.

2.2 File Classification

The files stored in the disk array can be classified into four types, according to their characteristics. They are as follows:

(1) Video files: They can be further classified into several standard types such as MPEG1, MPEG2, and MPEG4, etc. Each standard type has its own quality of services. However their common property is to require real-time transmission.
(2) Wave files: Similarly, audio files can be also classified into several standard types according to their compression rates. Although their sizes are far smaller than those of video files in general, they still require real-time transmission.
(3) Image files: Although they have different sizes according to different image compressing techniques, they are not necessarily transmitted in real-time. Maybe the only considerations are response time (i.e., initial latency) and total completion time.
(4) Text files and other types of files: Although they doe not belong to the first three types mentioned before, their properties are more similar to those of image files, owing to not requiring real-time transmission.

As a matter of fact, the major consideration we concern in the system is the requirement of real-time transmission of files. In Section 3, we will take into account the real-time requirement when placing files in the disk array.

2.3 Problem Definition

Since a zone disk has different transfer rates in different zones, the file placement has a close relationship with file access. In other words, if a file required with high quality of services is placed in a zone with the lower transfer rate, clients may not be satisfied with services provided by the system. In order to break through the disk bottleneck, the RAID framework is employed in our model to increase the system bandwidth. The merit of using the RAID is by virtue of the file-striping technology. However a file is not necessarily striped into all zone disks in the disk array. Here the consideration that we concern is how to place files efficiently in the disk array to meet the quality of services according to the available zones in the zone disks and the appropriate striping number across the zone disks.

3 File Placements on an MZR Disk Array

To place files on a disk array efficiently, firstly we have placement constraints for different requirements such as file characteristics and popularity levels. Then according to the placement constraints and current disk array configuration, a file placement algorithm is proposed to allocate appropriate disk locations for files.

3.1 Symbols Used to Define Placement Constraints

According to the file characteristics as mentioned in Section 2, we have different placement constraints for real-time and non-real-time transmissions. A real-time file must be transmitted in a fixed rate to strictly meet user demands in the quality of services, whereas a non-real-time file does not need to meet the demands as a real-time file, but could require being both responsive and transmitted completely in a finite time. Here the symbols used to define the placement constraints are shown in Table 1.

Table 1 Symbols used to define the placement constraints

Symbols	Illustrations	Units
K	No. of disks in a disk array	
N	No. of regions in a disk array	
R_{max}	Max. no. of requests serviced in a service round	
P_i	Popularity level of file *i*	
$Display_i$	Display rate of file *i*	MB/second
$Region_n$	Region no.	
α	Ratio of the initial-data portion	
T_i	time to transfer the initial-data portion	second
T_f	time to transfer the follow-up data portion	second
S_i	Size of file *i*	MB

3.1.1 Placement Constraints for Real-time Files

To meet the quality of services for a real-time file, its display rate must less than or equal to the bandwidth of the region storing the file. Moreover, since the required bandwidth for real-time files increases rapidly, a single disk is no longer satisfactory to them. Thus, we use data-striping techniques to store them across a multi-disk array. Basically the placement constraints to store a real-time file can be formulated as follows:

$$Display_i \le Bandwidth(Region_n) \times Striping\_no(Region_n)$$

In a client-server system, we can not exclude that more than one client access a file at the same time. Not considering the caching at the server end, the file must be transferred at a higher bandwidth. In other words, a file with more popularity will need to allocate a region with higher bandwidth for it. Here the popularity profile of files can be made according to accessed records in the past. The popularity level for a file can be expressed as follows:

$$P_i = \frac{number\ of\ requests\ accessing\ file\ i}{total\ number\ of\ requests}$$

Due to the finite bandwidth in the system, we have an upper bound R_{max} of requests serviced in a service round. According to the popularity level for a file, the maximum number of requests to access the same file in a service round can be estimated as $\lceil P_i \times R_{max} \rceil$. Thus the placement constraints to store a real-time file can be modified as follows when considering more than one client accessing it simultaneously:

$$Display_i \times \lceil P_i \times R_{max} \rceil \le Bandwidth(\ Region_n\ ) \times Striping\_no(\ Region_n\ ) \quad (1)$$

3.1.2 Placement Constraints for Non-real-time Files

Unlike the characteristics of real-time files, non-real-time files could require being both responsive and transmitted completely in a finite time, thereby adopting another type of placement constraints. For a non-real-time file being responsive in a finite time T_i, it can be viewed as two portions such as initial data and follow-up data. The initial portion of a non-real-time file should be transmitted as soon as possible to meet the time constraint T_i. On the other hand, although the follow-up portion does not require being transmitted in a fixed rate, but it should be done completely in a finite time T_f. Thus, the placement constraints to store the initial portion of a non-real-time file can be formulated as follows:

$$\frac{S_i \times \alpha}{T_i} \le Bandwidth(\ Region_n\ ) \times Striping\_no(\ Region_n\ ) \quad (2)$$

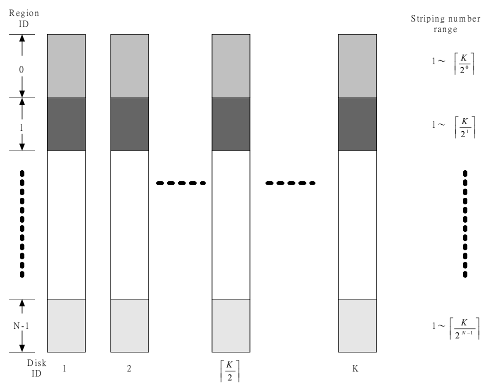
Accordingly, the placement constraints to store the follow-up portion of a non-real-time file can be formulated as follows:

$$\frac{S_i \times (1-\alpha)}{T_f} \le Bandwidth(Region_n\ ) \times Striping\_no(Region_n\ ) \quad (3)$$

In general, since a non-real-time file is more private than a real-time file, we do not consider its popularity level. Besides, when we try to place a non-real-time file into the disk array, both placement constraints must be met.

3.2 File Placement Algorithm

As shown in Fig. 2, the MZR disk array used to place files in our model has *K* zone disks and *N* regions for each disk. To enable storing huge-size files in the disk array, we arrange that the outer regions (i.e., with smaller IDs) have more striping number than the inner regions (i.e., with larger IDs) when placing files. Initially, the default maximal striping number of Region *n* is assigned as $\left\lceil \frac{K}{2^n} \right\rceil$. According to the placement constraints mentioned above and current disk array configuration, the file placement algorithm will allocate appropriate disk locations for files to meet the client requirements during file presentations. Basically, a file, regardless of real-time or non-real-time, should be allocated totally within a region.



Region ID
0
1
N-1
Disk ID
1
2
$\left\lceil \frac{K}{2} \right\rceil$
K
Striping number range
$1 \sim \left\lceil \frac{K}{2^0} \right\rceil$
$1 \sim \left\lceil \frac{K}{2^1} \right\rceil$
$1 \sim \left\lceil \frac{K}{2^{N-1}} \right\rceil$

Fig. 2 Layout of an MZR disk array

The basic concepts of our file placement algorithm can be described as follows. To place a file, we must determine its location in the disk array; i.e., its resident region ID and even striping number. Initially, as mentioned above, each region has a default maximal striping number. First, we find out all candidate pairs (region ID, striping no.) capable of transmitting the file, based on the placement constraints mentioned in Section 3.1. Then, we will select an *appropriate* pair from them, which has sufficient spaces in each striping disk to store the file. If, unfortunately, no pairs among the candidates have sufficient spaces in each striping disk in the current round, we mark the outermost region with K striping (i.e., the maximum bandwidth) "unusable" and double the default maximal striping of other inner regions. Then, as the procedure mentioned above, we repeat finding candidate pairs and selecting an *appropriate* pair in the next round. The procedure is repeated until file placement is successful or all the regions are marked "unusable". If we cannot find an appropriate disk location for the file during the allocation procedure, it could be the case such that although the total free spaces in a region are sufficient to store the file, at least one striping disk have no sufficient free spaces. Thus, we will try to migrate resident files in the region to make all striping disks enable to store the file. Here the file to be stored will have maximal data striping K to avoid violating placement constraints. Besides we have two claims for file migration; that is 1) the data striping of migrated files must be kept the same after migration, and 2) file migration is only done within a region. The file placement algorithm for an MZR disk array is formally given as follows:

File_Placement_Algorithm

```
Step 1 For a file to be stored, repeat from Step 2
       to Step 5 until file placement is
       successful or all the regions are marked
       "unusable".
Step 2 If its file type is real-time,
          then find all candidate pairs (region ID,
               striping no.) capable of
               transmitting it, based on placement
               constraint (1).
          else find all candidate pairs capable of
               transmitting it, based on placement
               constraint (2) and (3).
Step 3 Sort all the candidate pairs in an
       ascending order of transfer rates.
Step 4 Repeat from Step 4.1 to Step 4.2 until a
       candidate pair is selected to store the file
       or no candidate pairs exist.
          Step 4.1 Remove the first candidate pair
                   from the sorted pairs.
          Step 4.2 If the removed pair has no
                   sufficient spaces in each
                   striping disk to store the file,
                    then repeat from Step 4.
                    else place the file based on
                         the pair.
Step 5 If no candidate pairs exist, then
          Step 5.1 Mark the outmost region
                   "unusable" and double the
                   default maximal striping of
                   other inner regions.
          Step 5.2 Repeat from Step 1.
Step 6 If all the regions are marked "unusable",
       then
          Step 6.1 For the file, compute the
                   required spaces on each disk
                   (i.e., $S_i/K$).
          Step 6.2 Scan all the regions from the
                   innermost one until file
                   migration is successful or all
                   the regions are scanned.
                Step 6.2.1 If the scanned
                           region has
                           sufficient spaces
                           for the file, do
                           file migration.
          Step 6.3 If file migration is successful,
                    then place the file onto the
                         region.
                    else drop the file.
```

File_Migration

```
/* $D_s$: disks with free spaces less than $S_i/K$ and
       sorted in an ascending order of free sizes
```

```
/* $D_d$: disks with free spaces more than $S_i/K$ and
       sorted in a descending order of free sizes
Step 1 For the scanned region, classify the disks
       into two groups, such as $D_s$ and $D_d$.
Step 2 Repeat from Step 3 to Step 6 until $D_s$ is
       empty (i.e., file migration is successful)
       or file migration fails.
Step 3 Remove the first disk $s$ from $D_s$.
Step 4 Repeat from Step 5 to Step 6 until all the
       disks in $D_d$ are scanned (i.e., file
       migration fails) or file migration is
       successful.
Step 5 Select the next disk $d$ from $D_d$.
Step 6 Repeat from Step 6.1 to Step 6.2 until the
       free spaces in disk $s$ are more than $S_i/K$ or
       all the files in disk $s$ are scanned.
   Step 6.1 Select a file with data striping
            on disk $s$, not on disk $d$.
   Step 6.2 If the file's data striping size
            on disk $s$ is less than $Size$(the
            free spaces in disk $d - S_i/K$),
             then do file migration.
             else repeat from Step 6.
```

Here we use some examples to explain the file placement algorithm. All relevant information is shown in Table 2, 3, 4, 5, and 6.

Table 2 Parameters of the disk array

Zone #	Size(MB)	Transfer rate(MB/s)
0	5580	115.125
1	4860	104
2	4212	92.5
3	3728	78
4	2100	60

Table 3 System parameters

K	8
N	5
R_{max}	600
α	0.1
T_i	0.5
T_f	5

Table 4 Free spaces of the disk array

Zone #	Disk 1	Disk 2	Disk 3	Disk 4	Disk 5	Disk 6	Disk 7	Disk 8
0	56	29	42	53	300	251	77	450
1	200	401	512	171	212	110	100	112
2	251	211	245	132	145	167	197	118
3	140	220	300	211	220	450	190	220
4	221	115	249	224	654	109	247	129

Case 1:

Owing to the file type being real-time, it is verified, using placement constraint (1), that the required transfer rate must be more than 3.5(MB/s)*0.07*600. In Step 3, we sort all the candidate pairs in an ascending order of transfer rates as follows:

(2,2), (1,2), (0,2), (1,3), (0,3), (1,4), (0,4), (0,5), (0,6), (0,7), (0,8)

In Step 4, since the striping number for the file is 2 as indicated in candidate pair (2,2), at least two disks in region 2 must have 250MB free spaces to store the file. However, candidate pair (2,2) cannot meet the requirement. Then we find next candidate pair (1,2) is the solution.

Table 5 Layout of the disk array

Region ID	File ID	Striping no.	File size	Used disks
0	⋮	⋮	⋮	⋮
1	⋮	⋮	⋮	⋮
2	⋮	⋮	⋮	⋮
3	⋮	⋮	⋮	⋮
4	1	6	1944	1,2,3,4,5,6
	2	2	1000	2,6
	3	5	500	1,2,6,7,8
	4	3	600	2,6,8
	5	1	450	8
	6	8	300	All
	7	3	600	3,4,5
	8	4	800	1,3,7,8
	9	2	900	1,7
	10	4	400	3,4,5,7
	11	4	100	1,2,3,4
	12	5	2400	1,3,4,7,8
	13	6	1200	2,3,4,5,7,8
	14	7	700	1,2,3,4,5,6,8
	15	1	36	2
	16	1	12	3
	17	1	47	4
	18	1	267	6
	19	1	222	5
	20	1	23	7
	21	1	141	8
	22	5	500	1,2,6,7,8
	23	3	600	2,6,8

Table 6 Characteristics of four file cases

Case	QoS(MB/s)	Type	Popular prob.	Size
1	3.5	Real	0.07	500
2	Null	Non-real	Null	100
3	7	Real	0.15	800
4	3	Real	0.02	1600

Case 2:

Owing to the file type being non-real-time, it is verified, using placement constraint (2) and (3), that the required transfer rate must be more than both 100*0.1/0.5(MB/s) and 100*0.9/5(MB/s). In Step 3, we sort all the candidate pairs in an ascending order of transfer rates as follows:

(4,1), (3,1), (2,1), (1,1), (0,1), (2,2), (1,2), (0,2), (1,3), (0,3), (1,4), (0,4), (0,5), (0,6), (0,7), (0,8)

In Step 4, since the striping number for the file is 1 as indicated in candidate pair (4,1), at least one disk in region 4 must have 100MB free spaces to store the file. Then we find candidate pair (4,1) is the solution.

Case 3:

As similar to Case 1, the required transfer rate of the file must be more than 7(MB/s)*0.15*600. In Step 3, we sort all the candidate pairs in an ascending order of transfer rates as (0,6), (0,7), (0,8). In Step 4, we find that no candidate pairs have sufficient spaces in each striping disk to store the file in this round. Then we mark region 0 "unusable", and try the next round after doubling the default maximal striping of other inner regions. Again, in Step 3, we sort all the candidate pairs in an ascending order of transfer rates as (1,7), (1,8). In Step 4, since the striping number for the file is 7 as indicated in candidate pair (1,7), at least seven disks in region 1 must have 115MB free spaces to store the file. However, candidate pair (1,7) cannot meet the requirement. Then we find next candidate pair (1,8) is the solution.

Case 4:

Although the required transfer rate of the file is so low (i.e., 3(MB/s)*0.02*600) that any candidate pair can meet the bandwidth requirement, no candidate pairs have sufficient spaces in each striping disk to store the file. However, since total free spaces in region 4 are sufficient to store the file, we will migrate its resident files to make all striping disks enable to store the file. Here the required spaces on each disk are 1600/8MB, so the disks are classified into two groups and sorted within each group as follows:

D_s: Disk 6, Disk 2, Disk 8

D_d: Disk 5, Disk 3, Disk 7, Disk 4, Disk 1

Then we can migrate file 3 with striping size 100MB from Disk 6 to Disk 5, file 4 with striping size 200MB from Disk 2 to Disk 5, and file 22 with striping size 100MB from Disk 8 to Disk 5. Finally, we can place the file onto region 4 with data striping 8.

4 Performance Evaluations

In this section, we propose a simulation model and conduct several experiments to validate the superiority of our method. The simulation was done using the GPSS simulation package developed by Minuteman Software, Inc.

4.1 Simulation Model

The simulation model is depicted in Fig. 3. The request generator generates requests for file accesses and submits them to the waiting queue in an FCFS manner. Then the RAID server fetches the requests from the waiting queue for each service cycle, and dispatches disk bandwidth according to the QoS of files accessed. If the system can offer sufficient bandwidth for a file, the transferring unit will start to transfer it until completion; otherwise, the system will check the file type. If the type is real-time, the system will drop the request; otherwise (i.e., a non-real-time file), the request will be re-scheduled to the waiting queue and waits for services in the next round. Besides, the simulation parameters are shown in Table 7.

Table 7 Simulation parameters

Block size	128 KB
Spaces per disk	20 GB
Max. transfer rate per disk	115.125 MB/s
Min. transfer rate per disk	60 MB/s
System bus transfer rate	1600 MB/s
No. of disks	8
No. of regions	5
Max. no. of requests serviced in a round	400
Ratio of the initial-data portion	0.1
time to transfer the initial-data portion	0.1 sec.
time to transfer the follow-up data portion	5 sec.
QoS of MPEG1 files	1.14 Mbit/s
Length of MPEG1 files	5 ~ 60 min.
QoS of MPEG2 files	3 ~ 10 Mbit/s
Length of MPEG2 files	60 ~ 90 min.
QoS of wave files	0.17 Mbit/s
Length of wave files	3 ~ 8 min.
Size of non-real-time files	1 ~ 50 MB

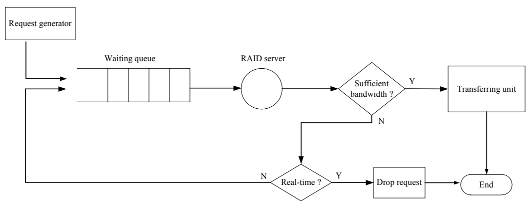


Fig. 3 Simulation model

4.2 Experiments and Analyses

In order to validate the performance of our method, we compare it with two placement strategies. One is a random placement in which all files are allocated in a random manner. Another is the popular-based variable way placement (PVW) [5] that divides the disk array into three groups, and each group has a fixed striping number. Then files are placed on the disk array according to their popularity. In addition, we also explore the influences resulted from the ratios of real-time files and non-real-time files, as shown in Table 8.

Table 8 Seven ratio cases

Case	Ratio (real-time : non-real-time)	MPEG2 : MPEG1 : Wave
1	1 : 3	5 : 3 : 2
2	1 : 2	5 : 3 : 2
3	1 : 1	5 : 3 : 2
4	2 : 1	5 : 3 : 2
5	3 : 1	2 : 1 : 1
6	3 : 1	1 : 1 : 1
7	3 : 1	1 : 2 : 2

Experiment 1: Average no. of currently serviced requests

In the experiment, we observe the average number of currently serviced requests of three placement strategies. We measure the number of current requests per 50 seconds, and take their average as shown in Fig. 4. We found that the PVW placement and our method have better performance than the random placement, since both the PVW placement and our method consider the characteristics of zone disks, and efficiently make use of disk bandwidth.

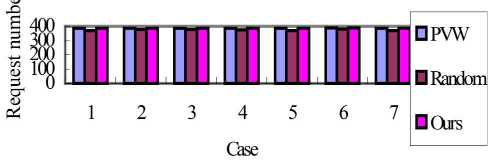


Fig. 4 Average number of currently serviced requests

Experiment 2: System throughput

In general, the throughput is a typical measure parameter. However, it is unfair to use the typical throughput to evaluate the performance in a multimedia system since different types of files have different required bandwidth. Thus, a new measure parameter called weighted throughput is defined here. In other words, a file requiring more bandwidth will be more weighted in measuring the system throughput. In the simulation, the weights of four file types are 18~59 for MPEG2 files, 7 for MPEG1 files, 1 for wave and non-real-time files, respectively. In the experiment, we observe the weighted throughput of three placement strategies. As shown in Fig. 5, the random placement is still the worst among all placement strategies. Besides, our method has better or the same performance than/as the PVW placement in all cases.

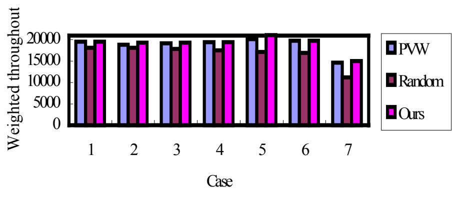


Fig. 5 System throughput

Experiment 3: Dropping rate of real-time file requests

In the experiment, we observe the dropping rate of real-time file requests of three placement strategies. As shown in Fig. 6, the dropping rate of our method is the least among all placement strategies in most cases. The reason is that our method always places real-time files onto the most appropriate locations in the disk, thereby satisfying the required bandwidth of different file types.

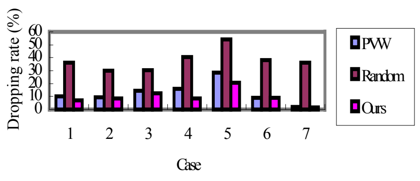


Fig. 6 Dropping rate of real-time file requests

Experiment 4: Available disk spaces

From the results of Experiment 1 and Experiment 2, it is not easy to judge which of our method and the PVW placement is the winner. Thus, in the experiment, we observe the available disk spaces of each region after using our method and the PVW placement. As shown in Fig. 7(a), 7(b), and 7(c), we found that the PVW placement tends to over-utilize high-bandwidth regions when placing high QoS files such as MPEG2. This makes our method more elastic than the PVW placement in the future when placing high QoS files. Following up Case 5, 6, and 7 as shown in Table 8 (i.e., decreasing the portion of high QoS files), we found that our method is gradually leaving more high-bandwidth regions, whereas the PVW placement still over-utilizes high-bandwidth regions.

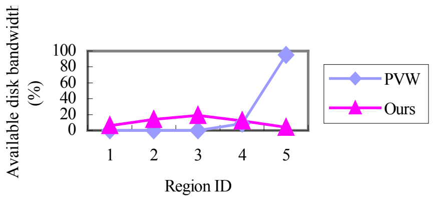


Fig. 7(a) Available disk spaces for Case 5

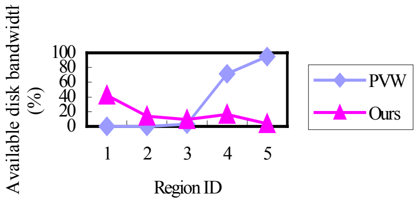


Fig. 7(b) Available disk spaces for Case 6

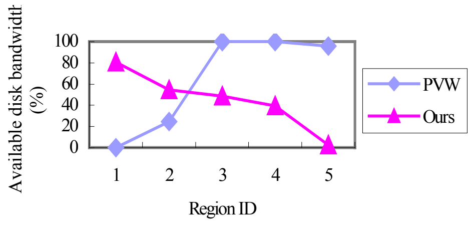


Fig. 7(c) Available disk spaces for Case 7

5 Conclusions

In this paper, we consider the characteristics of zone disks, and partition the disk array into several regions. Next, the placement constraints for different requirements such as file characteristics and popularity levels are presented. Finally, according to the placement constraints and current disk array configuration, a file placement algorithm is proposed to allocate appropriate disk locations for files. Through the simulation, we found that the dropping rate of real-time file requests in our method is explicitly less than those in the other two placement strategies. Furthermore, we also observed the available disk spaces of each region after using our method and the PVW placement, and the results show that our method is more elastic than the PVW placement in the future when placing high QoS files such as MPEG2.

References

[1] Y. Birk, "Track-pairing: a novel data layout for VOD servers with multi-zone-recording disks," Proc. Second IEEE International Conference on Multimedia Computing and Systems, 1995, pp. 248-255.

[2] R. Flynn and W. Tetzlaff, "Disk striping and block replication algorithms for video file servers," Proc. Third IEEE International Conference on Multimedia Computing and Systems, 1996, pp. 590-597.

[3] Shahram Ghandeharizadeh and Seon Ho Kim, "A comparison of alternative continuous display techniques with heterogeneous multi-zone disks," Proc. Eighth International Conference on Information and Knowledge Management, 1999, pp. 442-449.

[4] Jeong-Won Kim, Hyoung-Roung Lim, Young-Ju Kim, and Ki-Dong Chung, "A data placement strategy on MZR for VOD servers," Proc. International Conference on Parallel and Distributed Systems, 1997, pp. 506-513.

[5] Jeong-Won Kim, Young-Uhg Lho, and Ki-Dong Chung, "An effective video block placement scheme on VOD server based on multi-zone-recording disks," Proc. Fourth IEEE International Conference on Multimedia Computing and Systems, 1997, pp. 29-36.

[6] HweeHwa Pang, B. Jose, and M.S. Krishnan, "Resource scheduling in a high-performance multimedia server," IEEE Transactions on Knowledge and Data Engineering, Vol. 11, No. 2, 1999, pp. 303-320.

[7] Young-Sook Park, Jeong-Won Kim, and Ki-Dong Chung, "A continuous media placement using B-ZBSR on heterogeneous MZR disk array," Proc. International Workshops on Parallel Processing, 1999, pp. 482-487.

[8] C. Ruemmler and J. Wilkes, "An introduction to disk drive modeling," Computer, Vol. 27, No. 3, 1994, pp. 17-28.

[9] Sheau-Ru Tong, Yee-Foon Huang, and J.C.L. Liu, "Study on disk zoning for video servers," Proc. Fifth IEEE International Conference on Multimedia Computing and Systems, 1998, pp. 86-95.

[10] Jun Wang and Yiming Hu, "PROFS-performance-oriented data reorganization for log-structured file system on multi-zone disks," Proc. Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001, pp. 285-292.