An Effective Data Placement Scheme to Support Fault-Tolerance in Video on Demand Server System

Wen Shih Huang Cheng Zen Yang Cheng Chen

Institute of Computer Science
and Information Engineering
National Chiao Tung University
Hsinchu, Taiwan, R.O.C.
Email:cchen@eicpca5.csie.nctu.edu.tw

Abstract

Recently, the video server plays an important role in VO or near VOD system. In general, a good video server can not only support playback operation, but also provide reliable service. In this paper, we will propose an effective data placement scheme f or supporting fault -tolerance in distributed video server environment, called Two Level Clustering (TLC) method. This method includes three parts, popularity-based replication, two level clustering, and shift load scheduling. Using popularity -based replication t calculate the number of replicas, we place the videos with two level clustering, and schedule with shift load scheduling policy. This method can support fault-tolerance ability and balance the load of the disks, thereby helping upgrade system perfo rmance crucial to archiving maximal server throughput and improve the system reliability According to our preliminary performance evaluations, we have found that our method is superior than other schemes. The detailed information about design principles a nd performance evaluations will be described in the literature.

Keyword: fault tolerance ,video server, replication, Popularity-based, two-level cluster

1.Introduction

Recent advances on multimedia technologies have made versatile multimedia services feasible on entertainment and commercial business. One of the most common multimedia applications is Video On Demand (VOD) [1,2]. Generally speaking, video data need a large amount of bandwidth. So that the storage bandwidth and disk bandwidth ha ve become one of the most important issues of the video server system design. Video server typically stores a huge number of video files [7]. However, the larger the video server is, the more vulnerable to disk failures it becomes. In order to ensure unint errupted service even in the presence of a disk failure, a server must be able to reconstruct lost information. In general, there are two ways to add redundancy within a server: replication -based schemes [10,11] and parity-based schemes [9]. Replication-based schemes usually cost more redundancy space, and parit -based need to reserve partial resources.

In this paper, an effective data placement method, named "Two Level Clustering" and job scheduling policy with popularity -based replication are proposed for our VOD server system. It replicates and adjusts replica number of video corresponding to client request pattern t increase the reliability of video server. By replication, high degree fault-tolerant and high bandwidth is available. By this method, we take advantage of concept of clustering twice to handle the load skew problems. At the first level, we divide each video into fix -sized segments to eliminate the influence of video length. And at the second level, we limit each video file on a cluster group to decline the striping penalty and adding video cost. Our job schedulin policy exploits the property of video segment to detect and shift loads between high load DSG and light load DSG. When a stream is beginning to access a new segment, system will query the load of all replicas and shift to the lightest replica to balance the load distribution.

The remainder of the paper is organized as follows. In Section 2, concept of our video server system will be described. In Section 3, we will present the concept and principle of proposed data placement method and job scheduling policy. Related performance gains will be evaluated and analyzed in detail in Section 4. Finally, concluding remarks is given at the end of this paper.

2. Our System Architecture

We have designed and implemented an effective singl server management system for VOD applications [15]. As shown in Figure 1, SMU(Subscriber Management Unit) contains five modules: Admission Control, Job Schedule, Disk Schedule, Data Placement, and CD/HD Swappin g. The admission control module acts as the doorkeeper of video server. With the purpose of keeping service quality of viewing streams, this module guarantees that the service quality will not be degraded or be contravened by newly incoming request. The ne wly request will be accepted i and only if the admission constrain is not violated. And the admission constrain is set according to the capability of system and current service condition. In order to keep accepted streams for continuous playback, job schedule module is used to control VSE to retrieve video data from disks and then buffer them in a cycle fashion. Meanwhile, disk schedule module is used to optimize the performance of data reading from disks by rearranging the order of retrieval commands from job schedule module. In VSE(Video Server Engine), there may be multiple disks equipped as storage device to store hundreds of contents. Data placement module will control how to place the contents into storage space. And the CD/HD swapping module response to update the contents from CD-ROM into disks [15].

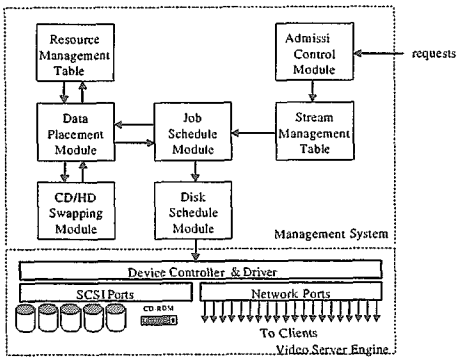

Figure 1. Basic Configuration of Our Single Video Server System

The original design of VSE aims at small -scale video-on-demand service. However, as the requirement of service grows up, the performance of single video server is limited. We may employ "multiple" autonomous servers t alleviate the problem of massive load. Nevertheless, the great skew in user access still makes some nodes overflow. If popular contents are replicated across all autonomous nodes for load balance, then the cost is too expensive. S we want to re-design the original video server configuration and a distributed architecture of video server cluster is proposed to meet the requirement of scalability. In order to extending the single video server to our distributed architecture, some modifications and assumptions have t be made in VSE. First, each VSE must be accessible by all SMU. This means each SMU can use all the resource t service users. Second, VSE equips the ability of routin video data to clients. Namely, each VSE is also accessible by all clients in some way. The required routin information comes from SMU.

Our distributed system maintains the base architecture of two-level management, i.e. SMU and VSE. The main difference is that we may have two or more SMU and VSE respectively. The overall system architecture is shown in Figure 2. Each SMU communicates with other SMU and VSE through Ethernet. All VSE nodes are interconnected with a high-speed network for exchanging video data, and the hig -speed network is in turn connected to outward service network. There are three entities in our design. The Master Server is used for admitting and dispatching user requests to slave SMU nodes. Slave SMU handles the service requests of admitted users, and VSE performs storage management, buffer management, and data retrieval.

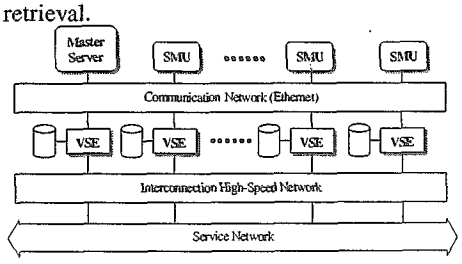

Figure 2. Architecture of Our Distributed Video Server System

3.Two Level Clustering Method

3.1 Popularit -Based Replication

In general, different videos have different popularity, the access patterns to so-called "hot" videos must be taken into consideration when designing a VOD system [4]. According to the access patterns we recorded, we can find that some hot video files may be concurrently accessed b hundreds of clients, while other video files may be accessed by few clients. Based on this fact, there are two basic approaches to solve such problem. One is "striping"[13,14], and the other is "replication"[9,10]. Data striping can increase the bandwidth of data, but striping penalty restricts the number of striping factor. If the number of clients is huge, the requirement of bandwidth may exceed the upper bound of single data bandwidth. In our approach, replicating certain frequently accessed video files in some disk arrays is another viable way to provide VOD services and achieve fault tolerant function. The concept of replication is that the figure of each video stored in storage system is not the same. That is, we can replicate the popular videos on several disks in order to provide sufficient bandwidth for all accesses to these videos. With storing more than one replicate video, the advantages are that replicate video can provide extra bandwidth and then fault-tolerant effect is available.

If we replicate video data according to popularity of video and place them in disks balanced, we can get good performance in beginning. As time passing, the popularity of video maybe changed, then the system performance will not be kept as usual. Another problem is that if the popularity statistics were not available, how to set up the initial placement? The better solution to this situation is that the number of replicas of each video s hould change over time, as the video access pattern be changed. That is, we can dynamically adjust the number of replicas by some ways, which is so called dynamic replication technique [18] In fact, it is difficult to implement dynamic replication efficiently, though it can solve the problem of popularit changed. Hence, we propose an effective policy, called Period Adjustment Policy to be implemented in our syste easily. The idea is that we can adjust the video replicas within appropriate time period (e.g. every day or every two days). The reason is that the popularity may not change very dramatically every day or every two days generally. If the popularity is changed quickly, we can adjust the time period to more short value. The change of access to each video can express as a linear function show below: let N_i be the access number of video i, t be the time period, V be the total number of video, and $C_i(t)$ be the replica number at period time t. For video i, we have[4]

$$N_i(t) = 2N_i(t-1) - N_i(t-2),\ N_i(t) \geq 0 \quad (1)$$

and $$C_i(t) = \frac{C_i(t-1) \cdot N_i(t)}{N_i(t-2)},$$

$$\text{if } C_i(t) - \lfloor C_i(t) \rfloor \geq 0.5, \text{ then } C_i(t) = \lceil C_i(t) \rceil \text{ else } C_i(t) = \lfloor C_i(t) \rfloor$$

$$, \ C_i(t) \geq 1 \quad (2)$$

We assume that Ni(t) is a linear relation of Ni(t-1) and Ni(t-2). So we can obtain formula 1 from Ni(t-1) and Ni(t-2) using extrapolation. And we assume that Ci is proportion al to Ni to obtain formula 2. These formulas are both based on linear relation, and the simple predictors are enough [4]. For simplicity, we round Ci into a integer. So we can adjust the number of replicas of video according to the calculatin result of formula 2. Besides, the alterable of adjustment period, t, provides more flexibility. That is, we can adjust the value at any time.

3.2 Data Placement Scheme

Here we introduce our placement scheme called "Two Level Clustering Method" (TLC) in some detail.

Basically, we take advantage of clustering technique twice to eliminate the negative factor caused by striping as well as attain the requirements of video server. At the first level, we divide each video into fix-sized segments. And at the second level, we limit each video file on a cluster group instead of across all disks. Then we can handle each video in segment unit rather than block unit. Combining the merits of mirrored and CSSP [15] methods, TLC is particularly useful for storing multiple video replicas in a disk-arra -based video server.

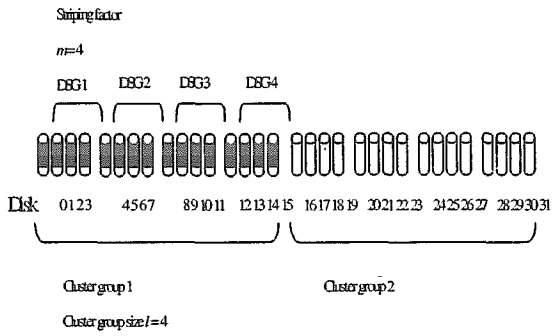

Figure 3. An example of TLC, m = 4, l = 4

In our video server architecture, the block unit size is 94kbytes [16]. The stored video may have different content size. In order to eliminate the load imbalance caused by content length, we adopt the concept of segment as similar to CSSP [15]. According to [7], we must avoid to use pseudo disk arrays in logical volumes with large stripin sizes Because performances between practical case and ideal case are enlarged from striping factor of 4 [7], the gap will grow with striping factor. Hence, we set the default value of striping factor to be 4. We take figure 3 as an example. Each DSG (Disk Striping Group) where a segment is placed consists of 4 disks. This is the first level clustering. So the problem is how to arrange the segments into all DSGs. For the sake of dynamically adding or removing video replicas, we limit the range of a vide replica, which is called "cluster group". We can find tha t the shadow parts in figure 3 illustrate only simple striping across DSG1 to DSG4, so DSG1 to DSG4 consist of a cluster group. This is the second level clustering. Thus it is not necessary to rearrange entire videos when adding / dropping occurs. Another important reason of not direct striping all segments into all DSGs is that there is stripin penalty among segments. If we direct simple striping all segments into all DSGs, the penalty between DSGs is not negligible. The cause of penalty is the same as striping penalty. Therefore we may take advantage of cluster group to avoid across too many DSGs in order to eliminate the penalty.

After deciding the striping factor and cluster group size, we need to choose a cluster group to place the replica of a video. There are two principles to choose proper cluster group. One is cluster group with more free space, and the other is no other replica of the video existed. We take figure 4 as an example. Each square represents a segment in a DSG. Video "A" has a replica " a". The sign "x.y" means the y segment of video x. The shadowed parts are those disk spaces being used. We place all videos sequentially. We can see clearly that when we place video "c", according to the first principle, we should choose cluster group 2. Because the same replica "C" has existed in cluster group 2, in order to follow the second principle, we choose cluster 1 to place video "c". The advantage of distributing video into different cluster group can increase the degree of fault tolerance. In other words, while disk failure, the traffic to the failed disk can be spread across all disks containing the replica. When each stream access to failed disk, scheduler will shift it to other active replica in order to spread load balanced.

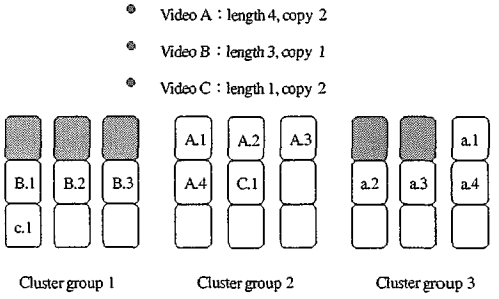

Figure 4 twolevel clustering method

By the above description, we can formulate our TLC in the following. For simplicity, each block of each segment is simple striping in a DSG and each segment is simple striping across a cluster group. Given a set of N=mlk disks, where m is the striping factor (SF), l is the cluster group size, and k is the cluster group number. If the segment size is defined as s, the cluster group is chosen as c, the start DSG of cluster group c is g, then we place the i th block of a replica of a video on the disk numbered d(i). It is not difficult to derive d(i) by the following formula

$$d(i) = mlc + (mg + \frac{i-1}{s} \cdot m + (i-1) \bmod m) \bmod ml \cdot$$

The formal algorithm of this scheme is shown in figure 5.

In our scheme, we achieve disk fault-tolerant b replication. Because all videos spread into all disks balanced, once disk fails, load can distribute into all disks. Taking mirror and interleaved declustering for example, mirror shift loads on failed disk into another disk, and interleaved declustering shift loads on failed disk into disks in the group. For TLC, it shifts loads on failed disk nearly total disks. If we spread the load into more disks, the load condition is more balanced. On the other hand, TLC usually replicates hot videos many times (e.g. 4 or more). So it can tolerate multiple error for single video.

3.3 Job Scheduling Scheme

The goal of our enhanced job scheduling policy is t shift the loads from heavier load segment into lighter load segment in order to achieve well-balanced condition based on our TLC method. The load distributed situation will be changed with time. It costs enormously to adjust each stream in each round. Within TLC, we can test and shift the stream at the beginning of access to a segment. We will show that the cost is low later. The policy of our job scheduling is executed by the following steps.

(1) In each round, we pick out the streams in two kinds of states. One is that the stream accesses the startin block of a segment. The other is that the access disk i failed and the segment has other replicas. For example if our segment size is 800 blocks, a stream with more than one replica will be picked when it access block no.1, no.801, no.1601...
(2) Scheduling remaining streams in turn. That is, t

arrange the disk to retrieve the video data for each stream.

(3) Sorting those picked streams with the number of replicas in ascend.

(4) Scheduling the picked stream to the segment of the replicas which the loads is the lowest.

The shifting of loads in our job scheduling scheme ca improve the system performance. Figure 6 is an example. The loads in cluster group A, B, C are 4, 2, 0 respectively, and small square symbol "a.b" represents a stream which accesses block b of video a. Because loads of cluster group B is lighter than A and two stre ams are accessing the beginning of a new segment(y.1 and y.801), the two loads in cluster group A can be shifted to cluster group B. Similarly, loads of cluster group C is lighter than B, and a stream(w.801) in cluster group can be shifted to C. After the shift, we can observe that the imbalance factor decreases. That is, the balance condition is better. So, our job scheduling improves the balance condition indeed. Formally, the overall scheduling algorithm is shown in figure 7

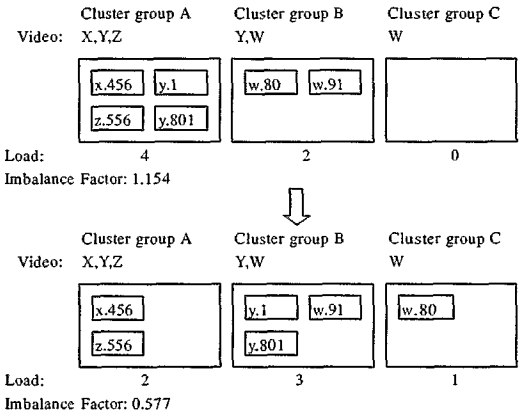

Figure6. Example of shift loads

Algorithm 1: Data Placement

Input: video_list with sort in ascend by video length,
Let A=(0,1,2,...m-1) denote m DSG, Thus DSG j denoted j, $0 \le j \le m-1$
Let C=(0,1,2,.....n-1) denote n cluster group.
Thus cluster group i denoted i, $0 \le i \le n-1$
$m = l * n, l \in N$
cluster group set S_i at time t, $S = R \cup C$
$R \cap C = \{\phi\}$
R : DSGs set can be chosen C : DSGs set have be chosen
l = cluster group size, m = striping factor
s = segment size, g = start DSG of xi

Ouput: video allocation table

Program:
While (video_list is not empty) **do**
set C=ϕ , set R = set S.
get video x from video_list
for i = 1 to x.replication number **do**
/*choose destination cluster group */
choosen cluster group $xi = \max\{FS_y(t) | y \in R_i(t)\}$
and $C_i \neq C_j, \forall i, j \in x$
/*choose max free space and no replica existed one */
C += {x_i}, R -= {x_i}
/* delete x_i from set R, and join set C */
for j = 1 to x.length **do**
/* choose destination disk */
block j place in
$disk(j) = mlx_i + (mg + \frac{j-1}{s} \cdot m + (j-1) \bmod m) \bmod ml$
end for
end for
end while

Figure 5. Algorithm of two level clustering method

Algorithm 2: Job Scheduling

Input:stream_list,alter_stream_list,
disk[total_disk].

Output: stream_list with load balance

Program:
while stream_list is not empt **do**
get stream i from stream_list
if i.block **equal** i.length
delete i from stream_list
else if (i.block % segment_size) **equal** 0
or i.access_disk **equal** failed
move stream i to alter_stream_list
else
schedule stream i /* retrieve video data */
end if
end if
end while
sort streams in alter_stream_list in ascend by replica number
while alter_stream_list is not empt **do**
get stream i from alter_stream_list
choose light load DSG for i
schedule stream i /* retrieve video data */
move i back to stream_list/* move back to stream_list for next round */
end while

Figure 7. Algorithm of job scheduling

Next we will give preliminary analysis of the complexity of our job scheduling algorithm. The state of our problem is: the video requested by each stream has different replica number. Each round, we try to assign each stream to access disks, or the stream will be interrupted. If each disk can accept one stream to access, it i s like a maximum-bipartite-matching problem shown in Figure 8(a) The maximum-bipartite-matching problem is defined as finding a maximum matching in a bipartite graph [19]. Given an undirected graph G =(V, E), a matching is a sub set of edges $M \subseteq E$ such that for all vertices $v \in V$, at most one edge of M is incident on v. We say that a vertex

$v \in V$ is matched by matching M if some edge in M is incident on v; otherwise, v is unmatched. A maximu matching is a matching of maximum cardinality, that is, a matching M such that for any matching M', we have $|M| \geq |M'|$ [19]. Based on this concept, we can transform our problem into corresponding bipartite graph by the following way. The nodes in the left side are set of request streams, and nodes in the right side are set of disks. Practically, each disk usually can be accessed by more than one requested stream. Consider an example shown in figure 8(b). The number at right of each disk number is the maximum requests of that disk. So we expand the number of maximum requests of each disk as nodes as shown in figure 8(b). That is because that in bipartite graph, only one to one relation is allowed. Our state allows many to one relation. By expanding, we can transform our state into one to one relation, which is equivalent to the maximu bipartite graph problem. For example, disk no.2, no.3, no.5 extend from one node to two nodes in figure 8(b), The same number nodes represent the same disk. Thus it can satisify the assumptions, one to one relation, of maximum-bipartite-matching. If we denote stream number which have more than one replica as S, candidate disks as D, and we assume each stream have N replicas. The worst complexity of this scheme is O((S+D)SN). That is, the calculating complexity is pretty low.

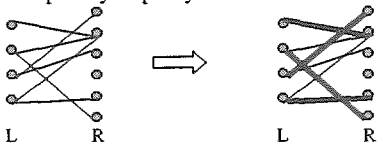

Figure 8(a). A bipartite graph G=(V, E) with V=LUR

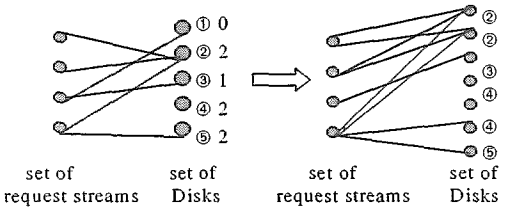

Figure 8(b) convert of our scheme

3.4 Map to Our distributed Environment

In this section, we will describe how to apply TLC to our distributed environment. We will introduce the state of multi-VSE and constrains of mapping procedures firstly. Then we describe the steps of placing video on multi-VSEs with TLC method.

In our distributed architecture, there are multiple SMUs and VSEs. Two basic assumptions are given here.

(1) The number of disks in each VSE must be the same.
(2) The number of disks in each VSE can't be prime number.

If these two constraints are not followed, it is more difficult to config and manage the VSEs. We will map the TLC method to our distributed system by the following procedures.

(1) By the capacity of single VSE, we decide the appropriate number of cluster group in it. Due to th number of VSEs is fixed, larger cluster group represents less choices when placing video and les replicas of a video allowed. Taking figure 9 as an example, if we set each VSE as a cluster group, ther are only three clusters group in system. For each video, it has three replicas at most. It implies that there ar only three choices at most when shifting loads, so w benefit less from our job scheduling scheme. On th other hand, if we set each VSE as two cluster groups. There are totally six clusters in system. Thus, it implies more choices for each video. However if we divid each VSE into too many cluster groups, we have t replicate each video more in order to provide large bandwidth. Thus, it will increase the redundancy cost.
(2) Choosing the value of striping factor, in general, 4 or is suggested [7]. The reason is that when the stripin factor is larger than 8, the striping penalty will cause load skews to degrade the performance and there are more disks which are influenced when adding video In additional, larger striping factor represents les DSGs. i.e. the effect of striping segment will decline So appropriate value of striping factor, like 4, i adopted in our scheme and environment.
(3) After the striping factor and cluster group size ar determined, the next step is to sort all video that i prepared to place in disks by length.
(4) With our policy, we get the longest video and select th most free space cluster group to place. If no othe replica of the video exist, we place the segments of th video replica onto the cluster group in simple stripin manner.
(5) Repeating step (4) to place others replicas of the video
(6) Repeating step (4) and (5) for the longest video in th remaining videos.

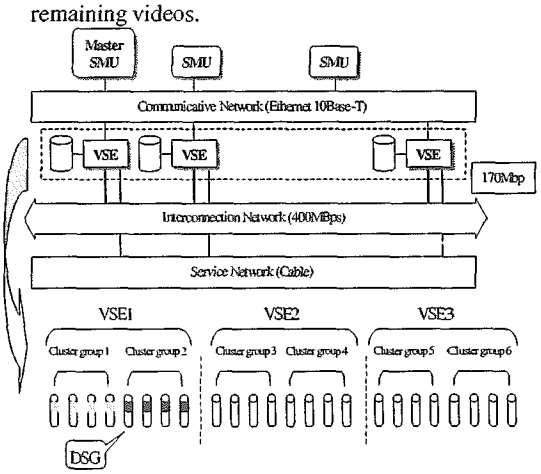

Figure 9. Two-level clustering in Multi-VSE

After placing all videos, the video allocation table wil save in each slave SMU for job scheduling. By the above procedures, we can operate our system both in normal mode and failure mode easily [20]

4.Performance Evaluations

Here, we will give the performance evaluations of our scheme compared with other schemes such as "mirroring"[5], "interleaved declustering"[6], "rotational mirr red declustering"[11], and "Shift duplex with cyclic permutations"[9] in some detail. These performance gains, maximum stream capacity, and reject probability, are evaluated and analyzed, from which some distinguished features of our method can be explored.

4.1 Simulation Parameters

Several parameters will be given or defined appropriately for our simulation and evaluations. i) Arrive Rate: this denotes the time period of a request to arrive. The unit is millisecond. We will evaluate this value in 2083,

2604, 2997, 3470, and 4167ms. ii) Striping Factor: it specifies the number of physical disks used in a logical group. We will evaluate this factor in 4 and 8. iii) Replication pattern: we divide the total 100 videos int three parts. No.1 to no.30 is the hot test part, no.31 to no.60 is middle part, and no.61 to no.100 is cold part. The pattern "xyz" mean the hottest part with x replicas, middle part with y replicas, and cold part with z replicas. We will evaluate our method under the case of "321", "411", "421", and "511". iv) Load Factor: this value represents the proportion of streams exist in system compared to full capacity. We will evaluate our method for the case of 0.5, 0.6, 0.7, and 0.8.

4.2 Number of Replication

Before our evaluation, we will show the performance of different number of replication. For easy to discuss, we divide all videos sorted by their popularity with ascend int three parts—1% to 30%, 31% to 60%, and 61% to 100%. We will compare several replication patterns, includin "321", "411", "421", and "511". Figure 10 is the maximu stream comparison among different patterns. And figure 1 & 12 are reject probabilities both in normal and failure mode respectively. We can find that the performances of "421" and "411" are very close to each other. However, "321" is much worse than the other two. And "511" is superior than "421", but their difference is small. That is because "321" can't provide enough bandwidth for hot videos. So when the replica number of hot videos increase from 3 to 4, the performance will be improved a lot. And the performance is improved only a little from "411"t "421". So we can realize that the first part influence the performance deeply. Thus, we will use "411" to represent our scheme in the each of the following eva luations.

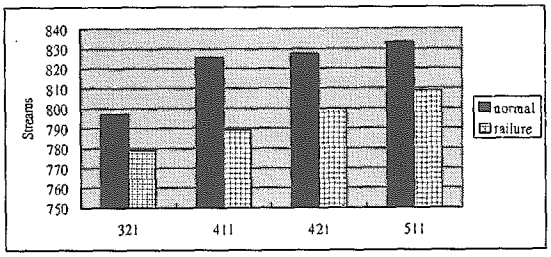

Figure 10. Maximum Stream comparison among different pattern

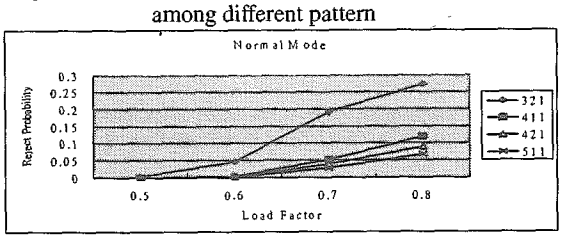

Figure 11. Reject Probability in Normal Mode

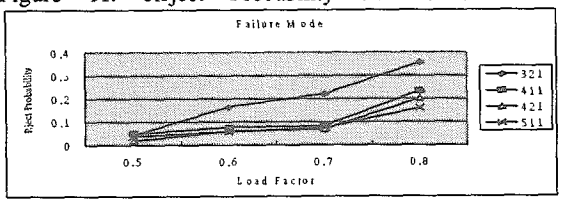

Figure 12. Reject Probability in Failure Mode

4.3 Influence of Striping Factor

As we mentioned above, the value of striping factor is suggested to be 4, 8, or 16. And we choose 4 as our default value. Theoretically, the performance of striping factor 8 should be superior than that of striping factor 4. From figure 13, we find the performance difference is only 0.5% in normal mode and 1.8% in failure mode. From figure 14, it is clearly that the performance with striping factor 8 is better than that of 4. But the gap is minor, especially under light load condition (e.g. load factor = 0.5 & 0.6). We choose 4 due to the low cost of adding videos.

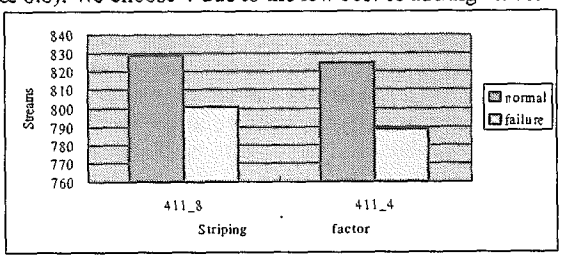

Figure 13.compare maximum in different striping factor

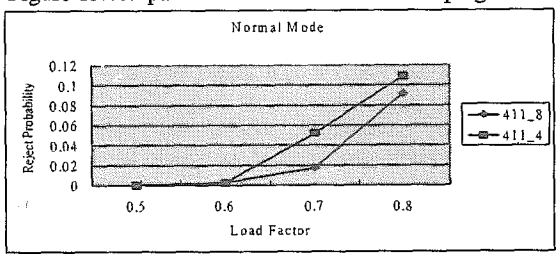

Figure 14(a) Striping factor in reject probabilit

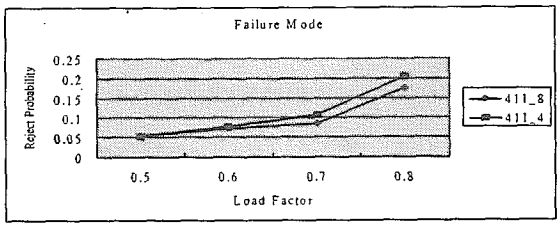

Figure 14(b) striping factor in reject probabilit

4.4 Comparison of TLC with other schemes

Figure 15 shows the evaluation of maximum strea capacity among those methods. From here, we can observe that the maximum streams of inter leaved declustering and mirroring are close to TLC under normal operation mode. But their performance under disk failure mode is pretty bad compared with TLC. The rotational mirrored declusterin has a stable performance no matter what operation modes. But its drawback is that although the performance under failure operation mode is not bad, the performance under normal mode still need to be improved more. It is necessar to have well performance in normal operation mode. Anyway, the system in normal mode i s much longer than failure mode. In contrast, the TLC and SDWCP are prett good under both modes.

From figure 16, we can see at load factor 0.8 that all value are close except TLC. That is because that there are all saturated and the value can't be improved any more. In failure mode, the performance gap is shown apparently. Thus, we conclude that TLC is superior than any other scheme in both modes and any load condition.

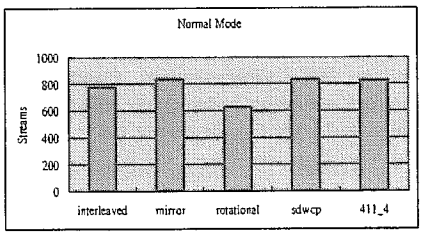

Figure 15.(a) maximum stream comparison

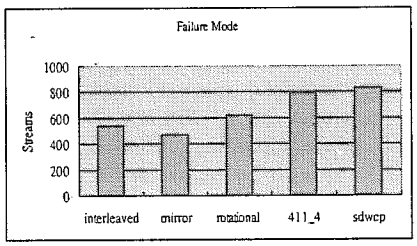

Figure 15(b) maximum stream comparison

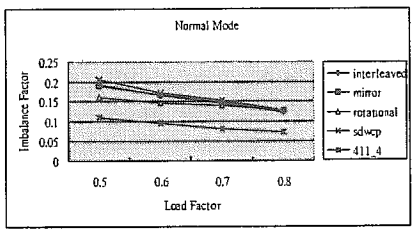

Figure 16(a) Imbalance factor in normal mode

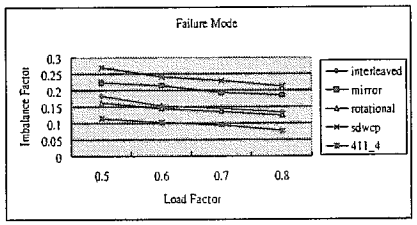

Figure16(b) Imbalance factor in failure mode

Last, figure 17 & 18 show the reject probability comparison in normal and failure mode respectively. Comparing these two figures, it is obvious that mirrorin method has fairly bad disk fault-tolerant effect. As usually, TLC performs better under both modes. The rotational mirrored declustering, prompting from worst in normal mode into the second one in failure mode, shows that it has well fault-tolerant ability than interleaved declustering and mirroring. From these two figures, we also find that in light load condition (50%), all schemes have very close performance, especially in normal mode. But the degradation to failure mode enlarges with the increasin load. Hence, our TLC is also better under this evaluation. In figure 19, it shows the reject probability when double failures, we can find that TLC is also better than other schemes under this evaluation.

So far, we have analyzed the basic performance of TLC and mirroring and parity check method. Then we will show the evaluation of TLC compared with SDWCP, mirroring, interleaved declustering, and rotational mirrored declustering. In summary, in normal operation mode, TLC has good performance compared with simple striping and mirroring. It means that TLC balances the load successfully. It reduces the imbalance factor to the minimum. In disk failure mode, TLC demonstrates even better performance. It spreads the loads on failure disk into nearly total disks successfully. So it shows better disk fault-tolerant ability in the heavy load condition. For TLC, the degradation of performance while disk failure is minimal. Although SDWCP has better performance than TLC, the difference is very limited. However TLC doesn't need an extra disk for failure condition.

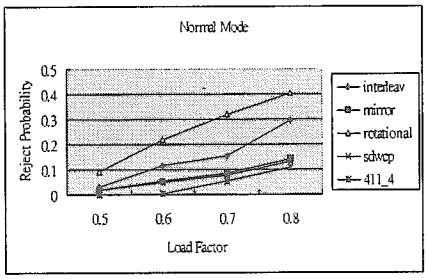

Figure 17. reject probability in normal mode

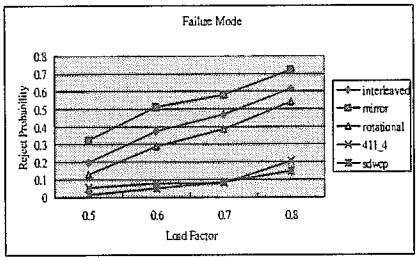

Figure 18. Reject probability in failure mode

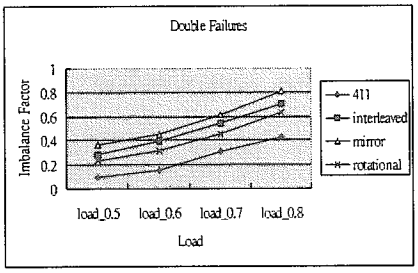

Figure 19. Reject Probability of double failures

5.Concluding Remarks

In this paper, we have described an effective data placement method and job scheduling policy with popularity-based replication technique. We also build a simulation environment to evaluate the performance of our methods and compared with other methods in some detail. In summary, the main features of our techniques are list below. i) A popularit -based replication concept is introduced. Because the behavior of clients, the popularity of each video are different. According to the popularity of the video, we assign it the different number of replicas. It is unnecessary to track and predict the load of every stream s as to reduce the cost and adjust the number of replicas effectively. ii) Our data placement method, named "Two Level Clustering", is a static placement method such that we take advantage of clustering technique twice to eliminate the negative factor caused by striping as w ell as attain the requirement of video server. By the evaluation of TLC, we find that TLC indeed reduces the influence of the highly skewed population distribution effectively, and hence, improves the load balance condition among DSGs. iii) Furthermore, an effective job scheduling scheme is als proposed to enhance the load distribution into more balance

condition. iv) Our placement provide a reliable disk fault-tolerant function. When disk failure occurs, our placement can spread the load on failed disk in to other disks balance so as to maintain the performance. Due to replication, our placement can tolerate multiple disk failures.

In the future, we may enhance our method to i) support VCR operation for client, ii) those heterogeneous video servers in the system, and iii) attach batch technique with fault-tolerance.

Acknowledgement

This work was supported by NSC 87-2622-E009-008

Reference

[1] T. L. Kunii, Y. Shinagawa, R. M. Paul, M. F. Khan, and A. A. Kbokhar, "Issues in storage and retrieval of multimedia data," *Multimedia Systems*, Vol. 3, No. 5/6, 1995, pp. 298-304.

[2] B. Ozden, R. Rastogi, A. Silberschatz, "On the design of a low cost video-on-demand storage system," *Multimedia systems*, Vol. 4, No. 1, 1996, pp. 40-54.

[3] Golubchik L, Lui JCS, Papadopouli M. "A survey of approaches to fault tolerant design of VOD servers techniques, analysis and comparison." *Parallel Computing*, vol.24, no.1, Jan. 1998, pp.123 -55. Elsevier, Netherlands.

[4] Thomas D. C. Little, Dinesh Venkatesh, "Popularity-Based Assignment of Movies to Storage Devices in a Video-on-Demand System". *Multimedia Systems*, Vol.2, No.6, pp.280 -287, Jan, 1995.

[5] D.Bitton and J.Gray. "Disk shadowing", *VLDB*, p 331-338. 1988.

[6] Leana Golubchik, Richard R. Muntz: Fault Tolerance Issues in Data Declustering for Parallel Database Systems. Data Engineering Bulletin 17(3): pp.14-28,1994.

[7] Jenwei Hsieh, Mengjou Lin, Jonathan C.L. Liu, and David H.C. Du "Performance of A Mass Storage System for Video-On-Demand," *INFOCOM'95 and A Special Issue on Multimedia Systems and Technology of Journal of Parallel and Distributed Processing*, Vol 30, No 2, Nov, 1995, pp.147-167

[8] Muntz R, Renato Santos J, Fabbrocino F. "Design of a fault tolerant real-time storage system for multimedia applications." *Proceedings of IEE International Computer Performance and Dependability Symposium.* IPDS'98.. IEEE Comput. Soc. 1998, pp.174-83. Los Alamitos, CA.

[9] Yuewei Wang, Du D.HC. "On providing highly available fault-tolerant video-on-demand services." Proceedings. IEEE International Conference on Multimedia Computing and Systems, IEEE Comput. Soc. 1998, pp.76-85. Los Alamitos, CA.

[10] Flynn R, Tetzlaff W. "Disk striping and block replication algorithmS for video file servers". *Proceedings of the International Conference on Multimedia Computing and Systems.* IEEE Comput. Soc. Press. 1996, pp.590 -597. Los Alamitos, CA, USA.

[11] Ming-Syan Chen, Hui-I Hsiao, Chung-Sheng Li, Yu PS. "Using rotational mirrored declustering for replica placement in a disk-array-based video server". *Multimedia Systems*, vol.5, no.6, Dec. 1997, pp.371-379. Springer-Verlag, Germany.

[12] Copeland G, Keller T, "A comparison of high-availability media recovery techniquis". *Procedings of ACM ISGMOD*, Portland, OR, pp.98-109, 1989.

[13] B. Ozden, R. Rastogi and A. Siberschatz. "Disk striping in video server envi ronments," *Data Engineering*, Vol 18. No. 4, Dec, 1995, pp. 4-16.

[14] Wang Y, Liu JCL, Du DHC, Hsieh J. "Efficient video file allocation schemes for video-on-demand services". *Multimedia Systems*, Vol.5, No.5, Sept. 1997, pp.283-296. Springer-Verlag, Germany.

[15] J.K.Chen,C.Chen,S.Y.Lee, "CSSP : An Effective Static Content Placement Policy for Load-balancing in Video Server", *Proceedings of Workshop On Computer Networks, Internet, and Multimedia (ICS'98)*, 1998, pp.104-110.

[16] *Video Server User Guide*, Institute of Mentor Data System, 1998.

[17] Steven Berson, Leana Golubchik, Richard R. Muntz, "Fault Tolerant Design of Multimedi Servers". *SIGMOD Conference*, 1995, pp.364-375.

[18] Asit Dan, Martin G. Kienzle, Dinkar Sitaram," Dynamic Policy of Segment Replication for Load-Balancing in Video-On-Demand Servers". *Multimedia Systems*, Vol.3, No.3: 93-103, Jul, 1995.

[19] Thomas H. Cormen, Charles E. Leiserson, *Introduction to Algorithms*, 1996, pp.600-604.

[20] W.S.Huang, *An Effective Data Placement Scheme to Support Fault-Tolerance in Distributed Video Server Environment*, Master Thesis, CSIE, NCTU, 1999