1998 International Computer Symposium
Workshop on Computer Networks, Internet, and Multimedia
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

CSSP : An effective Static Content Placement Policy for Load-balancing in Video Server

Ju-Kau Chen, Wen-Shih Huang, Cheng Chen* and Suh-Yin Lee
Department of Computer Science and Information Engineering
National Chiao Tung University,Hsinchu,Taiwan,R.O.C
Email : cchen@csie.nctu.edu.tw

Abstract.

It is well known that a high performance video server plays more and more important role in VOD system design. Thus, in this paper, we have proposed a static content placement method named content segment striping policy (CSSP) to balance the load of the disks, thereby helping to solve a load imbalance problem crucial to archiving maximal server throughput. This scheme is simple and easy to implement. It has been built in our video server management system platform. According to our preliminary performance evaluations, we have found that our policy can decrease the reject probability of requests and improve the service capacity of video server (more than 90% of maximal service capacity) effectively compared with other similar methods.
Key words: VOD, video server, CSSP, load imbalance, reject probability, service capacity, content placement methods, disk striping group.

1. Introduction

Recently, advances in computing and communication technologies make it possible to provide a wide range of interactive multimedia services in a variety of commercial and entertainment domain [1][2]. One of the most visible applications of multimedia systems is on-demand playback of video in a distribution environment [3][4]. In the video-on-demand system, video server must store video contents on disks and provide video-on-demand services to many clients over a network.
In the video server, the data placement policy is made more complicated by the fact that some videos are vastly more popular than others at any given time [5]. Furthermore, this highly skewed distribution varies on a weekly, daily, and even hourly basis due to the changing video popularity and customer mix. The popularity of the hottest videos can often be so great that storing them on a single disk may not be feasible from a performance standpoint. In order to balance the load among the disks, video data will be spread over all the disks appropriately. However, a simple striping approach has several limitations [6], such as the difficulty of supporting multiple types of disks and the complexity of system reconfiguration (e.q. add new disks). Furthermore, when a single disk fail, simple striping policy will lead to complex recovery procedure and unavailability of video data. These tradeoffs imply that the degree of striping should be limited to some extent. Thus it is natural to group multiple disks into separate striping groups. By combing, for example, each group of 8 disks into a 8-way disk striping group (DSG) the load generated by each video stream can be cut correspondingly, and the overall load across each of the 8 disks essentially balanced. The number of streams that a single DSG can support, N_{seg}, depends on the number of disks and the bandwidth of each disk. We can derive the maximal service capacity of video server, N_{total}, as equation (1). Nevertheless, most of the time, the maximum number of streams that being serviced concurrently can not be achieved due to the load imbalance of the DSGs.

$$N_{total} = D_{total} / D_{seg} \times N_{seg} \text{ , where } D_{total} \text{ is total disk number of server} \qquad (1)$$
D_{seg} *is the striping degree of DSG*

To balance the load of all DSGs, one of the simplest ways is to arrange all contents to DSGs according to their popularity [7]. In such a way that the summation of the probabilities of the contents being requested from each DSG is as equal as possible. By this method, we will assume that the popularity of all the contents are expectable in a certain period of time and all the contents have approximate size. Basically, there are three popular methods used to allocate the contents on each DSG [7], named round and round, round and backward and round and shift. Although all the three methods can make the server requests on each DSG in a generally balanced condition, unfortunately, the popularity of each content changes quite often and we may load other hot contents into disks by replacing some contents that are not popular now. After these swap in/out operations and the changing of popularity, the balanced condition of the video server will be skewed. Hence, it is important to design an efficient placement technique to solve the load imbalance problem for video server design.
In this paper, an effective static content placement techniques named content segment striping policy, CSSP, is proposed to optimize the service capacity of video server as

well as balance the load among DSGs appropriately. It partitions each video content into several fix-size segments and then allocates those segments into different DSGs according to content requested probability. Here, each segment is composed by hundreds of continuous logical blocks of DSG as shown in Figure 1. We assume that video contents are normally viewed in a sequential manner, and thus, each stream will play from first segment, then second segment, until the last segment of content. Hence, if we can strip the sequential segments of each content to different DSGs, the loading of each content can be dispersed to different DSGs instead of a single DSG. By our evaluations, we find that this policy will have better load balance condition compared with other previous methods, and hence, lead to a lower reject probability of requests and higher service capacity of video server.

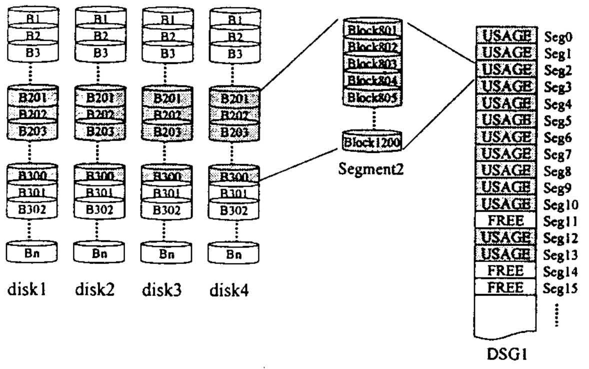

Figure 1 Segment of DSG

The remainder of the paper is organized as follows. In section 2, we will define a load imbalance factor of DSGs for judging reference point of content placement techniques. In section 3, our static content placement policy, CSSP, will be introduced and analyzed. In section 4, the implementation of our video server platform and the evaluation of system performance will be described. Finally, concluding remarks will be given in section 5.

2. Load Imbalance Factor

Before discussing our techniques to minimize load imbalance across DSGs, we will define a measure factor L_p to represent imbalance condition of the video server. Let M denotes the number of disk resident contents and G denotes the number of DSG. C_i is used to denote the *ith* content and DSG_j is used to denote the *jth* DSG. Associated with each content, P_i denotes the probability that a new client will request the *ith* content. Let $A_{i,j}$ denotes the assignments of the *ith* content to *jth* DSG. Thus A is a {0,1} $M \times G$ matrix defined as

$$A_{i,j} = \begin{cases} 1 & \text{if content i exists in DSG j} \\ 0 & \text{otherwise} \end{cases}$$

For DSG_j, the hit probability of the *jth* DSG, H_j, is defined as the summation of the probabilities that the contents in DSG_j be requested.

$$H_j = \sum_{i=1}^{M} P_i \times A_{i,j} \tag{2}$$

Hence, while total client requests, R, are collected, there may have $R \times H_j$ requests that reside in DSG_j. And then we can define the server balance factor L_p as:

$$L_p(DSG_1, DSG_2, \ldots DSG_G) = \frac{StdDev(DSG)}{Exp(DSG)}\text{, where} \tag{3}$$

$$Exp(DSG) = \frac{\sum_{i=1}^{G} DSG_i}{G} \cong \frac{\sum_{i=1}^{G} R \times H_i}{G} \text{ and} \tag{4}$$

$$StdDev(DSG) = \left[\frac{\sum_{i=1}^{G} (DSG_i - Exp(DSG)^2)}{G} \right]^{1/2} \tag{5}$$

In the following, L_p is used to judge the load balance condition of server. Basically, a smaller L_p value represents that the hit probability for each DSG is approximate equal and stands for a well load balance condition of server. On the contrary, a larger L_p value represents that the hit probability for each DSG is skewed and stands for a worse load balance condition of server. As described before, a worse load balance condition implies the higher reject probability of new request and more residual service capacity must be unemployed. Therefore, in the following sections, we will try to minimize the L_p value of server by using effective content placement techniques.

3. Content Segment Striping Policy

As described before, by using those initial placement methods to balance the load across DSGs, we assume that the popularity of all contents are expectable in a certain period of time and all the contents have approximate size. For the first term, it is hard to maintain the same popularity in real environment due to the changing of content popularity and customer mix. To solve this highly changing skewed distribution, several researches proposed to create multiple copies of some hot contents to store in different DSGs. Thus, however, it need more storage space and will be inconvenient rearrangement of content assignments. In order to overcome this drawback, we try to find another static placement method to place the video contents efficiently. Before describing our method, we analyze the influence of content length at first.

In a video server, stored video contents may have different content size. Assume that the request arrival process of client is a Poisson process with constant arrival rate λ. For each content C_i, L_i denotes the content size in block. The playback time of the *ith* content, T_i, can be derived by using equation (6).

$$T_i = L_i \times B_{block} / V_{play} \tag{6}$$

And the request arrival rate of the *ith* content will be $\lambda \times P_i$. Let N_i be the number of clients retrieving the *ith* content concurrently. According to the formula one of query theory, we can get the following equation directly.

$$N_i = \lambda \times P_i \times L_i \times B_{block} / V_{play} \propto P_i \times L_i \qquad (7)$$

From equation (7), we can find that the number of clients retrieving the *ith* content will be direct proportion to request probability and content length. If we arrange the contents in DSGs according to their request probabilities, the effect on load balance must be decreased due to the influence of content length. We show this by an example in Figure 2. There are four contents *A*, *B*, *C* and *D*. Suppose that P_A, P_B, P_C, P_D are the probabilities of being requested for the content *A*, *B*, *C* and *D* respectively and can be sorted as $P_C(0.4) > P_A(0.3) > P_D(0.2) > P_B(0.1)$.

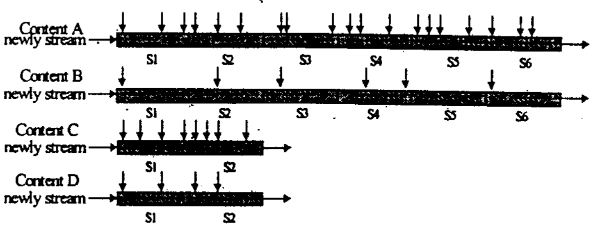

Figure 2 Example of the request condition of four contents

According to the round and round initial placement method, we will assign them as shown in Figure 3. Although the content *C* has the larger approximate request probability than that of the content *A*, the load of DSG_1 is approximate twicefold to DSG_2. This is because the content length of *A* is threefold to *B*. The similar situation also happens on DSG_3 and DSG_4.

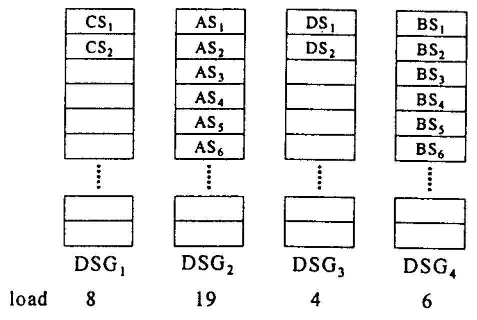

Figure 3 Placement by round and round method

In order to solve this problem, let's reexamine the retrieval characteristics of DSG as shown in Figure 2 and Figure 3 in some detail. We can find that for content *A*, which is resided in DSG_1, the number of retrieving clients for each segment of content *A* seems similar (about 3 clients per segment). This is because that the request arrival rate is a Poisson process and each segment, AS_1, AS_2, AS_3, AS_4, AS_5 and AS_6, belong to content *A* will have the same request probability P_A. That means that $P_{Aseg1} \cong P_{Aseg2} \cdots \cong P_{Aseg6} \cong P_A \cong 0.3$, where P_{Aseg1}, P_{Aseg2}, …, and P_{Aseg6} are the request probabilities of the AS_1, AS_2, …, and AS_6 respectively. Therefore, for each segment, AS_1, AS_2, AS_3, AS_4, AS_5 and AS_6, the number of retrieving clients may be $\lambda \times P_A \times S \times B_{block} / V_{play}$ (i.e. arrival rate × service time). The similar condition will also happen on content *B*, *C* and *D*. Hence, we can derive a general equation to calculate the retrieving number of client of each segment belong to content C_i, $1 < i < M$, by the following formula.

$$Nseg_i = \lambda \times P_i \times S \times B_{block} / V_{play} \propto P_i \qquad (8)$$

From equation (8), if we partition content C_i into several fixed size segments and allocate them to different DSGs appropriately, the retrieving streams of content C_i can be dispersed to different DSGs instead of a single DSG. However, how to arrange segments of content into different DSGs is one of the most important topics. Let's redefine the *A* as a {0,1,2,..} $M \times G$ matrix as follows:

$$A_{i,j} = \begin{cases} k & \text{if there are } k \text{ segments of content i in DSGj} \\ 0 & \text{otherwise} \end{cases}$$

We call the matrix *A* as content segment allocation matrix. From equation (2), in order to minimize the load imbalance factor, L_p, among DSGs, we will make the hit probability, H_j, for each DSG *j*, $1 < j < G$, as equal as possible. Considering the definition of *A*, if we can arrange the segments of each content to all DSGs as fair as possible, we can minimize the variance of the hit probability for each DSG. Based on this concept, we propose a content striping policy (CSSP) for static content placement. Figure 4 illustrates the principle of our method by using previous example. As described before, each content is partitioned into several fixed size segments and the probability being requested for each segment is listed as follows:

$$P_{Aseg\ 1} \cong P_{Aseg\ 2} \cdots \cong P_{Aseg\ 6} \cong P_A \cong 0.3$$

$$P_{Bseg1} \cong P_{Bseg2} \cdots \cong P_{Bseg6} \cong P_B \cong 0.1$$

$$P_{Cseg1} \cong P_{Cseg2} \cong P_C \cong 0.4$$

$$P_{Dseg\ 1} \cong P_{Dseg\ 2} \cong P_D \cong 0.2$$

Thus, the probabilities being requested for all segments can be ordered as:

$$P_{Cseg1} \cong P_{Cseg2} > P_{Aseg1} \cong P_{Aseg2} \cdots \cong P_{Aseg6} >$$

$$P_{Dseg1} \cong P_{Dseg2} > P_{Bseg1} \cong P_{Bseg2} \cdots \cong P_{Bseg6}$$

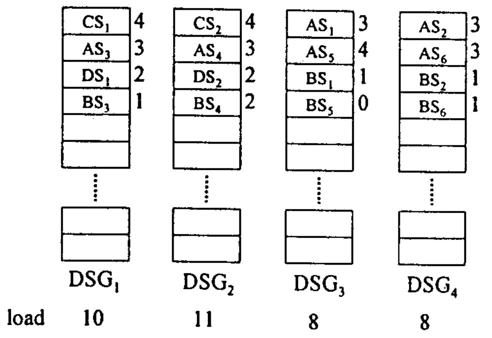

Figure 4 Placement by content segment striping method

Then, as shown in Figure 4, we can stripe segments sequentially into all DSGs by round robin method according to the probabilities being requested. And the matrix A is given by: $[A]=\begin{pmatrix}1&1&2&2\\1&1&2&2\\1&1&0&0\\1&1&0&0\end{pmatrix}$

By the above description, we can summarize our CSSP formally in the following:

Assuming that there are M video contents, C_1, C_2, C_3, ..., C_M, and P_1, P_2, P_3,..., P_M are their corresponding probabilities of being requested such that $P_1 > P_2 > P_3 > \ldots > P_M$.

(1) Selecting an appropriate segment size.
(2) Partitioning content C_1, into fix-size segments, S_{C1seg1}, S_{C1seg2}, S_{C1seg3}, ..., S_{C1segm}, where $S_{ciseg j}$ denotes the j th segment of content C_i.
(3) Placing the segments of content C_1 to all DSGs by round robin method such that S_{C1seg1} is placed on DSG_1, S_{C1seg2} on DSG_2, S_{C1seg3} on DSG_3, ..., and so forth.
(4) For each content, C_2, C_3, C_4, ..., C_M, repeats steps (2) and (3) until all contents are placed in DSGs or there are no sufficient storage space.

Obviously, the complexity of this method is $O(M)$ which is the same as that of initial placement methods. As for selecting the segment size, we will discuss and analyze it later.

4. System Implementation and Performance Evaluations

As shown in Figure 5, the overall configuration of our platform has two parts: one is video server engine and the other is video server management system. The video server engine produced by Mentor Data System Inc. is used to control hardware components of video server [8] includes storage devices, memory buffer and network ports. Basically, storage devices are used to store video contents and retrieve it on demand. Retrieved video data are stored in memory buffer and will be consumed by client continuously. The video server engine used Intel RISC CPU with 96MB memory and 10 fast & wide ports. It has total 16 4GB hard disks and 20 Ethernet 10 Base-T ports.

Based on the video server engine, we have developed a useful and effective video server management system for VOD or near VOD applications. In our system, there are five main function modules as shown in Figure 5. The admission control module is the doorkeeper of video server. With the purpose of keeping service quality of viewing streams, it may guarantee that the service quality will not be degraded or be contravened by newly incoming request. So, the newly request should be accepted if and only if the admission constraint is not violated. While the newly request is accepted, it will be added into stream management table and video server will serve the request immediately. To keep accepted streams for continuous playing, job schedule module is used to control the video server engine to retrieve video data from disks and then buffer them in a cycle fashion. Meanwhile, disk schedule module is used to optimize the reading performance of disks by rearranging the order of retrieval commands from job schedule module. As described before, video server engine adopts multiple disks as storage device to store hundreds of contents. The resource management table will maintain the information of disk storage space and hundreds stored contents. Data placement module will control how to arrange the contents into disk storage space. And the CD/HD swapping module responses to update and replace the contents from CD-ROM into disks. So far, the while platform can be speared normally. In the following subsections, we will discuss the performance evaluations about our CSSP by running several video data on our platform with some reasonable input models.

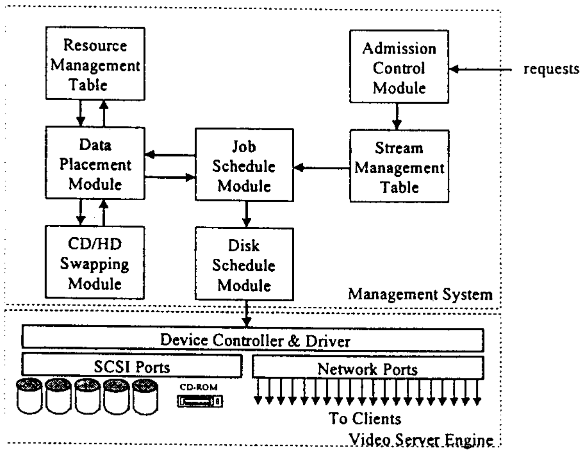

Figure 5 Overview of Platform

4.1 Evaluation System Parameters

The video contents we have tested are coded by using the MPEG-II with play rate, V_{play}, 3Mps for full-motion compression. In our server, we have adopted several contents types includes news, KTV and movies all borrowed from Mentor Data System Inc.. Table 1 lists the detail information of those contents.

Content Types	Content Size (blocks)	Total Number
News	50 ~ 1000	50
KTV	600 ~ 800	100
Movie	5000 ~ 7000	12

Table 1 Video content types

To test and evaluate our platform, we have to develop a request generator to simulate the input requests from the network. Without loss of generality, we assume that requested content ID is Zipf-like distribution and the input arrival rate is Poisson distribution. The Zipf-like distribution can be generated by using function $Z(C_i)$ and we can adjust the value of θ to obtain different data skew.

$$Z(C_i) = c / i^{(1-\theta)} \quad ,where$$

$$c = 1 / \sum_{k=1}^{M} (1/k^{(1-\theta)}) \tag{9}$$

M : total number of videos in disks.
C_i *: ith most frequently accessed content*

The request generator will check for the current time every 1-sec., and decides whether it will start a new stream. If the answer is yes, it will add a new node to the stream list. Then the content ID will be decided by the content popularity distribution. Also, the content information can be retrieved from the content table and the elapsed time will be cleared to zero. Formally, if we let T_{round} be the time of the job scheduling cycle, then all requests generated in $[n\ T_{round},\ (n+1)\ T_{round})$ are handled by the job scheduler at $(n+1)\ T_{round}$.

For our platform, there are 16 disks in the server and each disk stoage capacity is 4GB. The block size of disk, B_{block}, is 94 Kbytes. We will use 2-way disk striping group to group 16 disks into 8 disk striping group (DSG). Note that, however, we can use 4-way or 8-way disk striping group instead of 2-way disk striping group if necessary. The storage capacity of each DSG is 8GB (i.e. 80000 logical blocks) and can support 20 streams playback concurrently. In the following subsections, we will generate 5000 request for each evaluation experiment and some performance gains will be collected and analyzed.

4.2 Evaluations of Load Imbalance Factor

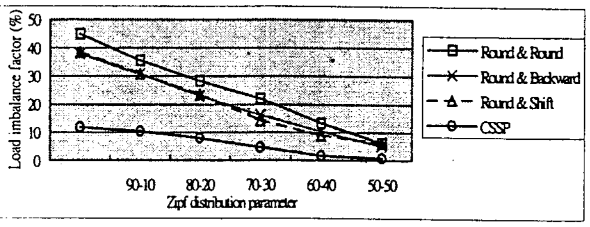

Figure 6 Evaluation of CSSP with content length 3200 blocks

Figure 6 and Figure 7 show the load imbalance factors for CSSP and initial placement methods based on 3200 blocks and 6400 blocks content length respectively. We can find that CSSP performs better than those initial placement methods because it spreads the load of contents out on different DSGs and thus, the influence of highly skewed popularity distribution is decreased. Moreover, with the increase in the popularity distribution skew, we also find that the performance distance between CSSP and initial placement methods will enlarge apparently. For example, in Figure 10, while the Zipf-like distribution parameter is 60-40, the distance between CSSP (1.64%) and initial placement methods (8.94%) is about 7.3%. However, while the Zipf-like distribution parameter is 90-10, the distance between CSSP (10.38%) and initial placement methods (30.78%) is enlarged to about 20.4%. Hence, while the popularity distribution skew increases, the increase in load imbalance factor of CSSP is slower than those of initial placement methods. That is to say that CSSP has better resistance to the highly skewed popularity distribution.

From Figure 6 and Figure 7, we can also find that CSSP performs better while the content length becomes larger. It is because that with larger content size we can spread each content to more DSGs and then get better load balance condition. Figure 8 and Figure 9 show this condition based on 80-20 and 70-30 Zipf-like popularity distribution respectively.

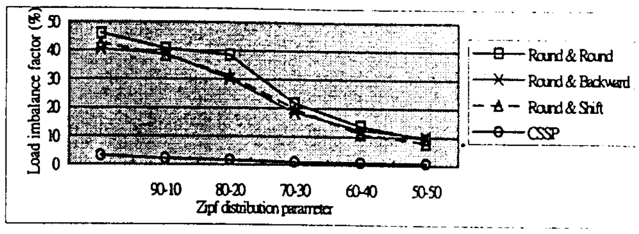

Figure 7 Evaluation of CSSP with content length 6400 blocks

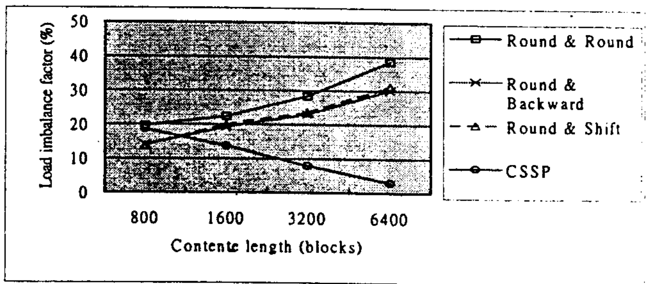

Figure 8 Evaluation of CSSP with 80-20 Zipf-like distribution

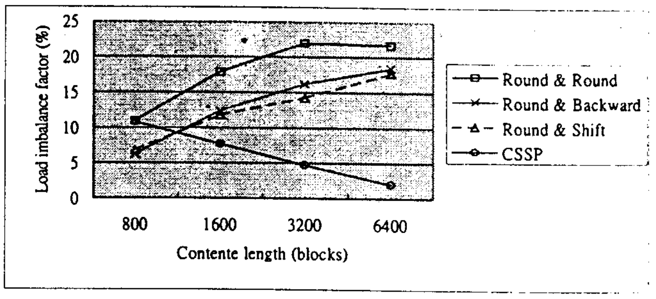

Figure 9 Evaluation of CSSP with 70-30 Zipf-like distribution

4.3 Selection of Segment Size

To select the size of segment properly is another important issue for our CSSP. We will consider it from both the performance and the complexity (size and calculate complexity of file system) view point. With a small segment size, we can spread each content out on more DSGs and get better load balance condition as shown in Figure 10 while the content length is 1600 blocks. Furthermore, the overhead of fragmentation will be reduced due to smaller segment size. However, spreading content extravagantly with too small segment size will result in few improvements on server load balance condition and also lead to larger size of the FAT of file system. From Figure 6 to Figure 10, we can find that the CSSP performs well while most of the contents can spread out on from half to all of DSGs (i.e. $G/2 \le L/S \le G$). Thus, for a server that stores contents with average L blocks size and has total G DSGs, selecting a segment size S such that $L/G \le S \le 2L/G$ is better. Besides, storage space is another consideration on selection of segment size. A video server with large storage space, we will adopt larger segment size (i.e. $2L/G$) to minimize the size FAT. On the contrary, a server with small storage space, we can adopt smaller segment size (i.e. L/G) to use storage space efficiently.

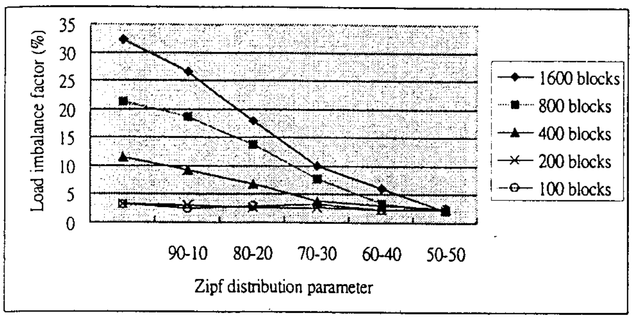

Figure 10 CSSP with different segment size while content length=1600 blocks

4.4 Improvement of Performance

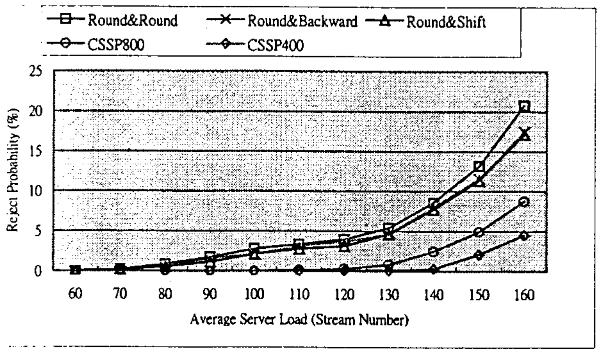

Figure 11 Reject probability of requests with 80-20 Zipf-like distribution

Due to that each DSG can support 20 streams, the maximal service capacity of server is 160 streams in theory. However, the load imbalance among DSG' will cause the server has higher reject probability and more residual service capacity must be unemployed (lower service capacity utilization). In this section, several simulations are performed to show that the CSSP can decrease the reject probability and improve the service capacity of video server effectively.

Figure 11 and Figure 12 shows the fractions of requests rejected and average service capacity of video server respectively while content length is 3200 blocks and Zipf-like distribution is 80-20. Here, the CSSP800 and CSSP400 denote the CSSP with segment size 800 blocks and 400 blocks respectively. For all of the policies, the load of each DSG increases with the increase in the server load, and hence, increases the reject probability of request due to the load imbalance among DSGs. However, we can find that both CSSP800 and CSSP400 have lower reject probability than those of initial placement methods as shown in Figure 17. This is because that CSSP800 and CSSP400 have better load balance condition among DSGs, and hence, avoid the overload of few DSGs efficiently. Moreover, with better load balance condition, CSSP800 and CSSP400 can improve the average service capacities of video server to 91.28% and 95.08% of the maximal service capacity (160 streams) respectively.

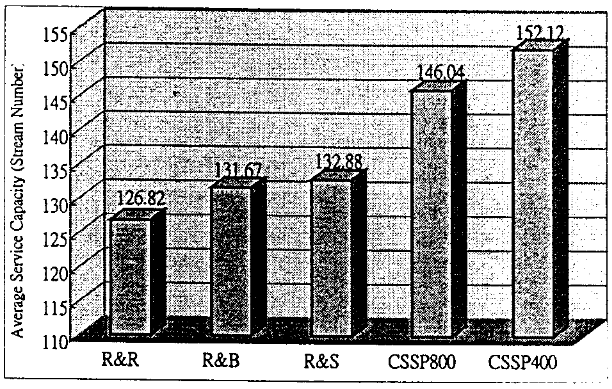

Figure 12 Average service capacity of server with 80-20 Zipf-like distribution

5. Concluding Remarks

In the on-demand video server environment, the use of disk striping groups makes it necessary to deal with the problem of load imbalance among striping groups. In this paper, we propose a static placement policy, content segment striping policy, to handle the load imbalance across DSGs. It partitions the contents into a small number of segments and striping them to all of DSGs to balance the load. We can find that the load imbalance due to highly skewed population distribution and content length will be decreased apparently. Furthermore, due to we can avoid the load imbalance among DSGs, the reject probability of request can be decreased apparently. Meanwhile, the average service capacity of video server can be enhanced to more than 90% of maximal service capacity. In the future, we will focus on that how to design an effective dynamic exchange technique to improve the load imbalance condition further.

Acknowledgement

This research was supported by the National Science Council of the Republic of China under contract numbers : NSC85-2622-E009-006R.

References

[1] V. Rangan and H. M. Vin, "Efficient storage techniques for digital continuous multimedia," IEEE Trans. On Knowledge and Data Engineering, Vol. 5, No. 4, August 1993, pp. 564-573.

[2] 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.

[3] A. N. Mourad, "Issues in the design of a storage server for video-on-demand," Multimedia Systems, Vol.4, No. 2, 1996, pp. 70-86.

[4] 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.

[5] J. L. Wolf, P. S. Yu, and H. Shachnai, "DASD Dancing: a disk load balancing optimization scheme for video-on-demand computer systems," Proc. Of the ACM SIGMETRICS 5, 1995, pp.157-166.

[6] A. Dan, M. Kienzle, and D. Sitaram, "A dynamic policy of segment replication for load-balancing in video-on-demand servers," Multimedia Systems, Vol. 3, No. 3, 1995, pp. 93-103.

[7] T. D. C. Little and D. Venkatesh, "Popularity-based assignment of movies to storage devices in a video-on-demand system," Multimedia Systems, Vol. 2, No. 6, 1995, pp. 280-287.

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