

ON THE EFFICIENT USE OF VIDEO-ON-DEMAND STORAGE FACILITY

Jun Guo, Peter Taylor, Moshe Zukerman

ARC Special Research Centre
for Ultra-Broadband Information Networks
Department of E&E Engineering
The University of Melbourne
Melbourne, VIC 3010, Australia

Sammy Chan, K. S. Tang, Eric W. M. Wong

Department of Electronic Engineering
City University of Hong Kong
Tat Chee Avenue, Kowloon
Hong Kong SAR
People's Republic of China

ABSTRACT

We consider a Video-on-Demand system in which multiple copies of each movie file are kept in separate disks. Various schemes for file allocation and retrieval are considered and compared. We have introduced a certain easy-to-implement scheme that requires 15% more storage than another certain harder-to-implement scheme.

1. INTRODUCTION

Advances in high-speed networks and multimedia technologies have enabled proliferation of Video-on-Demand (VoD) interactive services [1,2]. For the purpose of providing access to a large variety of movie contents, a typical VoD system is expected to have a very large disk farm. Each disk normally stores a range of large movie files encoded in MPEG format. Once a user request arrives, the requested movie file, in the form of a data stream, is transmitted from its disk to the user. For MPEG-2, each file typically needs 5.5 Mbps I/O bandwidth for transmission. Due to the large file size, the connection time of movie files is measured in hours. Moreover, in order to provide VCR-like interactive services, a single I/O stream is allotted to each user. Since the I/O bandwidth of commercially available disks is usually in the order of megabytes per second, the number of I/O streams a disk can provide is limited. As a result, in a VoD system, the number of requests a disk can serve concurrently is also limited. Because of the long-lived connection, it is not possible to queue the requests. We therefore consider a VoD system as a loss system. That is, a user request for a particular movie file is blocked if the system finds that all I/O stream resources of the file's residing disk are fully engaged. Therefore, to meet Quality of Service

and reliability requirements, multiple copies of movie files are stored on separate disks. We will use the terms *file* and *movie file* interchangeably throughout the paper.

Without exception, the allocation of movie files and their replicas onto the disks can significantly affect the overall performance of the VoD system. Though the file assignment problem (FAP) in parallel systems has received significant attention since the 1970's (see [3,4] and references therein), as far as we are aware, the algorithms proposed in the literature typically aim at minimizing the mean file request response time. These algorithms are suitable for FAPs of traditional file server systems, where files normally have small file size, short-lived connection. They have no on-demand transmission bandwidth requirements. Tang *et al.* [5] studied a FAP model where the blocking probability is used as the performance metric. However, the model of [5] rules out the retry mechanism. That is, when a request for a file arrives, one of the disks storing that requested file is chosen at random. If for the selected disk all the I/O streams are busy, the request will be blocked. No attempt to retrieve a replica of that file from other disks is made. In contrast, in this paper, all the relevant disks may be attempted before the request is blocked. In general, the problem of optimizing file assignment subject to blocking probability requirement is not tractable simply because exact blocking probability is not obtainable except for rare cases of file assignment. Therefore, we have to rely on heuristics. This paper aims to gain insight into what constitutes an efficient file assignment and retrieval scheme to support development of algorithms for VoD system optimization. To this end, we consider several file assignment and retrieval schemes and study their performance and efficiency.

2. THE MODEL

Consider a VoD system with N disks. Each disk has S I/O streams, henceforth called *channels*. Arrivals of requests for file i follow a Poisson process with rate λ_i and the connec-

The work described in this paper was partially supported by grants from City University of Hong Kong (Project No. 7001224 and Project No. 7001458) and partially supported by the Australian Research Council. J. Guo and M. Zukerman are visiting the Department of Electronic Engineering, City University of Hong Kong, between November 2002 and July 2003.

tion time of file i is generally distributed with mean $1/\mu_i$. Denote the traffic demand on file i by $A_i = \lambda_i/\mu_i$.

Let c_i be the number of copies required for file i . In particular, if $c_i = j$, we say that file i is of type j , and all these j copies must be stored on j different disks. In this paper, we shall take the c_i to be given. Normally, c_i is set in accordance with movie file popularity and reliability requirements.

Let $F(j)$ be the set of all type j files. We assume that no file requires more than J copies so j takes the values between 1 and J . Define $A(j) = \sum_{i \in F(j)} A_i$.

3. FILE ASSIGNMENT SCHEMES

We will now describe three file assignment schemes and, if applicable, their associated file retrieval mechanism.

3.1. Non-overlapping Load Balancing (NLB)

This heuristic and easy-to-implement scheme is based on a given worst case blocking probability B . For each file type j , we allocate an exclusive set of disks $D(j)$ in which we store all the files of the set $F(j)$. One copy of the first file in $F(j)$ is stored on each of the first j disks of $D(j)$. The same is true for the second and third files of the set $F(j)$. This process is continued until adding extra files to the first j disks violates the blocking probability constraint. We then start allocating files to disks $j+1$ to $2j$ of the set $D(j)$ until the blocking probability constraint is violated again and so on. Let n be the number of groups of j disks that we need in this method. Thus, there are nj disks in $D(j)$.

Each group of j disks has Sj channels and, for performance purposes, we can think of this group of disks as an Erlang system with Sj servers. The probability $B(j)$ that a file requiring j copies cannot be accessed is thus given by

$$B(j) = E(jS, a(j))$$

where $a(j)$ is traffic generated by access requests to each group of disks and

$$E(k, \rho) = \frac{\rho^k/k!}{\sum_{\ell=0}^k \rho^\ell/\ell!}$$

is Erlang's B Formula [6]. The reader is reminded that Erlang formula is insensitive to holding time (in our case connection time) distribution. An accurate evaluation for $a(j)$, is given by

$$a(j) = \frac{A(j)}{n}$$

Given $A(j)$, we can evaluate the number $N(j)$ of disks that we need to accommodate files requiring j copies by calculating

$$\min_n : E(jS, A(j)/n) \leq B$$

Number of copies	Proportion of traffic load
1	0.07
2	0.13
3	0.20
4	0.27
5	0.33

Table 1. Load distribution among different file types

and then putting $N(j) = jn$.

For NLB, there is no issue of optimal file retrieval mechanism because all copies of the same file share the same group of channels. Therefore, it does not matter which copy of a particular file is chosen first.

3.2. Combination Load Balancing (CLB)

While NLB is based on a given worst case blocking probability B , this scheme is based on a given number of disks N . To assign type j files, we group the N disks into $\binom{N}{j}$ combinations. Each of these combinations is assigned a set of type j files such that the traffic load on all these combination groups is equal. Clearly, such perfect load balancing is unlikely to be achieved in practice, and we will aim for near optimal solution. This procedure is then repeated for all j , $j = 1, 2, 3, \dots, J$. As will be numerically demonstrated later in Section 4, CLB is more efficient than NLB. Although CLB is much harder to implement, it provides a performance benchmark.

Unlike NLB, the performance of CLB is sensitive to the file retrieval mechanism chosen. We consider here three file retrieval mechanisms.

1. Single Random Trial (SRT)

After a file request arrives and the set of disks is determined, we choose one of these disks at random. If all the channels on this chosen disk are busy, the request is rejected. No further attempt will be made to try other disks. This is the method discussed in [5].

2. Repeated Random Trials (RRT)

Under this mechanism, the first step is as in SRT. However, if all the channels on the chosen disk are busy, we continue with repeated trials in all remaining disks of the set until we are successful, or if we check all the disks of the set and fail to find access to any of them, the request is blocked.

3. Least Busy Fit (LBF)

If all the channels on the set of disks are busy, the request is blocked. Otherwise, according to LBF, we shall choose the least busy disk, that is, the disk with the most unused channels. If there is more than one disk with the minimal number of busy channels, we choose one of them randomly.

Number of Disks	Traffic Load (Erlang)	Type 1	Type 2	Type 3	Type 4	Type 5
7	100	0.000717	0.000126	0.000057	0.000030	0.000025
13	200	0.001031	0.000121	0.000030	0.000021	0.000005
18	300	0.009939	0.001524	0.000427	0.000171	0.000074
24	400	0.006795	0.000713	0.000152	0.000054	0.000016
30	500	0.004983	0.000357	0.000080	0.000017	0.000005

Table 2. CLB: Blocking probability of different file types under various traffic loads

3.3. Non-overlapping CLB (NCLB)

One problem with the CLB scheme is that it does not allow control of the blocking probability of individual file types. In fact, CLB does not even provide fairness with regard to blocking probability.

NLB does allow full control of the blocking probability of each file type; however, as will be shown in Section 4, it is less efficient than CLB. A compromise could be in a new file assignment scheme we call *Non-overlapping CLB* (NCLB). Under this scheme, we first find the minimal number of disks required for all type 1 files to meet a given blocking probability requirement using CLB, and assign them to the type 1 files. Then, we repeat the process separately for each file type. The advantage of NCLB over CLB is that the blocking probability for each file type can be controlled.

4. NUMERICAL RESULTS

To compare between the three file assignment schemes and their associated retrieval mechanisms, we consider a case with five file types and their proportion of traffic load as described in Table 1. We assume that each disk provides 20 channels. In all simulations in this section that were performed to obtain blocking probability, the radius of the 95% confidence interval was kept below 20%.

In our first experiment, we aim to study the fairness of CLB. In Table 2, we present the blocking probabilities of the different types of requests, obtained by simulation for the CLB scheme with LBF retrieval mechanism, for various traffic loads. Each simulation was run with the appropriate number of disks to maintain the blocking probability below 1% for each file type. The number of disks in each simulation is recorded in the table. We see that under CLB, type 1 files will suffer the worst blocking probability, type 2 the second worst and so on. This can be explained as follows. Without loss of generality consider a request for a file type 1 located on a given disk. This request will have the same probability to be rejected from that disk as any other (say type 2) request arriving on the same disk. However, if blocked, the type 2 request has another opportunity to obtain a copy of that file on another disk while the type 1 request does not.

In Figures 1-3, we set the constraint for the worst case blocking probability at 1% for all schemes. The results for blocking probability for CLB and NCLB were obtained by simulation. In all three figures, we see that CLB and NCLB perform better than NLB and the performance of CLB and NCLB are close.

In Figure 1, we present the number of disks required to meet the blocking probability constraint. We observe step functions for all cases. This is due to the fact that in all cases as traffic increases, whenever the required number of disks is added to meet the blocking probability constraint, the traffic can further increase until congestion recurs. We observe that for NLB, the curve shape in Figure 1 does not exhibit constant size steps but the step size is variable. As the traffic load slowly increases, disks related to different file types become congested, so if a disk that contains files that require one copy becomes congested, then we have a step of one, while if a disk that contains files that require two copies becomes congested, then we have a step of two, and so on.

In Figure 2, we present the disk wastage [%] of NLB relative to CLB-LBF. We observe significant oscillation for low traffic. Notice that NLB has to start with $1+2+3+4+5=15$ disks even for very small traffic load while in the case of CLB, we can begin with 5 disks. For high traffic loads, we see constant wastage percentage of NLB relative to CLB at around 12-15%.

Figure 3 is a continuation of Figure 1 extending the traffic load from 500 to 4000 Erlangs. The near linear behavior of all schemes in high traffic load is much clearer in this figure.

In Figure 4, we plot the number of disks required to meet 1% blocking probability as a function of the traffic load for all three CLB file retrieval mechanisms as well as for the NLB scheme in a homogeneous case where only files of type 2 are considered. We see, as expected, that CLB-LBF performs the best. We also observe that NLB performs better than CLB-SRT. It seems that the wastage due to the single trial is significant.

5. CONCLUSION

We have introduced three file assignment schemes in a VoD system. The NLB scheme is simple to implement and ana-

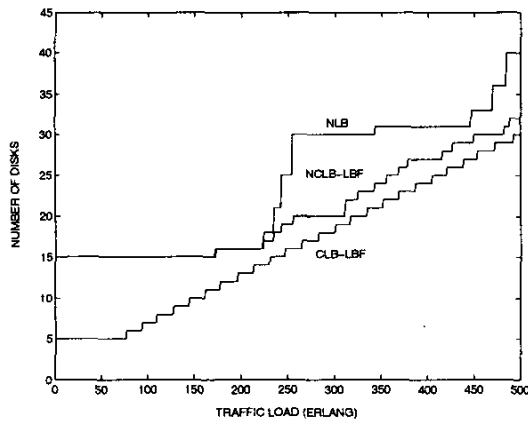


Fig. 1. Required number of disks of NLB, NCLB-LBF and CLB-LBF versus the traffic load, for the 5 file type case.

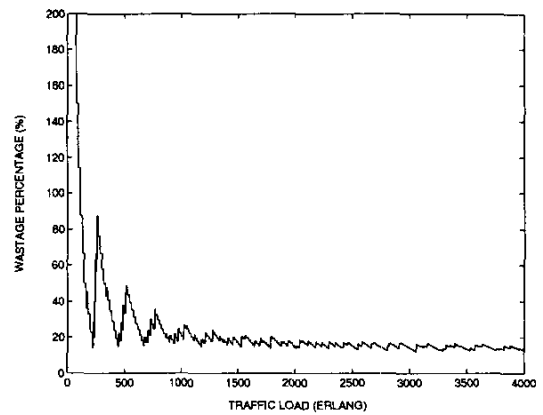


Fig. 2. Disk wastage percentage of NLB relative to CLB-LBF versus the traffic load, for the 5 file type case.

lyze, but it is not as efficient as the CLB and NCLB schemes. NCLB performs as well as CLB for high traffic loads and, can provide fairness and blocking probability control for each file type as NLB does. From numerical results, for the case where the traffic levels are high, the NLB scheme required 12-15% more disks than the CLB scheme.

6. REFERENCES

- [1] D. Deloddere, W. Verbiest, and H. Verhille, "Interactive video on demand," *IEEE Communications Magazine*, vol. 32, no. 5, pp. 82-88, May 1994.
- [2] T. D. C. Little and D. Venkatesh, "Prospects for interactive video-on-demand," *IEEE Multimedia*, vol. 1, pp. 14-24, 1994.
- [3] Lawrence W. Dowdy and Derrell V. Foster, "Comparative models of the file assignment problem," *ACM Computing Surveys*, vol. 14, no. 2, pp. 287-313, 1982.
- [4] D. V. Foster, L. W. Dowdy, and J. E. Ames, "File assignment in a computer networks," *Computer Networks*, vol. 5, pp. 341-349, September 1981.
- [5] K. S. Tang, K. T. Ko, S. Chan, and E. Wong, "Optimal file placement in VOD system using genetic algorithm," *IEEE Transactions on Industrial Electronics*, vol. 48, no. 5, pp. 891-897, October 2001.
- [6] H. Akimaru and K. Kawashima, *Telettraffic: Theory and Applications*, Springer-Verlag, London, 2nd edition, 1999.

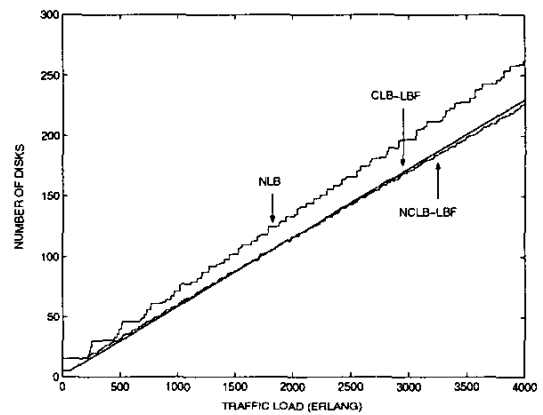


Fig. 3. Required number of disks of NLB, NCLB-LBF and CLB-LBF versus the traffic load, for the 5 file type case.

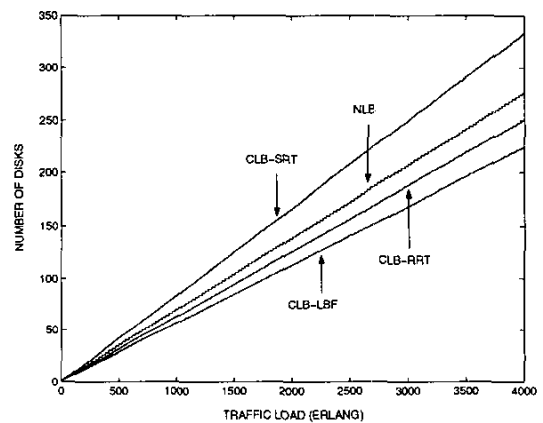


Fig. 4. Required number of disks of NLB, CLB-LBF, CLB-RRT, and CLB-SRT versus the traffic load, for the homogeneous file type case.