# Performance Study on Video Placement and Load Balancing of Distributed Video-On-Demand Systems

Jonathan C. Lu and Chung-Hao Chen
Department of Computer Science and Information Engineering
Fu-Jen Catholic University
Taipei, Taiwan, R.O.C.
e-mail: {jonlu, armain86}@csie.fju.edu.tw

**Abstract**

Distributed VOD server is a solution to the problem of the capacity limit posed by single server system. There needs to be an algorithm for the head-end to use to evaluate the current status of traffic loading in the whole VOD system, so that it can decide whether a new user request should be served locally, forwarded to a remote server, or simply be rejected. The computation is further complicated by the fact that server capacities and communication costs between different servers may be variable.

In this paper, we first define a mathematical model used to analyze the performance of our VOD systems. We then present an algorithm to compute the number of copies each film should be duplicated, as well as where to place them in order to gain maximum revenue. Numerical results show that our algorithm offers improvement over Chen's algorithm [7]. We also present a method to evaluate whether a new user request should be served locally, or forwarded to a remote server, or simply be rejected. It can be seen under the prerequisite of maximal revenue that while a local video server is too busy to serve a local request, it is a good idea to send it to a remote video server which has extra capacity to process it.

**Keyword:**
**Video-On-Demand, Film Duplication and Placement, Load Balancing, Performance Analysis.**

## 1. Introduction

These years there have been a great number of research activities on multimedia applications because of the rapid advances in broadband networking, mass storage, and video/audio encoding such as MPEG technologies. Among them, the research results on VOD have especially received a lot of attention in both the industries and academia [1][2][3][8][13][14][15]. The main bottleneck, however, resides in the development of large video storage systems.

Some researches [4][5][6][7][12][17] have suggested putting in a video server multiple disks, which are divided into several clusters. But most of those solutions is to stripe every video file over all servers. The advantage of this approach is that data retrievals can be interleaved as to achieve load balancing between clusters. However, the system is not reliable, since the failure of one server will result in data loss.

Another idea [6] [19][20] is to place buffer space between the video server and the client's set-top box. Sufficient amount of video data can be first downloaded to the buffer via high-speed communication networks before being further passed down to the client side and played back. This scheme not only smoothes the playback by reducing the impact of delay jitter caused by network traffic fluctuations, but the video server now does not need to be tied up to the clients as long as before, thus has more effective capacity and can serve more customers. Nonetheless, it may not be cost-effective to construct many large-sized buffers, compared with the cost of disk storage.

Because the number of users a single video server can serve is limited, some research [9][10][11][18][21] had started to study distributed VOD servers. The cost of this approach is extra disk space. Though video programs has to be duplicated at suitable intermediate levels, depending on the trade-off among the bandwidth, storage cost, and load balancing. At the same, each switch must decide whether a new request should be served by the local server, or by a remote server when the local server does not have the requested program.

Therefore, we produce some methods to resolve above problem in this paper. The rest of the paper is organized as follows: In section 2, we define the adopted network model and cost function. In section 3 the film placement and duplication algorithms for our network model are described. In section 4 we present a method to decide whether a new user request should be served locally, remotely, or simply be rejected. Finally, section 5 gives the conclusion.

## 2. The Network Model and The Cost Function

In this section we define our network architecture as well as the mathematical models and assumptions that we will apply to evaluate the performance of the VOD system.

### 2.1 The Network Model

We assume that there are $S$ communities in the systems, which are denoted by $C_i = \{ 1 \leqq i \leqq S \}$. Each community consists of video servers, users and switching equipment connecting the community to the others as shown in Figure 2.1. Let $U(i)$ represent the number of users in $C_i$. Each video server has certain capacity for storing video films, and users in a community may generate requests to ask a particular film to be downloaded and displayed.

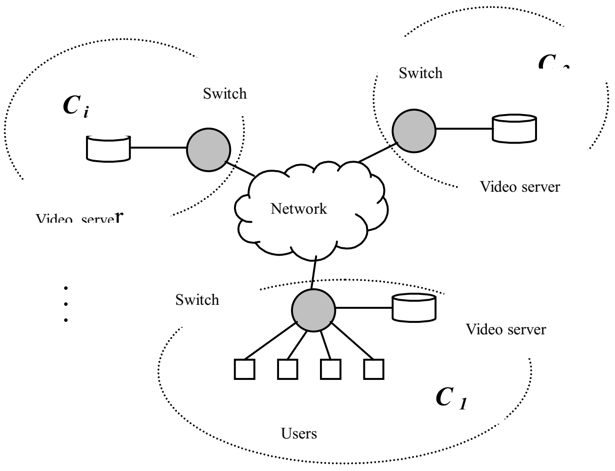


Figure 2.1 VOD network model

### 2.2 The Cost function

Let $cost(i,j)$ represent the communication cost between $C_i$ and $C_j$. We assume that no communication cost will be incurred if a request is served by the video server (called its local server) in the same community. For community $C_i$, let $CAP(i)$ represent the capacity (in number of films) of video storage, and $fp(i,k)$ denote the number of copies of the kth film placed at the video server in $C_i$. Thus, we can say:

$$\sum_{k=1}^{F} fp(i,k) = CAP(i) \qquad (2.1)$$

Where $F$ represents the number of different films available in the whole VOD system.

We assume that the viewing probability of a film could be different in different communities depending on its local popularity. The sum of the viewing probabilities of all the films available in the system is equal to 1 in each community. Let *vp(i,k)* denote the viewing probability of the kth film at $C_i$. We have:

$$\sum_{k=1}^{F} vp(i,k) = 1 \qquad (2.2)$$

Let $\lambda$ represent the rate an idle user generates a viewing request, and *req(i,k)* denote the aggregate rate of viewing request for the kth film in $C_i$, where

$req(i,k) = U(i) \times \lambda \times vp(i,k)$.

Since the communication bandwidth between the user and the video server is usually much faster than the client's playback rate, each copy of the film at the server disk is able to serve multiple client requests simultaneously. Given this observation, we will model the behavior of each film and its viewing requests as an M/M/m/R queue(see appendix) [16], where the request arrival rate is equal to *req(i,k)*, and the service rate $\mu$ is assumed to be known. We further assume that a copy of film stored on disk at $C_i$ can handle $b_i$ viewing requests simultaneously. The value $b_i$ may be different from server to server depending on CPU and the speed of hard disk, etc. The system space, R, which equals $fp(i,k)\ x b_i$, represents the number of viewing requests that can be served simultaneously by the *fp(i,k)* copies of the kth films placed at the video server. Also *m* is set to be equal to R, which means that there is no waiting room for any accepted request, because typically a user would expect a film to start playing very soon after his request is accepted. The performance of a video server is mainly limited by the number of films itself has. The CPU and network connection is assumed to be very fast that they will never become the bottleneck. We next define our objective functions for two cases:

**Case ( Ⅰ ):**

A new request generated by users in $C_i$ will either be served or rejected by its local video server, i.e., no request is forwarded. We have:

$$TR = \sum_{i=1}^{S}\sum_{k=1}^{F}[1 - P_B(i,k)] \times req(i,k) \times R_k \qquad (2.3)$$

where *TR* is the total revenue collected, $P_B(i,k)$ means the blocking probability for an arbitrary viewing request for the kth film in $C_i$, and $R_k$ denotes the fee a user pays for watching the kth film .

**Case ( Ⅱ ):**

A new request from users in $C_i$ could be transferred to and served by video servers in a different community $C_j$ $(i \neq j)$. We have:

$$\sum_{i=1}^{S}\sum_{k=1}^{F}[1 - P_B^*(i,k)] \times req^*(i,k) \times R_k - total\_communicaton\_cost \qquad (2.4)$$

where: $req^*(i,k)$ is the composite viewing rate for the kth film in $C_i$ . This value may be different from the original *req(i,k)*, because some fraction of *req(i,k)* may be forwarded to a different video server, while requests generated at other communities may also be forwarded to $C_i$. $P_B^*(i,k)$ means the blocking probability for $req^*(i,k)$. When a request generated at $C_i$ is redirected and served by video server at $C_j$, the communication cost incurred is equal to *cost(i,j)* as defined previously. The total_communication_cost represents the aggregate communication cost incurred in the overall system.

## 3. The Film Duplication And Placement Algorithms

In this section we will present both a duplication algorithm for calculating how many copies the ith film should be duplicated, and a placement algorithm for determining where to place those copies in our VOD system. Numerical results are shown to evaluate the performance of the algorithm.

## 3.1 Film Duplication Algorithm

Our duplication calculation follows the basic idea that the higher a film viewing rate, the more the number of its duplicates. The number of duplicates of the kth film, denoted by *fc(k)*, can be calculated as follows:

```
Initially $fc(k) = 1,\ 1 \leq k \leq F$;
While ( total_capacity $\neq 0$ )
  Choose the film k which provides the maximum value for
  $\frac{\sum_{i=1}^{S} U(i) \times vp(i,k)}{fc(k)}$;
  Set $fc(k) = fc(k)+1$ and total_capacity = total_capacity -1;
```

Where *total_capacity* is the total storage capacity of our VOD system.

For approach 1, the idea is that the number of copies of a film placed at $C_i$ should be proportional to its viewing rate at $C_i$. Our method is defined as following:

```
List fc(k)'s in a nonincreasing order and put them in a queue;
While ( queue is not empty )
    Remove the first item from queue and suppose that it is fc(k);
List fc(k)'s in a nonincreasing order and put them in a queue;
While ( queue is not empty )
    Remove the first item from queue and suppose that it is fc(k);
List fc(k)'s in a nonincreasing order and put them in a queue;
While ( queue is not empty )
    Remove the first item from queue and suppose that it is fc(k);
Set $fp(i,k) = 0,\ 1 \leq i \leq S$;
  While ( $fc(k) \neq 0$ )
   Let community i be the one where the placement of a copy of film k
   produces the maximum increase in TR based on equation (2.3);
Set $fp(i,k) = fp(i,k) + 1$, $CAP(i) = CAP(i) - 1$ and $fc(k) = fc(k) - 1$;
If (CAP(i) equals 0) remove community i from consideration;
```

where *fp(i,k)* presents the number of copies of the kth film to be placed at $C_i$. After the calculation, we have:

$$\sum_{i=1}^{S} fp(i,k) = fc(k) \qquad (3.1)$$

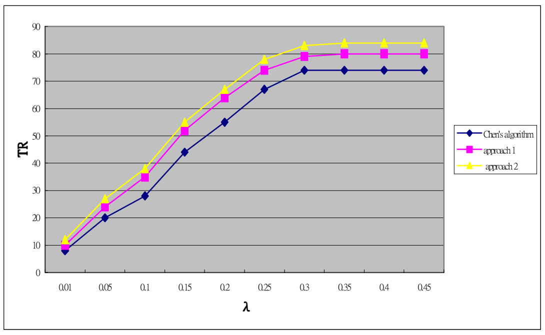
and

$$\sum_{i=1}^{S} \sum_{k=1}^{F} fp(i,k) = total\_capacity \qquad (3.2)$$

On the other hand, for approach 2, the basic principle is to place a new film copy in the community where maximum incremental revenue can be generated.

## 3.3 Numerical Results

Below we present some performance results of our algorithm compared to Chen and Lin's. In our numerical experiments using M/M/m/R model, we set $\mu$=0.3 and assumes that a copy of a film can serve 5 requests simultaneously (i.e. $K = 5 \times fp(i,k)$). The viewing fee $R_k$ is randomly drawn from the interval between 2 and 3. $\lambda$ denotes the average number of requests each user generates in a unit of time. *TR* stands for the total revenue calculated by Equation (2.3). Since we assume that the viewing probability of a film is different in different communities, the viewing probability *vp(i,k)* is drawn from a geometric distribution, where $vp(i,k) = (1 - P_i) \times P_i^k$



| Community | *U(i)* | *CAP(i)* |
|---|---|---|
| $C_1$ | 80 | 35 |
| $C_2$ | 130 | 45 |
| $C_3$ | 180 | 55 |

Figure 3.1. Total revenue versus load for the case of three communities

From the numerical results above, it can be seen that both of our approaches offer improvement over Chen's algorithm, with approach 2 producing slightly higher revenue than approach 1. Because the conditions on other numbers of $S$ are similar to the numerical results above, we only show the results of $S = 3$.

## 4. Request Redirection Among Different Servers

In this section we are to study the case where switches can communicate with each other. Once a switch has decided that a new request should be forwarded to a remote video server, we next need to know which video server the request should be forwarded to. The communication cost between the two servers has to be subtracted from the total revenue, as shown in Equation (2.4). Also note that when a remote server agrees to accept a request sent by another server, the blocking probability at that server will become larger, which in turn results in a reduced revenue.

Our objective is to lower communication cost and to obtain maximum revenue. In this section, we will divide our problem into two cases. Case one is that a server will not redirect any of its local requests unless it does not have the requested film at its local storage.

### 4.1 Request Redirection When No Film On Local Server

In this section we discuss the case 1 where a local request will be forwarded only if there is no any copy of the requested film at the local server.

Let *ec(i,k)* represent the extra capacity that server in $C_i$ can use to process requests for the kth film forwarded by other servers. Its calculation is as follows:

```
If ( fp(i,k) =0 ) then ec(i,k) = -req(i,k)
Else
    ec(i,k) = fp(i,k)×b_i×(μ-req(i,k));
    if ( ec(i,k) < 0 ) then ec(i,k) = 0;
```

where *ec(i,k) = 0* means that video server in $C_i$ cannot accept any foreign request from the other video servers. If *ec(i,k) < 0*, it means that certain amount of the local requests for the kth film in $C_i$ should be forwarded to other servers. If *ec(i,k) > 0,* it means that the video server in $C_i$ has extra capacity to accept remote requests for the kth film. Our procedure is as follows:

```
Consider those pairs where ec(i,k)>0 and ec(j,k)<0,
where $i \neq j$, $1 \leq k \leq F$;
While (pairs still exist)
Begin {while}
    Find the pair, (ec(x,k) , ec(y,k)), that has the maximal
    $\frac{R_k}{cost(x,y)}$ value;
    If ($\frac{R_k}{cost(x,y)} > 1$)
    Begin {if}
        Redirect the amount of W = min{|ec(y,k)|,
        ec(x,k), bandwidth(x,y)} from y to x;
        Add $W \times cost(x,y)$ to the
        total_communication_cost;
        ec(y,k)=ec(y,k)+W;
        ec(x,k)=ec(x,k)-w;
        bandwidth(x,y)=bandwidth(x,y)-w;
        If (ec(x,k)=0) remove any pair including ec(x,k) from
        consideration;
        If (ec(y,k)=0) remove any pair including ec(y,k) from
        consideration
        If (bandwidth(x,y)=0) remove any pair(ec(x,m) , ec(y,m)), 1
        ≤m≤F, between x and y among consideration;
    End {if}
        If (ec(y,k)=0) remove any pair including
ec(y,k) from
        consideration
        If (bandwidth(x,y)=0) remove any
            pair(ec(x,m) , ec(y,m)), 1 ≦ m ≦ F, between x
            and y among consideration;
Else stop;
End {while}
```

Where *bandwidth(i,j)* to represent the number of requests that can be transmitted between $C_i$ and $C_j$ per unit time.

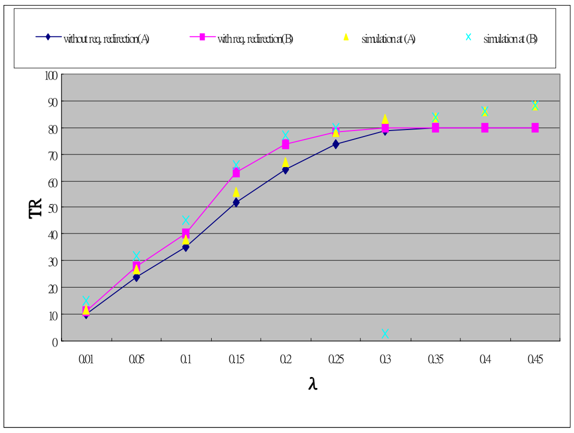
## 4.2 Request Redirection Among Servers With Heavy Load On Local Server

In this section we will discuss the case 2 where a server has the requested film may also forward a local request because it is too busy to handle all the requests. The whole procedure is roughly the same as the case one in the previous section, except that the initialization of *ec(i,k)* is slightly different:

```
ec(i,k) = fp(i,k)×b_i×μ-req(i,k);
```

## 4.3 Numerical Results

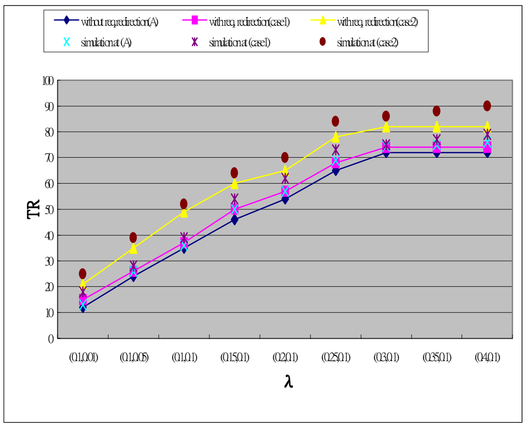
*TR* stands for the total revenue given in equation (2.4). The *bandwidth(i,j)* is randomly drawn from between 60 to 90. For film duplication and placement algorithm (approach 1) and other parameters described in section 3 is used. Figures 4.1 displays the result at case on e. On this paper we ignore the cost that video servers frequently exchange current traffic load information.



| Community | *U(i)* | *CAP(i)* |
|---|---|---|
| $C_1$ | 42 | 7 |
| $C_2$ | 56 | 12 |
| *F = 10 films, S = 2,* $\mu = 0.3$, $P_1=0.2$, $P_2=0.15$ | | |

(2 communities)

Next we present the numerical results for case 2. In our experiment we assume that half of the video servers are heavily loaded while the other half are lightly-loaded, and viewing requests may be forwarded from heavily-loaded servers to lightly-loaded ones. The label *(X,Y)* on the abscissa means that *X* and *Y* represent *λ's* for each of the heavily-loaded servers and lightly-loaded servers, respectively.



| Community | *U(i)* | *CAP(i)* | $b_i$ |
|---|---|---|---|
| $C_1$ (lightly-loaded) | 80 | 35 | 4 |
| $C_2$ (heavily-loaded) | 130 | 45 | 6 |
| $C_3$ (lightly-loaded) | 180 | 55 | 4 |
| $F = 45$ *films*, $S = 3$, $\mu = 0.3$, $P_1=0.15$, $P_2=0.1$, $P_3=0.05$ | | | |

Figure 4.2 Total revenue versus load for case 2 (3 communities)

The results (case1 and 2) show that while a local video server cannot serve its local requests, it is a good idea to send it to a remote video server that has extra capacity. Because the conditions on other numbers of *S* are similar to the numerical results above, we only show the results of *S* = 2 and 3.

## 5 Conclusion

Because the number of a user a single video server can serve is limited, we had started to study distributed VOD servers. Therefore, we defined our network architecture and the mathematical models. At a time, we presented both a duplication and placement algorithms, which offer improvement over Chen' algorithm. We also provided an algorithm that can decide where a new user request should be served and proved that it is a good idea to send it to a remote video server that has extra capacity to accept them.

## References

[1] Daniel Deloddere,Willem Verbrbiest, and Henri Verhille, "Interactive Video On Demand", *IEEE Communications Magazine*, pp. 82-88, May 1994.

[2] M.Kumar, "Video-server designs for supporting very large numbers of concurrent users", *IBM J. RES.DEVELOP.*, VOL.42, NO.2, pp. 219-231, March 1998.

[3] Thomas D. C. Little and Dinesh Venkatesh, "Prospects for Interactive Video-on-Demand",*IEEE Multimedia*, pp. 14-24, Fall 1994.

[4] B. Ozden, R. Rastogi, and A. Silberschatz, "Fault-tolerant architectures for continuous media servers", *in Proceedings of SIGMOD Conference*, 1996, pp. 79–90.

[5] Y. Wang et al., "Video file allocation over disk arrays for video-on-demand", *Proceedings of the third IEEE International Conference on Multimedia Computing and Systems*, June 1996.

[6] J. Leon Zhao, Doron Rotem, and Su-Shing Chen , "Data Management for Multiuser Access to Digital Video Libraries" *Journal of Parallel and Distributed Computing 56, 208_234,* 1999

[7] Chen Yu-Tang and Lin Ming-Yi, "Video Placement and Local Balancing for VOD Services", *National Conference for Computing, pp.* D105-D110, Taipei, 1997.

[8] W. D. Sincoskie, "System architecture for a large scale video on demand service",*Computer Networks and ISDN Systems*, pp. 155-162, (22) 1991.

[9] Frank Schaffa and Jean-Paul Nussbaumer, "On Bandwidth and Storage Tradeoffs in Multimedia Distribution Networks", *IEEE Infocom 95*, pp. 1020-1026, 1995.

[10] Chatschik C. Bisdikian and Baiju V. Patel, "Issues on Movie Allocation in Distributed Video-on-Demand System", *IEEE ICC '95*, pp. 250-255,1995.

[11] Giacinto Dammicco and Ugo Mocci, "Optimal Server Location in VOD Networks", *IEEE Globecom*, pp. 197-201,1997.

[12] Chen Shun-Ang, Liu Xiang-Jun, Huang Yue-Min, and Ma jin-Gou, "Cluster-Pairing Data Placement : A Novel Scheme to Efficiently Utilize Data Transfer Rate of Zoned-Disk for VOD Servers", *Second workshop on Real time and Media Systems, pp.* 67-73, Taipei, 1996.

[13] Ren-Hung Hwang and Jang-Jiin Wu, "Scheduling Policies for an VOD System over CATV Network", *IEEE Globecom*, pp. 438-442, 1997.

[14] Ying-Dar Lin, and Chia-jen Wu, and Wei-Ming Yin, "Pipelined Cyclic Upstream

[14] Ying-Dar Lin, and Chia-jen Wu, and Wei-Ming Yin, “Pipelined Cyclic Upstream Protocol Over Hybrid Fiber Coax” *IEEE Network January/February*, pp. 24-34, 1997.

[15] A. Mu¨ fit Ferman and A. Murat Tekalp, “Efficient Filtering and Clustering Methods for Temporal Video Segmentation and Visual Summarization” *journal of visual communication and image representation Vol. 9, No. 4, December, pp*. pp. 336–351, 1998

[16] Donald Gross and Carl M. Harris, *Fundamentals of Queueing Theory*, Jon Wiley & Sons, 1974.

[17] Emmanuel L. Abram-Profeta and Kang G. Shin Emmanuel L. Abram-Profeta and Kang G. Shin, “A Practical Approach to Resource Allocation in Video-on-Demand Servers“, *journal of visual communication and image representation Vol. 9, No. 4, December, pp*. 314–335, 1998

[18] V. O. K. Li, W. Liao, X. Qiu, and E. W. M. Wong, “Performance model of interactive video-on-demand systems”, *IEEE J. Selected* Areas Commun. 14, 6, 1996, 1099–1109.

[19] A. Dan et al., “Buffering and caching in large-scale video servers”, *Compcon__Technologies for the Information Superhighway, Digest of Papers, Jan. 1995*, pp. 217_224.

[20] D. J. Makaroff and R. T. Ng, “Buffer sharing schemes for continuous-media systems”, *Inform. Systems 20(6) 1999,* 445_464.

[21] A. Dan, M. Kienzle, and D. Sitaram, “A dynamic policy of segment replication for load balancing in video-on-demand servers”, *ACM/Springer Multimedia System* 3, 1995, 93–103.