December 2000

Video Management in Commercial Distributed Video on Demand (VoD) Systems

K. Tsang  
Hong Kong University of Science and Technology

Sai Kwok  
Hong Kong University of Science and Technology

Follow this and additional works at: http://aisel.aisnet.org/pacis2000

Recommended Citation
http://aisel.aisnet.org/pacis2000/17

This material is brought to you by the Pacific Asia Conference on Information Systems (PACIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in PACIS 2000 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
Video Management in Commercial Distributed Video on Demand (VoD) Systems

K. F. Tsang, S. H. Kwok
Department of Information and System Managements
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
{kaifung.jkwok}@ust.hk

Abstract
In this paper, we present a predictive video allocation algorithm for video management in commercial distributed Video on Demand (VoD) system. The predictive video allocation algorithm contains two parts: replication and allocation. With the use of the proposed video allocation algorithm, the load balancing of the Video on Demand (VoD) system can be greatly improved. To better manage the video in a VoD system, a "prediction" approach is proposed. The "prediction" approach is based on the expected viewing pattern, or the popularity of the movies formulated by Zipf's law. The predictive video allocation algorithm is designed to operate in a "dirty" video server environment, where previously released movies are already kept there. Since the number of video server is fixed in a real world system, the proposed algorithm will not minimize the number of server used, but maximize the servers' utilization. This is a common scenario in commercial VoD systems but not addressed and considered in many previous studies. Experimental results show that the proposed algorithm achieved a very good load balancing in "dirty" system, while resource-demanding reallocation of previously released movies is not needed.

Keywords: Video Management, Resource Scheduling, Video on Demand.

1. Introduction

Advancements in video compression and high-speed communication network have made Video on Demand (VoD) services possible in recent years. Many telecommunication companies provide VoD services and other broadband services to residential users. Transmitting video data to residential end-user requires high network bandwidth and large amount of storage spaces at the server side. In order for concurrent access by the users to a variety of movie titles, efficient use of resources, including storage capacity and network bandwidth becomes a crucial factor towards the success of VoD services. In-balanced placement of movies will lead to a waste of resources and an increase of the request failure rate.

Video servers can be characterized into two categories: centralized and distributed [6]. Centralized, or Single-server, VoD system has two limitations: 1. Capacity, and 2. Fault tolerance [2]. Centralized approach is a single point network connection to the server and requires taking care of all network traffic; as a result, it needs much larger bandwidth than the distributed approach. On the other hand, distributed video servers, or known as parallel video servers, work co-operatively to store the movies and handle requests. There are some studies on single-server system to achieve better resource utilization. Movie stripping over disk arrays is a common technique for centralized video server, as this technique can increase the throughput and results in better load balancing. Due to the limitations of centralized
approach, large-scale commercial VoD systems favor distributed video servers. The distributed video server architecture provides higher scalability and robustness.

Quality of service in VoD heavily depends on the load and storage balancing. For example, in distributed system, each server should bear the same amount of loads in order to minimize the rate of request failure [4]. The rate of request failure may refer to the probability that an end user accesses a particular movie unsuccessfully when the associated server is overloaded. Thus, resource and load optimization is one of the important research problems in the field.

A solution to this problem is server-level stripping [2], which applies the concept of disk-level stripping in disk array system [9], like RAID. Server-level stripping breaks a video into pieces of data files and stores them in multiple servers. This enhances the VoD service by offering more concurrent multiple accesses and downloads of popular movies. An advantage of using server-level stripping in distributed video server system is that the space requirements remain unchanged while a movie can be shared by many end-users simultaneously. As the set-top-box (STB) at the client side requires a continuous stream of video from the VoD system, pieces of video are reordered and streamed to the STB. However, a proxy is needed to manage video pieces. There are three commonly used strategies to integrate the proxy to the system [2]: 1. proxy at client, 2. independent proxy and 3. proxy at server. Thus, the shortcoming of the server-level stripping approach is that the cost of the system increases due to the additional cost of proxy server. Moreover, fault tolerance in this system is not high. The VoD service may be interrupted when a single server fails. To resolve this, extra copies of movies become necessary but this leads to a highly complex data allocation scheduling. Thus, server-level stripping is not a common commercial video server solution.

In addition, there are not many commercial video servers supporting server-level stripping. These video servers are just working like standalone servers. For the same reason, hybrid approach, which combines stripping and replication, in literature [10], is not practical, while it is expected to give optimal performance.

Multicast is regarded as a superior methodology for increasing network efficiency. It batches a number of clients that access to the same video stream to increase the number of simultaneous streams that the system can support. Although multicast enables an efficient use of network resource, earlier multicast techniques do not support “forward” and “rewind” VCR operations until Liao and Li’s proposal [3] of “split and merge” protocol to address this issue. However, the advancement of multicast technique does not relax the storage and load-balancing problem at the server side.

Replication is an alternative solution for resource allocation problem. It replicates videos according to the popularity of the videos, and/or the concurrent access profile. Multiple copies of videos are placed at different video servers using packing algorithm, which optimizes the required disk spaces and the number of servers in use. As proved by Wang et al. [10], when the minimum number of disk arrays (or servers) needed to satisfy the concurrent access requirements of an movie popularity profile is larger than the minimum number of server needed to store all the movies (a normal case in real world systems), replication can keep the storage minimum, and fulfill the concurrent access requirements of the movie popularity profile.
Serpanos et al. [6] proposed a MMPacking replication technique for balancing the probability of the servers of being accessed, i.e. each server will have a probability of \(1/N\) of being accessed when there are \(N\) servers. MMPacking sorts the movies in ascending order according to their popularity, and places them in a round robin fashion. If the cumulative probability of a server is larger than \(1/N\), the movie will be replicated and placed to the next server. The main problem of this scheme is that it assumes the file sizes of the movies are the same, so it does not handle storage capacity balancing. This assumption does not hold in real application and therefore MMPacking is not suitable for many commercial systems, since disk spaces or storage capacity is an expensive resource in commercial systems.

Most of the previous studies and systems are referred to as “clean” systems; i.e., all videos are required to be reallocated when the VoD system refreshes with new movie titles. However, some previously released movie titles may actually be carried forward to the current and/or even future release. As these movies are already in the video system and they probably consume up to 40-60\% of the total storage capacity, reallocating them in order for newly released movies is impractical and inefficient in most cases. Video reallocation requires a great deal of system resources and this affects the quality of the existing services in terms of the user respond time, bandwidth, fault deletion of movies, and so on.

With the above considerations and limitations, this paper proposes a predicable video allocation algorithm for video management in distributed commercial VoD system. The word “predicable” implies that the placement of the movies is achieved prior to newly released movies launched. The proposed method is designed to operate in “dirty” systems, where movie popularity profile is mixed with previously and newly released movie titles. Replication approach is used in the proposed method to ensure the movies have enough replicas to fulfill the popularity profile, while load balancing is a concern. Since the number of video server is fixed in a real world system, we also propose to maximize the utilization of system resources, including disk space and bandwidth, rather than minimize the number of servers used suggested by other methods in the literature.

The proposed Predicable algorithm is designed for applying in the pre-deployment phase. Adaptive algorithms [5, 7], which can reallocate movies according to the changing viewing pattern on the fly, is usually used in deployment phase and we will not go into this area in this paper.

Section 2 provides the definitions of the load of movie and server and all other notations. Section 3 illustrates the proposed predicable video allocation algorithm in details. Section 4 summaries the experimental results. We conclude the paper and discuss the future directions in Section 5.

2. Definitions of the load of movie and video server

To achieve load balancing in the allocation of movies, we need to characterize the load of a single movie. Then we can allocate the movies to different servers according to their loads, so that each server will have the same load assigned finally.

General speaking, the load of a movie can be characterized as its consumption of the server resources, for example, storage, memory, buffer, network bandwidth, etc. Besides storage resource in a server, we can simplify all other resources as the concurrent streams capacity of
the server. With the constraint of memory, network bandwidth, and buffering, each server has a maximum number of concurrent streams that it can support; for example, the maximum number of concurrent streams of the server in [1] was 72. After a movie is placed onto the server, it will consume part of the server’s concurrent streams capacity and we can see it as the “concurrent access requirement”, \( c_{ai} \), of a movie.

Concurrent access requirements can then be defined as the expected number of requests of a movie in an instance of time. In our proposed scheme, we adopted Zipf’s law to calculate the concurrent access requirements based on the popularity or the “expected popularity” of the movies. Zipf’s law is a well-known distribution for modeling the rental pattern in movie rental services. The probability \( P_n \) of choosing the nth most popular movie from \( M \) movies is defined in Zipf’s Law as:

\[
P_n = \frac{K}{n}
\]

where \( K = \frac{1}{1 + 1/2 + 1/3 + \cdots + 1/M} \)

As the calculation of \( P_n \) is based on the popularity of movies, system administrators will have to decide each movie’s popularity. Popularity of carry-forward movies can be determined with their hit-rate in previous months. Popularity of newly released movie is an uncertainty, since they are never served on the system. However, it can be approximated accurately by referencing to box office records, movie reviews, etc.

From (1), we can then define \( c_{ai} \) as \( P_n \times S \), that the whole system can support simultaneously.

\[
C_{ai} = P_n \times S
\]

After the concurrent access requirement of movies is defined, the load of server \( k \), \( C_k \), can be characterized as a linear combination of concurrent access requirements and the storage requirements of the movies stored on the server, as proposed by Wang et al. [10]. The following formulations are derived from Wang et al. [10].

\( C_k \) is defined as the linear combination of the concurrent access load, \( L_{access} \), and storage load, \( L_{storage} \), which are normalized by the maximum stream that a server can support, \( d \), and the storage capacity of the server, \( z \), respectively, as follow:

\[
C_k = R_{access/storage} \times L_{access_k} + \alpha \times R_{storage/access} \times L_{storage_k}
\]

(3)

After a video with size\( i \) is placed on a server, the concurrent access load of the server will increase by \( c_{ai} / d \) and the storage load will increase by \( size_i / z \). i.e.:

\[
L_{access_k} = L_{access_k} + \frac{c_{ai}}{d}
\]

(4)

\[
L_{storage_k} = L_{storage_k} + \frac{size_i}{z}
\]

(5)

Since \( C_k \) will be increased by \( R_{access/storage} \times \frac{c_{ai}}{d} + \alpha \times R_{storage/access} \times \frac{size_i}{z} \) after a movie is placed on it, we can then define the load of movie \( i \), \( L_i \), as:

\[
L_i = R_{access/storage} \times \frac{c_{ai}}{d} + \alpha \times R_{storage/access} \times \frac{size_i}{z}
\]

(6)
\( R_{access/storage} \) and \( R_{storage/access} \) are the ratios of the minimum number of servers needed to satisfy the concurrent access requirements of an movie popularity profile \((N_{access})\) to the minimum number of servers needed to store all the movies \((N_{storage})\) and verse visa. It is noticed that \( L_{access} \) and \( L_{storage} \) should not be greater than 1 for any server. \( \alpha \) in the equation (3) is a scaling factor which can determine how the access load or the storage load are biased.

3. Predictive Video Allocation

The remaining problem is how to place the movie to the N number of servers in the system, which should be optimized that the system can provide the maximum number of streams when the requests fulfill the predicted pattern, or popularity profile. Thus we propose a predictive video allocation in this paper and the predictive video allocation consists of two parts: replication and allocation of movies.

3.1 Replication

Replication is to make copies of popular movies, as long as the system has such a storage capacity for them. It can provide more concurrent streams of movies at the cost of storage space. In our situation, \( N_{access} > N_{storage} \), replication is desirable, since it only increases \( N_{storage} \), and will not increase the upper bound of the number of servers used, which is \( N_{access} \) in this case [10]. Replication enhances the accessibility of popular movie titles by providing more concurrent streams. In order to reduce the request failure rate during the peak hours, the replication is optimized for the concurrent access requirements of the movies. To do this, we first sort the \( M \) movies, which include both previously released movies and newly released movies, in descending order, based on their \( ca_i \). The first entry in the list is then duplicated, and each child copy shares the concurrent access requirements of its parent equally. The total number of movies \( M \) will also be increased by 1 after each replication. After each replication, the list is re-sorted. The replication process will be recursively created until predefined criteria meet. For example, when the total size of the movies in the list nearly reaches the system’s total storage capacity, there will be no more new replication process. This condition can be expressed as follows.

\[
\sum_{i=1}^{M} size_i >= \sum_{k=1}^{N} d_k - E
\]  

(7)

where \( M \) is the total number of movies; \( size_i \) is the size of movie \( i \); \( N \) is the total number of servers in the system; \( d_k \) is the storage capacity of server \( k \); \( E \) is a constant which ensures there will have enough space left for allocating all the movies in the list.

The replication process can be summarized as follow:

Step 1. Sort movies in descending order, based on their \( ca_i \)
Step 2. \( M' = M' + 1 \)
Step 3. \( ca'_i = ca_i / 2 \) (Duplicate the popular movies with concurrent access split over them evenly)
Step 4. Goto Step 6 if \( \sum_{i=1}^{M} size_i >= \sum_{k=1}^{N} d_k - E \)
Step 5. Repeat Step 1.
Step 6. Find out all exceeded copies of previously released movies and remove them from the system.
After the replication process, we have to check whether the number of copies of previously released movies existing on the system exceed our newly predicted number of required copies. Any additional copies will be marked for deletion. For example, video 2 in Figure 1 is categorized as previously released movie and 5 copies of video 2 are already in the system. However, the newly predicted number of copies in the new release is 3, then 2 copies of video 2 will be removed from the system.

![Diagram of replication process]

**Figure 1. The replication process.**

### 3.2 Allocation

After movies are replicated, they are then allocated to the servers. We propose an allocation algorithm based on the idea of Matching Ordering Best Fit algorithm [10] to achieve a balanced distribution of movies over the servers. The idea of Matching Ordering is to balance the load of the servers with movie titles with different loading demands, $L_i$. In the Matching Ordering Best Fit Algorithm, movie titles are first arranged in a descending order. Whenever a movie with the highest $L_i$ is placed, one or more movie titles with the lowest $L_i$ are of higher priority to be placed next as long as no new server will be used (i.e., the server that have been used so far will be filled as full as possible with the movie titles with the lowest concurrent access requirements). This will ensure the minimum number of required servers.

As discussed in pervious sections, the mission of this work is not to minimize the number of required servers, but to maximize the utilization of the existing servers. We should make a somehow “invert” to the Matching Ordering Best Fit algorithm.
Therefore, we propose to first place the highest $L_i$ movie to the server, which has the lowest $C_k$. If the load of this server is not the largest among all the servers after placing the movie, we then fill the server with the lowest $L_i$ movies until the server’s $C_k$ is not greater than the highest $C_k$ among all the servers in the system. This will ensure each server has an equal load assigned and balanced with less popular movies. It is observed that no server holds two or more copies of the same movie, since this will not provide more concurrent streams to the movies and it will be a waste of storage capacity.

One important policy is that current replicas of previously released movies should not be reallocated. This policy ensures minimum loading or unloading of movies with high system resource consumption. In doing this, the resultant placement of movies will not be optimal. However, this will converge to optimal if the algorithm repeats for several times.

Our scheme can also handle server failure while the VoD is up and running. In the case of server failure, say two video servers down, movies on these servers have to be reallocated to other servers. A consideration that must be taken into account is that movies in other servers are serving clients and they must be untouched in this situation. Moreover, it is not practical to reallocate all movies as the reallocation process usually takes a few days, and the normal service has to be terminated. In dealing with this situation, we can simply apply the proposed replication and allocation algorithm again, in considering movies in the failure servers as new movie titles, and other existing movies as previously released movies, with the number of servers in the system be $N-2$.

The proposed allocation algorithm can be summarized as follow:

Step 1. Calculate the existing load of the servers, $C_k$.
Step 2. Sort the movie titles in decreasing order with respect to $L_i$.
Step 3. Repeat Steps 4 and 5 until all the movies are allocated to servers.
Step 4. Place the movie with the largest $L_i$ to the server that has the smallest $C_k$.
Step 5. Place zero or more movies with least load (i.e., the movie titles which have not been placed yet and have the smallest $L_i$) until the server’s $C_k$ is not greater than or equal to the current greatest $C_k$.

4. Experimental Results

One of the examples of distributed VoD system is the iTV system of Cable & Wireless HKT in Hong Kong, which is mainly designed by NEC. This is the first interactive multimedia service, via ATM network, to households in the world. Figure 2 shows the system configuration of the iTV system [1].

The MPEG Servers are used to store the movies, which are compressed as 3Mbps MPEG files. Each of the MPEG Servers works as a standalone server and can support up to 72 concurrent streams [1]. The compressed movie will be first loaded to the Distribution server. Placement algorithms, say our proposed algorithms, will then determine how many copies of each movies are required and which MPEG servers the movies should be allocated to.
In order to evaluate the performance of the proposed predictive video allocation algorithm; namely replication and allocation, we implemented replication and allocation in MATLAB. System parameters were based on the above NEC iTV system architecture [1] and some were based on our assumptions. They are given in Table 1.

The popularity of movies in our experiment was randomly assigned. The size of movies was also randomly assigned with the mean size is 2.025GB and the standard deviation is 1GB.

Figure 3a shows the server load before the placement of the newly released movies. Figure 3b shows the server load after the newly released movies are placed to the system, according to our proposed replication and allocation algorithm. It can be seen that the load of the servers was increased after placing the movies, while the load for all servers was equally balanced.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of MPEG server, N</td>
<td>60</td>
<td>Number of Movie, M</td>
<td>600</td>
</tr>
<tr>
<td>Max number of concurrent stream that a server can support</td>
<td>72</td>
<td>Previously released to newly released movie ratio</td>
<td>2:3</td>
</tr>
<tr>
<td>Bit rate of Movie (constant)</td>
<td>3Mbps</td>
<td>Storage capacity of a single server (72*2.025GB)</td>
<td>145.8GB</td>
</tr>
<tr>
<td>Average size of movie (3Mbps * 90mins)</td>
<td>2.025GB</td>
<td>Movie size Standard Deviation (for adding variation on the size during the test)</td>
<td>1GB</td>
</tr>
</tbody>
</table>

Figure 2. System configuration of iTV system.
5. Conclusions

In this paper, we present a predictive video allocation algorithm for video management in commercial distributed VoD system. The proposed algorithm comprises of two major components; namely replication and allocation. The proposed predictive video allocation algorithm is designed for “dirty” system, where previously released movies are already in the servers, and they are taken into consideration when movie allocation takes place.
Experimental results demonstrated that the proposed algorithm achieved a very good load balancing, while reallocation of previously released movies is not needed.

As emphasized in the title of this paper, the proposed video management technique is a predictive video allocation algorithm for using in the pre-deployment phase. The prediction criteria are based on the expected client usage pattern. However, these may not properly reflect the actual user-viewing pattern. It is because the user’s interest may vary from day by day. For example, weekday pattern is different from weekend pattern; newly released movie is getting less popular as time goes by; etc. An adaptive movie reallocation mechanism such as [5, 7] becomes necessary during the deployment phase. The adaptive approach can reallocate movies according to the changing viewing pattern on the fly, and it can also minimize the loading and unloading of movie during reallocation. A proper load management technique, such as [8] is also needed since this adaptive approach requires large amount of system resources. Further improvements of the proposed video allocation algorithm will be an integration of adaptive video reallocation and load management methods in video management.

References


