Efficiently Allocating Video Data in Distributed Multimedia Applications

James Z. Wang Ratan Guha

School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816-2362, U. S. A.
E-mail: {zwang, guha}@cs.ucf.edu

Multimedia data management has been thoroughly studied in the non-distributed multimedia environment. The data management in this environment focuses on device level data allocation and management. In this paper, We discuss the server level data allocation problems in the distributed multimedia systems, especially in the distributed Video-on-Demand systems. We proposed two data allocation algorithms, Bandwidth Weighted Partition (BWP) algorithm and Popularity Based (PB) algorithm, based on the bandwidth and storage capacity limits of the distributed multimedia servers. We compare those two algorithms with the traditional Round Robin (RR) data allocation algorithm. The analysis and simulation studies show that PB algorithm is a simple and practical video data allocation algorithm for the distributed video servers. It provides near optimal system performance in any system conditions. We have also developed a video data allocation tool for Video-on-Demand systems based on BP algorithm.

Keywords: Distributed Multimedia Systems, Data Allocation, Video on Demand

1. Introduction

Technologies have always played an important role in the development of multimedia systems. Recent advances in workstation architecture, high-speed networking, high-speed modem and storage technologies have set the foundation of delivering the streaming media to the ordinary families. Together with the improvement in hardware technologies, advances in software technologies, such as data compression algorithm, Internet browser and streaming media player, have brought the Video-On-Demand (VOD) from concept to reality. On the other hand, the rapid growth of the Internet and WWW has driven the technology industry into a new arena in which web hosting and content hosting have become very popular multimedia applications. Many companies operate several distributed data centers where the other companies' Web servers and Web contents are hosted. To succeed in this rising industry, one of the key factors is to efficiently manage the network bandwidth and storage capacity among those distributed multimedia servers.

Most of recent research studies have focused on multimedia servers in non-distributed environment. Because two most important features of multimedia data are the large storage and bandwidth requirements, multimedia data management strategies have taken the center stage in those studies. Most of these works deal with storage management and data allocation among various storage devices in a non-distributed environment. Some of them have investigated the storage management techniques in the hierarchical storage systems [Wang (1996), Wang (1997A), Triantafillou (1997), Christodoulakis (1997), Kraiss (1997)]. The others have focused on the data management techniques in the disk-array-based storage systems [Wang (1997B), Chua (1996), Berson (1994), Berson (1995)]. Although there are a few studies discussing the data allocation issues in the distributed multimedia environment

[Brubeck (1996), Gafsi (1999)], those works have focused on how to manage the multimedia data among the different storage devices. The objective of those data management schemes is to efficiently manage the multimedia data among different storage devices in the storage system, in order to provide smaller user request latency and larger system throughput. We categorize those data management schemes as device level data management. The principle of the device level data management is to allow the most frequently requested data accessed through the online devices and migrate the data in the near-line/off-line devices into online devices as fast as possible.

In this study, we examine the problems associated with the data allocation among the distributed multiple video servers located in different geographical locations across a WAN environment. This environment is different from that of the clustering servers where all servers are located in the same location and are sharing the network bandwidth. We treat those clustering servers as a single server in our distributed environment. Instead of investigating the device level data management, we discuss the server level data management. At the server level, we only see the online multimedia data that are ready to be delivered to the clients when requests arrive. How to make the data online is the problem of the device level data management. We analyze the impact of the online multimedia data allocation among the distributed servers to the system performance in terms of the average request latencies. We discuss three online video data allocation algorithms and compare their performance using simulation studies. The purpose of this research is to find the best data allocation algorithm for a video data allocation tool used by distributed VOD systems.

The rest of the paper is organized as follows. We discuss the distributed multimedia environment in section 2 and raise the problems with the data allocation policies. In section 3, the distributed multimedia data allocation algorithms are proposed and discussed in details. The advantages and disadvantages of each algorithm are also addressed in this section. In section 4, we use simulation to model the distributed VOD servers and compare the proposed algorithms based on the system performance in terms of the average user request latency. We conclude our current study and discuss the future work in section 5. We give the acknowledgement in section 6.

2. Distributed Multimedia Environment

A distributed multimedia service environment consists of multiple multimedia servers located in different data centers across various geographical locations. Of course, there are many reasons for having such a distributed multimedia service environment. Those reasons include reliability of the services, geographical demands and company strategies. Figure 1 depicts the typical distributed multimedia environment. The data servers store all the multimedia data. The web servers or presentation servers present the data to the end user's workstations. For the streaming media services, such as Video-on-Demand, the presentation servers connect the user's workstations to the data servers and allow the data servers to deliver the streaming media to the user's workstations through the established channels. The storage devices in the data servers consist of online storage devices, such as disk-array or disk farm, and off-line or near-line storage devices, such as tape libraries and DVD jukeboxes.

Besides the problems in the non-distributed environment, there are many challenges in the distributed environment. For instance, the presentation of the distributed multimedia data requires additional synchronization and scheduling. There are many outstanding studies dealing with this issue [Jarmasz (1997), Mielke (1999), Song (1999)]. In some cases, the presentation server needs to retrieve the multimedia data from the distributed data servers, then composite them and deliver the data to the user's workstations [Hwang (1998)]. In the

other cases, such as streaming media services, the presentation server needs to connect the user's workstation to the distributed data servers and manage the streaming channels between the workstations and the data servers. As in the non-distributed environment, data allocation, retrieval and delivery are most important problems in the distributed multimedia environment [Jarmasz (1997), Hwang (1998), Mielke (1999), Song (1999), Billot (1997), Candan (1999), Luling (1999)]. In this study, we focus on the distributed data allocation problem and investigate the impact of the online data allocation to the system performance in the distributed environment. We find that the storage system capacity is not the sole factor used to determine the data allocation policy for the distributed video servers. There are some other important factors such as server bandwidth and data access pattern. Our data allocation algorithms include all those factors into consideration.

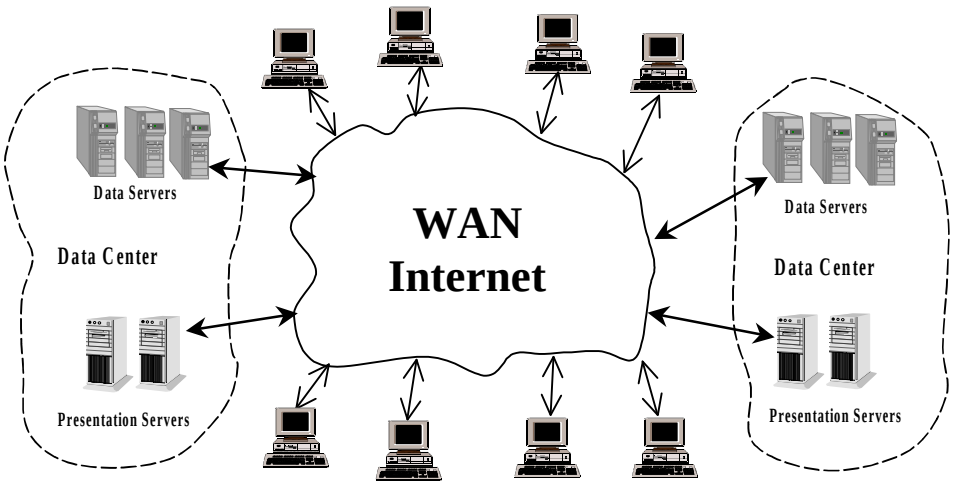

Figure 1. Distributed Multimedia Environment

In our algorithms, there are two very important factors need to be examined for designing the data allocation algorithms in the distributed Video-On-Demand environment. One factor is the online storage capacities of the video servers. The online storage capacity determines how much data the server can provide that are ready to be delivered without device level delays caused by migration. The other factor is the effective bandwidths of the video servers. The effective bandwidth of the server is decided by many factors. They are I/O bandwidth, network bandwidth, system bus bandwidth, etc. The server I/O bandwidth is the realized (by outsider) bandwidth of the storage devices in this server. The network bandwidth is the sum of the bandwidths of the network devices in this server. We must note that the effective bandwidth of the server are not necessarily the total advertised network bandwidth of the network devices or the total advertised I/O bandwidth of the storage devices in the server. For example, if the servers in a data center share a limited fiber network bandwidth which is less than the total advertised bandwidths of the network devices in the data center, the effective bandwidth for each server will be determined by the network traffic of the data center and some other factors. Those other factors include hardware conditions of the servers, the operating systems in the servers, and efficiency of the application software running on the servers. We simply call the effective bandwidth of a server as the server bandwidth. The server bandwidth can be obtained through experiment.

For convenience of the discussion, we model the distributed multimedia system in Figure 2. We present each server with a server bandwidth B and an online storage capacity C. Server i can be represented as (B_i, C_i). Given a set of video objects $\{v_1, v_2, \cdots, v_m\}$, the goal of the distributed multimedia data allocation algorithm is to distribute the objects across the servers $(B_1, C_1), (B_2, C_2), \cdots, (B_n, C_n)$, in order to achieve the best system performance. Usually we have $n << m$. We discuss the data allocation algorithms in the following sections.

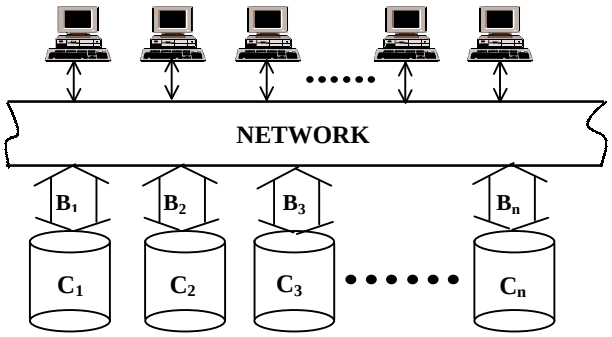

Figure 2. Model of the distributed multimedia system

3. Data Allocation Algorithms for Distributed Video Servers

Besides the server bandwidth and storage capacity, the video data set itself is the most important factor in deciding the data allocation algorithms. For each video object v_i, there are two attributes contributing to the data allocation algorithm. One is the video object size. It determines how much server storage capacity this video object will take. The other is the user request frequency of the video object determining the number of concurrent streams accessing this video object. This in turn determines the server bandwidth requirement for the object. Given a set of video objects, the user access frequencies to the video objects are drastically different. In the survey conducted by [Dan (1994)] showed that the access pattern to the videos in a video renting store follows a zipf-like function with $z = 0.273$. That means the user access to video objects is severely skewed. A few video objects contribute most of the user requests to the video object database. The minorities of video objects are very hot and attract the most attention, while the majority of the video objects are cold and only get scarce requests. The goal of the video allocation algorithm is to distribute the video objects to the video servers so that the respective servers can handle the user requests with minimum request latencies. We discuss the video allocation algorithms in the following subsections.

3.1 Round Robin Allocation Algorithm

We know the user access pattern to the video objects is highly skewed. If we allocate all the hot video in one server, this server will have to handle most of the user requests while the other servers are mostly idle. Obviously this video allocation is not good. Conventionally we assume that each server has similar process power, i.e., they can handle equal number of concurrent streams at the same streaming rate. So the task of the video data allocation is to evenly distribute the hot video objects to the distributed servers so that no one server will have to handle most of the user requests. Let us assume the video object set is $V = \{v_1, v_2, \cdots, v_m\}$ and the corresponding user access frequency set is $F = \{f_1, f_2, \cdots, f_m\}$, where $\sum_{i=1}^{m} f_i = 1$. Without losing generality, we also assume $f_1 \geq f_2 \geq \cdots \geq f_m$. We assume the server set is $\{S_1, S_2, \cdots, S_n\}$ with the storage capacities $\{C_1, C_2, \cdots, C_n\}$, respectively.

The idea of the *Round Robin* (RR) algorithm is depicted in Table 1. The actual algorithm varies slightly from this basic idea because we need to consider the storage capacity of the servers. In the actual algorithm, we first use the *Round Robin* idea on the entire set of servers.

During the process, if there is a server running out of the storage capacity, we take this server out of the rotation and continue using the same process on the remaining servers. We assume the storage requirement for object v_i is C_{vi}. Then $C_v = \sum_{i=1}^{m} C_{vi}$ will be the video database size. For each server S_i, if we assume its storage capacity $C_i \geq \frac{C_v}{m} + Max\{C_{v1}, C_{v2}, ..., C_{vm}\}$, we can guarantee the algorithm successfully distribute the entire video database to the video servers.

Table 1. Round Robin video data allocation

S_1	S_2	……	S_{n-1}	S_n
v_1	v_2	……	v_{n-1}	v_n
v_{2n}	v_{2n-1}	……	v_{n+2}	v_{n+1}
……	……	……	……	……

This algorithm tries to evenly distribute the workloads among the distributed servers. When those video servers have the same server bandwidth, the *Round Robin* data allocation algorithm yields good system performance. However, in the real distributed multimedia systems, the bandwidths of the video servers are different. Usually the newer server has better system processing power, higher storage capacity, higher storage I/O bandwidth and larger network bandwidth. If we uniformly distribute the workloads among the servers with different server bandwidths, either we waste the system resources in the servers with higher server bandwidth or we overload the servers with lower server bandwidths. To solve the problem, we propose the *Bandwidth Weighted Partition* (BWP) data allocation algorithm.

3.2 Bandwidth Weighted Partition Algorithm

Instead of letting a single server handle the user requests for a specified object, we allow video servers to coordinate the delivery of the data for a single video object to the client's workstations. We partition each video object into the video servers in the way that the servers with higher server bandwidth will be allocated with a larger portion of the video data. BWP algorithm is presented in Table 2. We assume the server set is $\{S_1, S_2, \cdots, S_n\}$ with the respective server bandwidths $\{B_1, B_2, \cdots, B_n\}$. For the object v_i, we partition its data into n continuous chunks $v_{i1}, v_{i2}, ..., v_{in}$ where $size(v_{ij}) = size(v_i) \bullet \frac{B_j}{\sum_{k=1}^{n} B_k}$. The data partition for a single object v_i is depicted in Figure 3.

Table 2. Bandwidth Weighted Partition Algorithm

S_1	S_2	……	S_{n-1}	S_n
v_{11}	v_{12}	……	$v_{1,n-1}$	$v_{1,n}$
v_{21}	v_{22}	……	$v_{2,n-1}$	$v_{2,n}$
……	……	……	……	……

Object v_i:

v_{i1}	v_{i2}	v_{i3}	……	$v_{i,n}$

Figure 3. Partition the video object

This algorithm partitions the video data based on the server bandwidths. It balances the server bandwidth utilization very well and hence derives excellent system performance (See section 4). Our study shows the striping algorithms used in the device level data management

[Berson (1994), Chua (1996), Wang (1997B)] are not feasible in our distributed multimedia environment. If the presentation server is responsible for retrieving the data from the data servers and delivering them to the client's workstation, the presentation server has to maintain n real time connection channels to n different data servers for each video stream. Then the presentation server becomes a hot spot.

Although we can use some batching algorithms [Dan (1994), Aggarwal (1996)] to reduce the number of concurrent streams for each video object, there are still too many real time channels in the presentation server for each object unless the users can tolerant large request latencies. If the presentation server is only responsible to direct the client's request to the data servers, the client's workstation needs to retrieve the data from the servers and processes the video data itself. Processing the video data at the client's workstation is not a problem considering how powerful the current workstations are. The bottleneck of the client's workstation is its limited network bandwidth. Usually those clients are connected to the Internet through the high-speed modems. It is not practical to require the client's workstation to maintain n real time connections to n different data servers. In this case, the workstation needs to reconnect frequently to the distributed servers; thus the overhead is too high. Due to the above reasons, we do not use the striping techniques in our data allocation algorithms. Actually, even in the device level data allocation, the striping techniques do not always provide better system performance [Reisslein (1999)]. Once the technology advances to the level that there is no network bandwidth bottleneck in the client's workstations, the striping technique may be investigated as an alternative to BWP algorithm.

Although BWP algorithm balances the server bandwidth utilization very well, it has ignored another important factor of data allocation we discussed before. It does not consider the storage capacities of the video servers. To use this algorithm, we require the server with higher server bandwidth to be equipped with larger storage capacity. In the device level data management, we often assume that the newer hard disks have higher I/O bandwidth with higher storage capacity than the older hard disks because of the technology advances. Nevertheless, this is not always the same at the server level. The server bandwidth we discussed here is determined by many factors. Two servers identical in hardware can be realized with different effective server bandwidth if they are placed in two different data centers. The general network traffic conditions of those two data centers are different. The operating system and the application software have impact on the realized server bandwidths. For example, a typical hard disk's advertised I/O bandwidth is 40 MB/Sec. However, the maximum disk I/O bandwidth in the Microsoft SQL Server under the Windows NT operating system is only 9.6 MB/Sec using the 64-KB transfer size [Lau (1998)]. To solve the problem, one solution is to increase the server storage capacity for the higher bandwidth servers. Of course, this is not a good idea economically if we can design a good data allocation algorithm without increasing the hardware. On the other hand, many people don't want to partition the video object across the distributed data servers because it increases the presentation complexity. The system has to guarantee there is no jitter or delay on presentation during the data server switch. To address the problems stated above, we propose the *Popularity Based* (PB) data allocation algorithm in the next subsection.

3.3 Popularity Based Data Allocation Algorithm

A good data allocation algorithm for the distributed multimedia systems should balance the server bandwidth utilization of the video servers while maintaining their storage capacity constraints. If we assume all video objects have equal size, and the video servers have the same storage capacity, and the user requests to the video objects are uniformly distributed, i.e., all the video objects have equal opportunity being requested by the users, then we have to put more objects in the higher bandwidth servers to balance the bandwidth utilization. But it

violates the storage constraints on the higher bandwidth servers. Fortunately the user access to the video objects is highly skewed. That means some video objects are very hot and get most of the user requests while the rest objects get only a few user requests. The video data access pattern is also predictable according to study by [Choi (1999)]. For instance, the new movies are usually hotter than the old ones. Thus the newer video objects have higher user requests and older video objects have lower user requests. We exploit this video data skew condition to design our PB algorithm. As in RR algorithm, we assume the video object set is $V = \{v_1, v_2, \cdots, v_m\}$ with the corresponding user access frequencies to the video objects is $F = \{f_1, f_2, \cdots, f_m\}$ where $\sum_{i=1}^{m} f_i = 1$. We also assume the server set is $\{S_1, S_2, \cdots, S_n\}$ with the storage capacities $\{C_1, C_2, \cdots, C_n\}$ and server bandwidths $\{B_1, B_2, \cdots, B_n\}$, respectively. The idea of the PB algorithm is to let the server with higher bandwidth handle larger portion of the user requests. Without losing generality, we further assume the access to each object requires equal server bandwidth. To allocate the video objects, we first calculate what percentage of the total user requests a certain video server may handle based on its server bandwidth. We transform the server bandwidths $\{B_1, B_2, \cdots, B_n\}$ into their normalized server bandwidths $\{\overline{B_1}, \overline{B_2}, \cdots, \overline{B_n}\}$ where $\overline{B_i} = \frac{B_i}{\sum_{j=1}^{n} B_j}$. Obviously we have $\sum_{i=1}^{n} \overline{B_i} = 1$. Since we have $n << m$, it is reasonable to assume $f_i < \overline{B_j}$ for any $i \in [1, m]$ and $j \in [1, n]$. It is also reasonable to assume the total storage capacity of the servers is not less than the entire video database size, i.e., $\sum_{i=1}^{n} C_i \geq \sum_{j=1}^{m} size(v_j)$. Thus the ideal video data allocation algorithm divides the video set V into n subsets $\{V_1, V_2, ..., V_n\}$ and allocates subset V_i to server S_i where $i \in [1, n]$, so that we have $C_i \geq \sum_{j=1}^{k} size(v_{ij})$ and $\overline{B_i} = \sum_{j=1}^{k} f_{ij}$ for any video object subset $V_i = \{v_{i1}, v_{i2}, \cdots, v_{ik}\}$. However, implementing such an ideal algorithm requires $C_m^n = \frac{m \cdot (m-1) \cdot \cdots \cdot (m-n+1)}{n \cdot (n-1) \cdot \cdots \cdot 2 \cdot 1}$ computation complexity. Furthermore, there is no guarantee the optimal solution exists. So we design the approximate video allocation algorithm as depicted in Figure 4. Without losing generality, we also assume $f_1 \geq f_2 \geq \cdots \geq f_m$. In the algorithm, RC(j) represents the remaining storage capacity for server j and RB(j) stands for the remaining server bandwidth for server j. We assume the video object sizes are $\{s_1, s_2, \cdots, s_m\}$. This algorithm first finds the object with highest access frequency and the server with largest normalized server bandwidth. If the server has enough storage capacity to contain the object, it allocates that object to the server. Otherwise, it takes this server out of the consideration. After the allocation, it reduces the normalized bandwidth of this server by the object's access frequency. This algorithm repeats the same process on the remaining objects and servers. Similar to RR algorithm, for each server S_i, if we assume its storage capacity $C_i \geq \frac{C_v}{m} + Max\{C_{v_1}, C_{v_2}, \cdots, C_{v_m}\}$, we can guarantee the algorithm successfully distributes the entire video database to the video servers. For any server S_i and its allocated video object subset $V_i = \{v_{i1}, v_{i2}, \cdots, v_{ik}\}$, this algorithm ensures $C_i \geq \sum_{j=1}^{k} size(v_{ij})$. But we might not have $\overline{B_i} = \sum_{j=1}^{k} f_{ij}$. However, we can have $\overline{B_i} \approx \sum_{j=1}^{k} f_{ij}$ since we allocate the

hot objects first. Those hot objects contribute most of the user requests in the real VOD system. Our performance study will validate this assessment.

```
Algorithm : Popularity_Based_Allocation
  for(i = 1; i < n; i++)   RB(i) = B̄i;
  ServerNum = n;
  for(i = 1; i < m; i++){
    Find the server j with the largest RB;
    if(RC(j) ≥ si){
      RC(j) = RC(j) − si;    RB(j) = RB(j) − fi;
      Allocate Object i to server j;
      Consider all servers for the next object;
      ServerNum = n;;
    }
    else {  /* This server is full, take it out of rotation */
      Delete server j from consideration;
      ServerNum = ServerNum − 1;    i = i − 1;
      if(ServerNum = 0)
        Fail due to lack of storage capacity;
    }
  }
```

Figure 4. Popularity Based Video Data Allocation Algorithm

4. Performance Study

Based on the discussion in the previous section, the *Popularity Based* (PB) allocation algorithm addressed all the problems we have observed in the data allocation for the distributed Video-On-Demand servers. However, this algorithm is only an approximation to the ideal situation. We need to determine how close it performs compared to the ideal situation in a real distributed VOD system. We have developed a detailed simulator to compare its performance to the other algorithms. In this section, we first describe the simulation model, and then discuss the simulation results.

4.1 Simulation Model

Our simulator consists of four components. The structure of the simulator is depicted in Figure 5. The *Request Generator* is responsible for creating requests. Each request is characterized by its arrival time and the object requested. Requests arrive at the *Dispatcher* that simulates the presentation server. The *Dispatcher* routes the requests to the server where the requested object is located. There are four servers in our simulator. Each server is characterized by its storage capacity and server bandwidth. The requests are submitted to a FIFO queue associated with each server where they wait for admission. A simple admission algorithm is used in each server. If there is enough server bandwidth for servicing the request, the server reserves the requested bandwidth till the video playback completes.

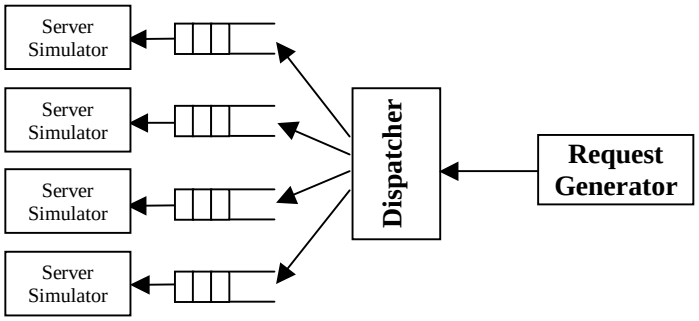

Figure 5. Structure of the simulator

For BWP algorithm, our simulator takes a simple approach for the performance study. In a real VOD system, we have to switch the data servers during the video display process if BWP algorithm is used. This server switch process requires extra process effort and server bandwidth to ensure the quality of services. However, the assurance of the quality of services is beyond the scope of this study. Theoretically, during a long system-running period, the server bandwidth utilization is equal for all servers in the system using BWP algorithm. So the average bandwidths for every servers used in one object are proportional to their server bandwidths. Our simulator keeps all the requests in one FIFO queue for BWP data allocation algorithm. A request is granted only when there are enough server bandwidth in all servers. But for each server, the required server bandwidth is just a portion of the object's required bandwidth. For instance, we have four servers $\{S_1, S_2, S_3, S_4\}$ with server bandwidth $\{B_1, B_2, B_3, B_4\}$. The required bandwidth for video object v is B_v. Then for server S_i, the required server bandwidth is $B_{vi} = B_v \bullet \frac{B_i}{B_1+B_2+B_3+B_4}$. This server bandwidth is reserved for the object until the video playback completes. This simulation approach for BWP data allocation algorithm not only simplifies our performance study but also provides optimal results that we can use to compare with. Comparing the performance of the other algorithms with that of the optimal BWP algorithm, we can determine the performance of other algorithms. In our study, since PB algorithm is an approximation, we can determine how close it performs compared to the optimized solution. For simplicity, we assume that all the video files are 1-hour MPEG-2 encoded videos. The playback rate is *6 Mbits/sec*. They all have the same video size of *2800 MBytes*. User request inter-arrival time is modeled using a Poisson process with the average inter-arrival time of 5 seconds. The access frequencies of objects in the database follow a zipf-like distribution. The access frequency for each video object v_i is determined as follows:

$$f_i = \frac{1}{i^z \cdot \sum_{j=1}^{m} 1/j^z},$$

where m is the number of objects in the system, and $0 \le z \le 1$ is the zipf factor [Wang (1997A)]. A larger z value corresponds to a more skew condition, i.e., some objects are accessed considerably more frequently than other objects. When $z = 0$, the distribution is uniform, i.e., all the objects have the same access frequency. This zipf-like distribution is similar to the distribution used in [Dan (1994)]. But there are some differences between those two zipf-like functions. In [Dan (1994)], a smaller z value represents a more skew condition while a larger z value stands for a more uniform distribution. The distribution of $z = 0.273$ using the zipf-like function in [Dan (1994)] is very close to the distribution of $z = 0.7$ using our zipf-like function. We summarize the simulation parameters in Table 3. We use those parameters to perform our simulation unless otherwise stated. There are many ways to evaluate the performance of the replacement policies. However, users are most concerned about the latency of their requests. So we use average request latency as our performance metric.

Table 3. Simulation parameters

Server Storage Capacity	675,000 MB (≈ 659 GB)
Playback rate B_p	6 Mbits/sec
Zipf factor	0.7
Requests inter-arrival time	5 seconds
Number of objects	1000
Object size	2,700 MB
Number of requests	20,000

4.2 Determine the server bandwidth

First we need to determine a reasonable server bandwidth for us to use to compare the difference of the algorithms. If the server bandwidth is unlimited or too large, the average request latency becomes zero for all data allocation algorithms. In that case, any data allocation algorithm can be selected. If the server bandwidth is too small, the average request latency is too large for users to tolerate. We assume all servers have the same storage capacity of 659 GB. Each can store 250 video objects. Their server bandwidths are also the same. We vary the server bandwidth between 500 MB/Sec and 600 MB/Sec. We apply the three different data allocation algorithms to the system. To make sure every algorithm gets the same workload, the *Request Generator* creates the request sequence first and applies the same sequence to the system under three different data allocation algorithms. We compute the average request latencies for a total of 20,000 requests and get the simulation results depicted in Figure 6.

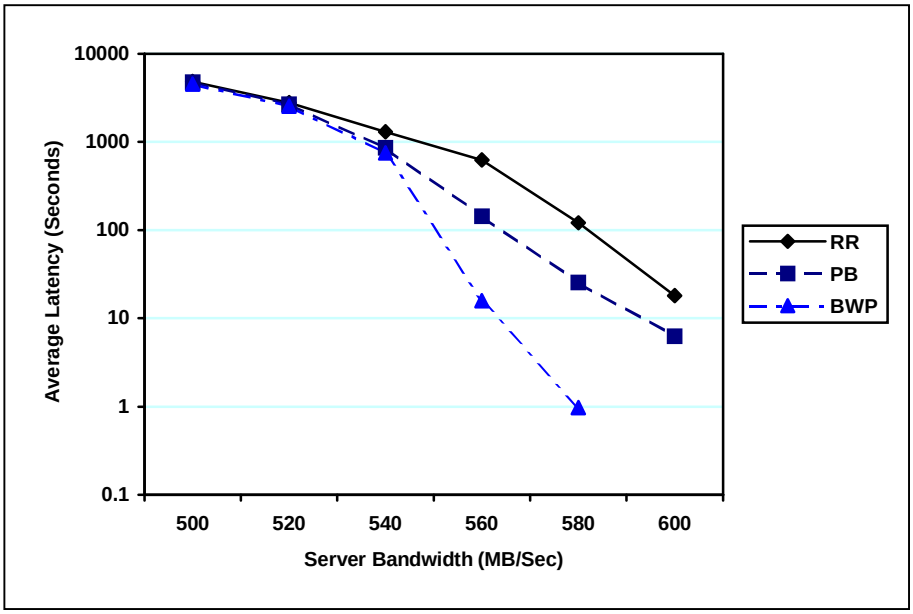

Figure 6. Determine the server bandwidth

As we expected, BWP algorithm gives us the best performance among the three algorithms. When the server bandwidth is only 500 MB/Sec, all three algorithms derive very large average request latencies. They are all over 75 minutes. It is obviously not practical to require the users to wait for 75 minutes before viewing a 1-hour video. When the server bandwidth increased to 600 MB/Sec, the average latency for BWP algorithm dropped to zero. The average latency for PB algorithm is about 6 seconds and RR algorithm has 18 seconds average latency under this server bandwidth. We choose the average server bandwidth to be 580 MB/Sec in the rest of performance study because BWP algorithm has non-zero average

latency, which is good for drawing the comparison figures. On the other hand, the other two algorithms have practical average latencies with PB algorithm at 25 seconds and RR algorithm at 120 seconds. Because the average latencies on the plots vary too much, we use logarithmic scale on the average latency to draw the figure.

4.3 Performance under various data skew conditions

As we discussed before, the data access skew condition plays a very important role in the performance of the video data allocation algorithms. In this simulation, We varied the skew factor between 0.0 (a uniform pattern) and 1.0 (a severe skew condition). The server bandwidth for each server is 580 MB/Sec and each server has a storage capacity of 659 GB. The results are plotted in Figure 7. It shows BWP algorithm does not affect by data skew condition at all due to two factors. First, this algorithm balances the bandwidth utilization very well. Second, the way we simplified the performance study for this algorithm made the algorithm the best. Actually the average latency presented by BWP algorithm is the optimal results. When the data skew condition was mild (less than 0.5), RR algorithm performed well with the average latency less than 40 seconds. However, when the data skew condition became severe, the average latency for this algorithm increased drastically. PB algorithm performed very well in any access skew condition. All of its latencies were less than 40 seconds which is practical for a 1-hour video.

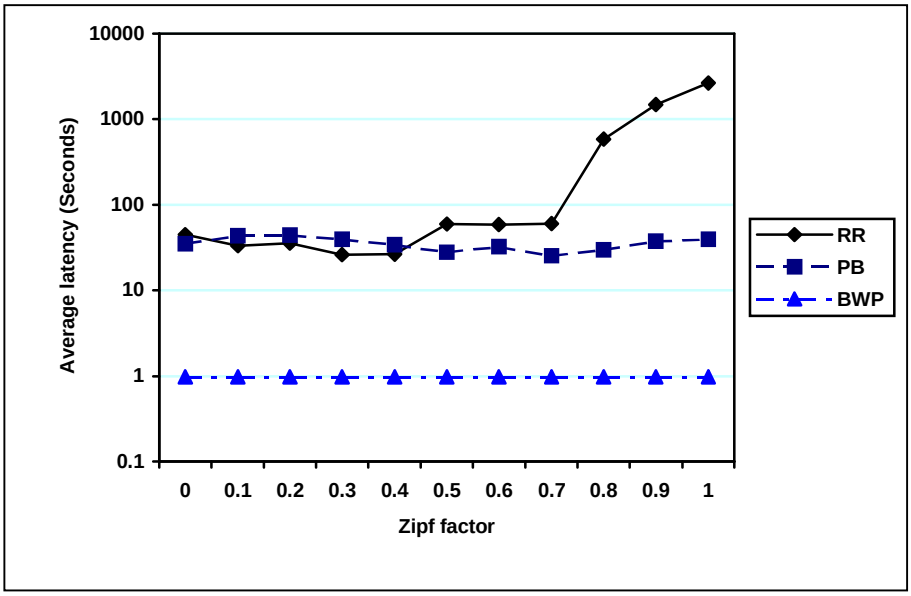

Figure 7. Performance under different access skew

4.4 Servers have different bandwidths

In the previous two performance studies, we assumed the servers have the same server bandwidth. However, in the real distributed Video-On-Demand systems, the server bandwidths are different in different video servers. We studied system performance under this condition. We let the servers have an average bandwidth of 580 MB/Sec. Their individual server bandwidth were different as $B_1 = 505$ MB/Sec, $B_2 = 555$ MB/Sec, $B_3 = 605$ MB/Sec and $B_4 = 655$ MB/Sec. We also assume each server has enough storage capacity to complete all three algorithms. We can see that server S_4 requires 30% more

storage capacity than server S_1 does if BWP algorithm is used. We varied the skew factor between 0.0 (a uniform pattern) and 1.0 (a severe skew condition) as in the previous simulation. The results are plotted in Figure 8. It shows that the average latencies for RR algorithm turned out to be very high in any data skew condition. It is practically useless in this situation. On the other hand, BWP algorithm and PB algorithm performed very well. Comparing the Figure 8 and 7, we notice that BWP and PB algorithms were not affected by the server bandwidth changes as long as there are enough storage capacity in the servers to allow those algorithms to distribute the user requests based on the server bandwidths.

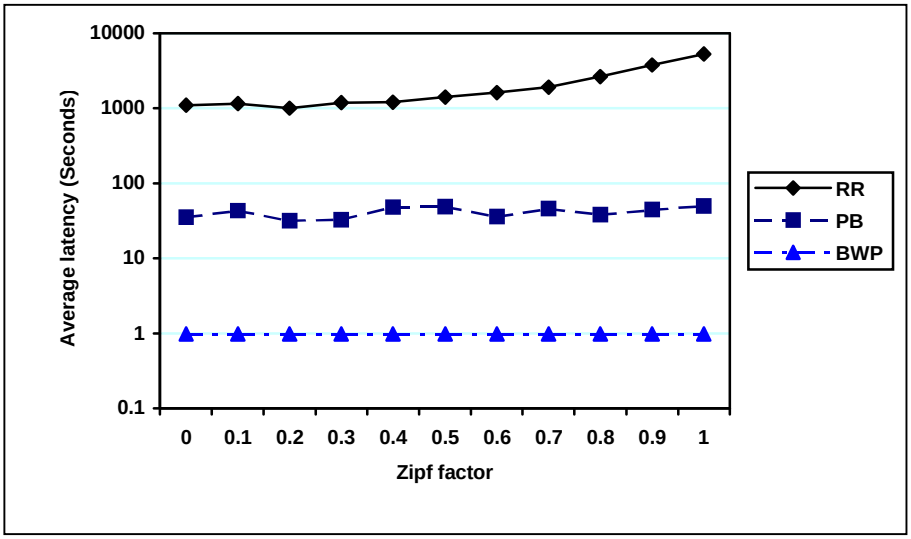

Figure 8. Performance when servers have different bandwidth

4.5 The impact of the server storage capacity

In the previous simulation, we assumed all the servers have enough storage capacity to distribute the objects by the data allocation algorithms. However, in a real VOD system, it is not always the case. For instance, two hardware identical servers have different server bandwidth when placing in different locations. In this situation, BWP algorithm and PB algorithm have to comply with storage limitation on the servers. We studied the similar situation using our simulator. We let all servers have same storage capacity of 659 GB, i.e., each server could contain 250 video objects. The server bandwidth in the servers were $B_1 = 505$ MB/Sec , $B_2 = 555$ MB/Sec , $B_3 = 605$ MB/Sec and $B_4 = 655$ MB/Sec . Because the servers have equal storage capacity, the data partition in BWP algorithm should not be based on the server bandwidths. Each server has to store equal portion of an object. We also varied the zipf factor in the simulation. the results are presented by Figure 9.

The simulation shows that RR algorithm and BWP algorithm are practically useless in this situation. Their average latencies were all over 4000 seconds. When the zipf factor was zero, i.e., all objects had equal opportunity being accessed, the average latency for PB algorithm was almost the same as that for the other two algorithms. This result is not surprising at all because there is no way for PB algorithm to distribute the user requests to the servers based on their server bandwidth due to the server storage constraints. However, when the data skew factor is increased, PB algorithm takes advantage of the data access skew condition and put the hotter objects in the higher bandwidth servers. Thus the average latency dropped when the skew factor increased in PB algorithm. When the zipf factor was greater than 0.5, the average latency time dropped below 1 minute which is reasonable for 1-hour video objects.

Based on the results of simulation we expect PB algorithm will perform very well in the real distributed VOD systems.

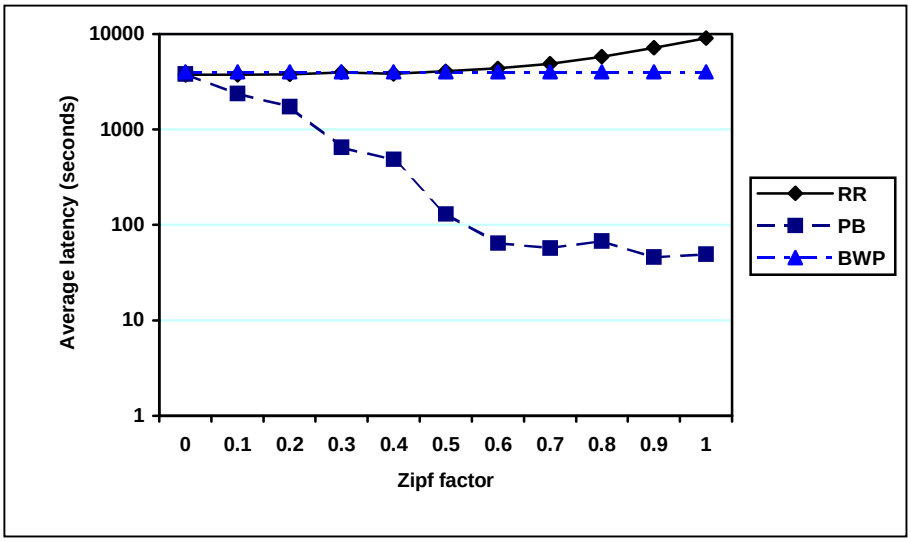

Figure 9. Servers have equal storage capacity but different bandwidth

5. Concluding Remarks

In this paper, we thoroughly discussed the data allocation problems for the distributed multimedia applications. We proposed two different algorithms, *Bandwidth Weighted Partition* (BWP) algorithm and *Popularity Based* (PB) algorithm, to address the problems that affect the system performance. We compared those two algorithms with the conventional *Round Robin* (RR) data allocation algorithm. Through our analysis and simulation, we found BWP algorithm balanced the server bandwidth utilization very well if the server storage capacity is not limited. Its performance is not affected by the user request patterns. In the ideal situation, BWP algorithm provides the optimal system performance. However, there are several problems associated with this algorithm. This algorithm requires the higher bandwidth servers have larger storage capacity. The ideal situation is the server storage capacity proportional to the server bandwidth. But those conditions are not guaranteed in the real VOD systems. On the other hand, partitioning the data across the servers causes problems in video presentation. We need extra processing effort and extra bandwidth to ensure there is no jitter or delay when switching from one data server to another data server. PB algorithm does not partition the video objects. It allocates the video objects based on their access frequencies. This algorithm not only factors in all resource constraints in video servers but also takes advantage of the access skew to the video objects. The simulation results show it provides near optimal results in any condition. Thus, in the VOD systems where the data access skew condition is usually severe, PB algorithm is an excellent video data allocation algorithm. We have developed a video data allocation tool based on BP algorithm. This tool detects the effective server bandwidth based on experimental sampling and then instructs the servers to load proper video data online.

The effective server bandwidth is not necessarily the total advertised bandwidth of the network devices or the total I/O bandwidth of the storage devices of the server. There are other factors that affect the effective server bandwidth. Those factors are network traffic of the data center, physical condition of the hardware, the operating systems in the servers, and

the efficiency of the application software running in the servers. Our future study will investigate the best method to determine the effective server bandwidth. It is one of the most important parameters for data allocation algorithms. We will also study how the effective server bandwidth changes dynamically affect our data allocation algorithms in the real VOD systems.

6. Acknowledgement

The authors acknowledge the partial support provided by the Army Research Office (Contract No. DAAH04-95-1-0250). The views and conclusions herein are those of the authors and do not represent the official policies of the funding agency.

References:

Aggarwal, C. C., Wolf, J. L., and Yu P. S. (1996). On Optimal Batching Policies for Video-On-Demand Storage Servers. *In Proc. Of the IEEE int'l Conf. On Multimedia Computing and Systems*, pages 253-258, Hiroshima, Japan, June 1996.

Berson, S., Ghandeharizadeh, S., Muntz, R., and Ju, X. (1994). Staggered striping in multimedia information systems. *In Proc. Of ACM SIGMOD Conference*, pages 79-90, May 1994.

Berson, S., Golubchik, Leana, and Muntz, R. (1995). Fault tolerant design of multimedia servers. *In Proc. Of ACM SIGMOD Conference*, pages 364-375, May 1995.

Billot, M., Puaut, I., Issany, V., and Banatre, Michel (1997). Improving reliability of distributed VOD servers. *In Proc. Of the International Conference on Multimedia Computing and Systems'97*, pages 253-260, Ottawa, Ontario, Canada, June 1997.

Brubeck, D. W. and Bowe, L. A. (1995*).* Hierarchical storage management in a distributed VOD system. *IEEE Multimedia*, pages 37-47, Fall 1996.

Candan, K. Selcuk and Yang, YiFeng (1999). Least-cost high-quality object retrieval for distributed multimedia collaborations. *In Proc. Of the International Conference on Multimedia Computing and Systems'99*, volume 1, pages 649-654, Florence, Italy, June 1999.

Chua, T.-S., Li, J., Ooi, B., and Tan, K-L. (1996). Disk striping strategies for large video-on-demand servers. *In Proc. Of ACM Multimedia*, pages 297-306, Boston, MA, November 1996.

Choi, Tae uk, Kim, Young-Ju, and Chung, Ki-Dong (1999). A prefetching scheme based on analysis of user pattern in news-on-demand system. *In Proc. Of ACM Multimedia*, pages 145-148, October 1999.

Christodoulakis, Stavros, Triantafillou, Peter, and Zioga, Fenia A. (1997). Principles of optimally placing data in tertiary storage libraries. *In Proceedings of the 23rd VLDB conference*, pages 236-245, Athens, Greece, 1997.

Dan, A., Sitaram, D., and Shahabuddin, P. (1994). Scheduling policies for an on-demand video server with batching. *In Proc. Of ACM Multimedia*, pages 407-416,October 1994.

Gafsi, Jamel and Biersack, Ernst W. (1999). Performance and reliability study for distributed video servers: Mirror or parity?. *In Proc. Of the International Conference on Multimedia Computing and Systems'99*, volume 2, pages 628-634, Florence, Italy, June 1999.

Hwang, Eenjun, Subrahmanian, V. S., and Prabhakaran, B. (1998). Distributed video presentations. *In Proc. Of the Int'l Conf. On Data Engineering*, pages 268-275, Orlando, Florida, February 1998.

Jarmasz, J. and Georganas, N. (1997). Designing a distributed multimedia synchronization scheduler. *In Proc. Of the International Conference on Multimedia Computing and Systems'97*, pages 415-423, Ottawa, Ontario, Canada, June 1997.

Kraiss, Achim and Weikum, Gerhard (1997). Vertical data migration in large near-line document archives based on markov-chain predictions. *In Proceedings of the 23rd VLDB conference*, pages 246-255, Athens, Greece, 1997.

Lau, Henry (1998). Microsoft SQL server 7.0 performance tuning guide. *Technical Report 098-81529*, http://www.microsoft.com/sql, 1998.

Luling, Reainhard (1999). Managing large scale broadband multimedia services on distributed media servers. *In Proc. Of the International Conference on Multimedia Computing and Systems'99*, volume 1, pages 320-325, Florence, Italy, June 1999.

Mielke, Markus and Zhang, Aidong (1999). Optimally ensured interactive service in distributed multimedia presentation systems. *In Proc. Of the International Conference on Multimedia Computing and Systems'99*, volume 1, pages 661-666, Florence, Italy, June 1999.

Reisslein, Martin, Ross, Keith W., and Shrestha, Subin (1999). Striping for interactive video: Is it worth it?. *In Proc. Of the International Conference on Multimedia Computing and Systems'99*, volume 2, pages 635-640, Florence, Italy, June 1999.

Song, Yuqing, Mielke, Markus and Zhang, Aidong (1999). Synchronized streaming of multimedia presentations in distributed environments. *In Proc. Of the International Conference on Multimedia Computing and Systems'99*, volume 2, pages 585-590, Florence, Italy, June 1999.

Triantafillow, Peter and Papadakis, Thomas (1997). On-demand data elevation in a hierarchical multimedia storage server. *In Proceedings of the 23rd VLDB conference*, pages 226-235, 1997.

Wang, James Z., Hua, Kien A., and Young, Honesty C. (1996). SEP: a space efficient pipelining technique for managing disk buffers in multimedia servers. *In Proc. Of the IEEE int'l Conf. On Multimedia Computing and Systems*, pages 598-607, Hiroshima, Japan, June 1996.

Wang, James Z. and Hua, Kien A. (1997A). A bandwidth management technique for hierarchical storage in large-scale multimedia servers. *In Proc. Of the International Conference on Multimedia Computing and Systems'97*, pages 261-268, Ottawa, Ontario, Canada, June 1997.

Wang, Y. and Du, D. (1997B). Weighted striping in multimedia servers. *In Proc. Of the International Conference on Multimedia Computing and Systems'97*, pages 102-109, Ottawa, Ontario, Canada, June 1997.