

Video Placement, Distributed Load balancing and Buffer Management Policy for Video-on-Demand (VoD) Systems

Sudhir N. Dhage

Department of Computer Engineering
VJTI, Matunga, Mumbai-400 019.

B.B.Meshram

Department of Computer Engineering
VJTI, Matunga, Mumbai-400 019.

ABSTRACT

As computer and network technology advance, multimedia data can be transferred in real time on the internet. The increasing user demands for various multimedia data make VoD (video on demand) services to be developed. Video on demand (VoD) services are being used in lots of fields such as entertainment, distant learning, home shopping, and interactive news. Video-on-Demand (VoD) has been gaining increasing popularity in recent years. The computational problems arising in such systems are very large and require appropriate mechanisms for distributing the data among the processors. The goal is to decide the rank of the videos, the no. of copies and place them on the proxy servers so that the no. of hits is maximized.

In this paper, we focus on various architecture and scheme for video placement, Distributed load balancing and buffer management for video on demand system. We then proposed the Architecture and Mathematical Model for Video placement, Distributed Load Balancing and Buffer Management Scheme for Video-on-demand systems. We then proposed algorithm for calculates the rank of the videos in a novel way. It is very effective as it also calculates the access difference using access rate of 48hr and 24hr. Also the expected bandwidth of a video is changed in every 24 hr. The algorithm also considers size and bandwidth of a video while calculating its rank. This gives optimum solution for short and mostly viewed videos; finally, we proposed the algorithm for Distributed load balancing and Buffer Management.

General Terms

Algorithms, Computer Networks and Communication, Video on demand Systems, Distributed Load Balancing.

Keywords

Video Placement, Distributed Load Balancing, Buffer Management, Video-on-Demand, NP Hard.

1. INTRODUCTION

Thousands of videos are placed at a VoD Server. As the I/O bandwidth of storage devices like disks or arrays of disks is limited, multiple copies of the same videos are placed on different storage devices. This leads to two problems, first problem is the distribution of multiple copies of videos on different disks and the second problem is to forward a request to a particular disk so the dynamic load of the disks is minimized. The task of allocating the multiple video copies on different disks is currently handled by an experienced operator. This problem becomes very complex, cumbersome and tedious as the number of videos as well as VoD Servers increases. The number of copies of each video is calculated as

an apportionment problem [18]. The proposed video placement algorithm allocates a copy of the video on a particular disk using a greedy approach as follows. The video copies of each video starting from the most accessed video to the least accessed video are allocated on disks at each stage so that the static I/O bandwidth load of all disks is minimal. The second problem (called disk load balancing) forwards the incoming client's request to a particular disk so that the I/O bandwidth loads among the disks is balanced. The proposed algorithm forwards a request to a disk in such a way that the variance of I/O bandwidths of all disks after the allocation of the disk is minimal.

The buffer management scheme is used in a VoD system to overcome the disk bandwidth limitation resulting in the increase of effective bandwidth of the disks. Recently, various interval-based buffer management schemes have been proposed to take advantage of the sequential access pattern of video streams, e.g., interval caching (IC), greedy interval caching (GIC) and Greedy-Interval-Strategy (GIS) [22, 23, 24, 26]. The interval-based buffer management scheme uses the temporal locality of the consecutive requests for the same video. The amount of memory required by the interval-based buffer management scheme is defined to be the number of data blocks that need to be stored in the main memory for streaming the video. The concept of the interval-based buffer management scheme is to use the data blocks retrieved by the preceding I/O stream which are kept in the memory for the reuse of the next I/O stream of the same video. The main problem with the interval-based buffer management scheme is that the amount of memory required is large, if the inter-arrival time of the requests is large and not uniform. The rest of the paper is arranged as follows. In section 2, Related Work is presented, In section 3, Design of VoD System is given, In section 4, Proposed Architecture and Mathematical Model is given In section 5, Proposed video Ranking algorithm, Distributed load Balancing Algorithm and Buffer Management algorithm are given. In section 6, System Implementation is given. In section 7, system performance is given. Finally, some concluding remarks are given in section 8.

2. RELATED WORK

The video-on-demand system is broadly categorized as near VoD or true (inter-active) VoD. The near VoD system broadcasts its videos mainly at a set time, and often through multiple channels at multiple times, and a viewer can choose whether or not to view a video. In the case of the true VoD system, the users can select any video from a list of those available and can perform VCR-like functions with the selected video at any instance of time. Generally true VoD is known as a video-on-demand system. A video-on-demand

allocates a single channel to each user and they can perform VCR-like interactive functions remotely. The VCR-like functions includes play, pause, stop, rewind, fast-forward etc. In the design of video-on-demand System, one must consider various parameters like architecture of the VoD system, size of video library, placement of replica servers (proxy servers), replication of videos, rate of content ingestion, content delivery and request routing [1]. Replication of videos at the proxy server (called video placement) is an important issue because their popularity changes over time. The video placement needs to be done either online or offline as all videos cannot be placed at a VoD Proxy Server. The allocation of videos to the disks at the time of system initialization is called offline video placement whereas the allocation of videos to the disks at the time of streaming is called online video placement [20]. We will refer to offline video placement as simply video placement. The request routing mechanism is used if the number of VoD servers is more than one at a proxy server. The main function of a request routing server is to forward the request to one of the VoD servers where the video is stored. The request forwarding mechanism can be done with the help of either a query-based scheme or a digest-based scheme. In the case of a query-based mechanism, the request routing server broadcasts the request to all VoD servers and a server having the respective video responds to the client. In the later mechanism, the request routing server keeps the list of videos and where each is stored, so that the request is forwarded as soon as the request comes from the client.

The Content delivery is the process of delivering a video from the proxy server to clients. The issues concerning content delivery are how to provide the quality of video streaming and buffer management at the server and the client side. There are generally two methods of delivering video viz. client-pull content delivery and server-push content delivery. In the case of client-pull content delivery, the client program requests a segment of the video continuously, so the client must know how the video is stored at the proxy. While in the case of server-push content delivery, the video is delivered to the client while it does not interrupt the delivery of content [1]. The storage subsystem of a VoD system is important, since the I/O bandwidth of a disk is low compared to other parts of the system. Due to the limited I/O bandwidth of disks, the data placement problem has been solved in different ways e.g. RAID with time or space stripping [4, 5, 6]. The design of the Stony Brook video server with double buffer management for video transmission has been discussed in [9, 10, 11, 12]. In the literature, two types of system architectures are mentioned for the VoD system viz. centralized and proxy-based architecture as shown in Fig. 1 [1]. The former architecture of the VoD system has a content server and the users' requests are sent to it. The later architecture of the VoD system consists of a central content server and various proxy servers (replica servers). The central content server stores all the videos of the VoD system. The proxy servers are used to store the most frequently used videos by a set of users so that load of the central content server is minimized. All clients' requests are sent to the proxy server. If the video is not available at the proxy server, it is downloaded from a central content server. The centralized architecture has some serious drawbacks as compared to a proxy-based architecture.

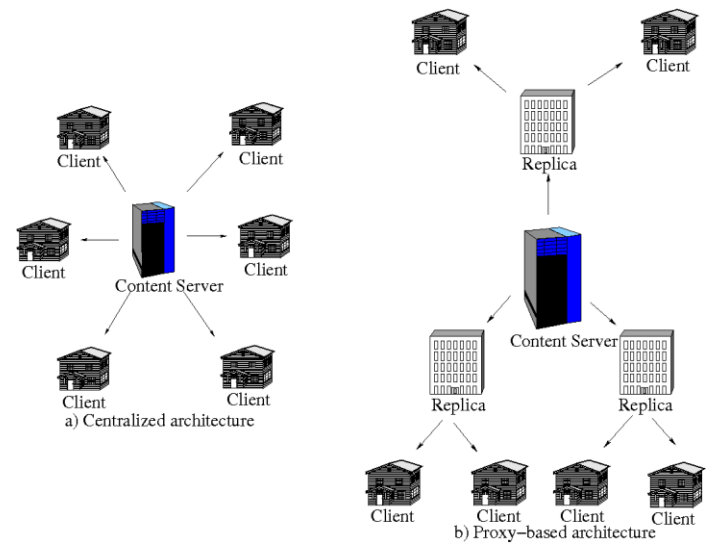


Fig. 1: Types of VoD System Architectures

The first drawback is the single point of failure, if the central content server goes down. The second drawback is the very high load on the central content server as well as the links.

2.1 Video Placement and Disk Load Balancing Algorithms

The graph-theoretic clique based CLLF video placement algorithm first allocates the videos with multiple copies using clique trees and then single copy videos are assigned using least loaded first scheme [10, 11]. The DASD disk load balancing algorithm allocates a play request to a disk with minimum load. If the performance of the system degrades below the predefined threshold, the DASD dancing algorithm which is also graph-theoretic is initiated and has the effect of shifting load from relatively overloaded to relatively under loaded disks [10, 11]. The video placement algorithm using genetic algorithm (GA) is proposed in [16, 17].

2.2 Buffer Management Schemes

The storage subsystem of the VoD system is important, since the I/O bandwidth of the disk is low as compared to other parts of the system. Due the limited I/O bandwidth of a disk, various buffer management schemes have been proposed in the literature to increase the effective bandwidth of the disks. The design of Stony-Brook video server with double buffer management for the transmission has been discussed in [9, 10, 11, 12]. Two classes of buffer management schemes are used in the literature for a VoD system viz. block-based and interval-based. Several block-based buffer management schemes have been proposed in the literature viz. LRU, LRU-k, BASIC, L/MRP, Q-L/MRP AND MPEG-L/MRP [24, 26]. All the variation of LRU-like buffer management schemes are easy to implement but not efficient due the large size of video files and sequential access pattern. While other block based buffer management schemes are too CPU-intensive and might not be appropriate for a VoD system [31]. While interval-based algorithms are heavily used in the VoD system. Interval Caching(IC), DISTANT, Dynamic Grouping (DG), Greedy Interval Caching (GIC) and Greedy-Interval-Strategy (GIS) are the examples of interval-based schemes [23]. The dynamic grouping buffer management scheme merges and decomposes in two groups with largest profit [28]. The greedy

interval caching is an extension to the IC algorithm to take care of both small and large video objects. All the above interval-based caching algorithms select the smallest intervals to be cached. The popularity-aware interval caching considers the reference popularity as well as request intervals [27]. In [29], the performance metric has been proposed to determine which parts of the streams should be cache at any point in time based on the maximum number of simultaneous cache misses.

3. DESIGN OF VOD SYSTEM

The very first step in the design is to decide on the architecture of a VoD system depending on the number of users, growth rate of users and growth rate of media ingestion. If the growth rate of users as well as video is high, the VoD system should be scalable. Fig. 2 shows the design steps of the VoD system which has been modified according to the problem definition. The original design steps are given in [1].

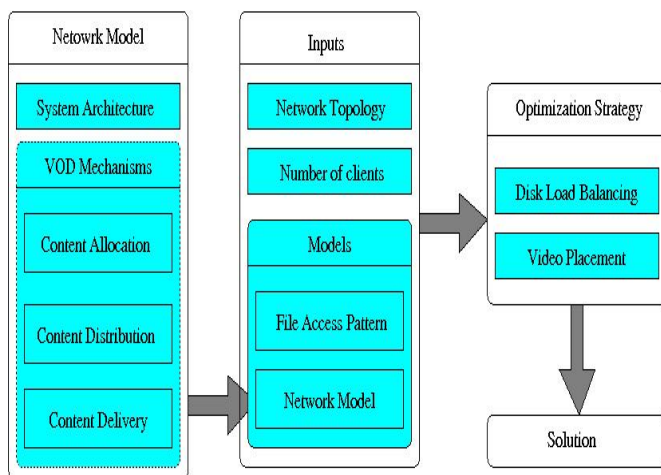


Fig. 2: Design of Video-on-Demand system

As shown in Fig. 2 shows, three components of the VoD system design are various models, inputs parameters and optimization techniques. The network model consists of the architecture of the VoD system and various VoD mechanism e.g. content allocation, content distribution and content delivery etc. File Access pattern, the growth rate of users as well as videos are used to decide on the size of the VoD system. Network topology, number of clients(C) and various models are inputs to the optimization techniques. A buffer management scheme, disk load balancing algorithm and video placement algorithm are the various optimization techniques. The video file format used for the VoD system is also important decision as the file format uses different compression technique. The mpeg format has been used in the research work due to open standard and support for the heterogeneous devices. The network topology used in the VoD system for the Research Work is star as shown in Fig. 3. The file access pattern used in the VoD system is zipf's distribution.

4. PROPOSED ARCHITECTURE AND MATHEMATICAL MODEL

Our proposed system aims at combining the features of proxy based and content distribution architecture and is distributed in nature. Our system will provide high availability while supporting a high number of video streams. It will eliminate the drawbacks of the existing system.

The basic components of the system would be

1. Central Server/Main Server
2. Proxy Servers
3. Clients

Main Server: The main or central server is the main multimedia server where all multimedia contents are available. It is the main video library from where videos are placed on the proxy server.

Proxy Server: Proxy servers are a subset of the central server. A set of clients are connected to each proxy server i.e. client requests are served by the proxy server.

Client: The clients are the users of the VoD system. They request to proxy server for any multimedia content they wish to view.

The above steps clearly represent the modular break down of the entire setup required for the proposed system.

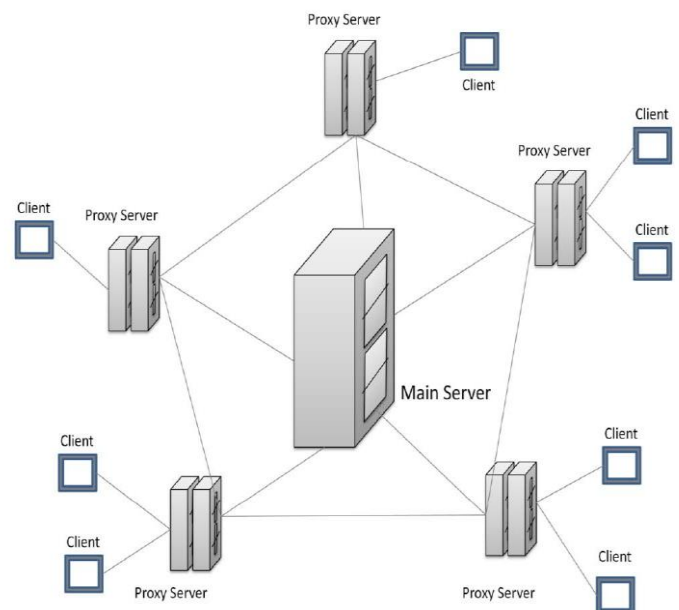


Fig. 3: Proposed Basic Layout of VoD System

The proposed System Architecture of Video-on demand System is given in fig. 4, which consists of multimedia server which contains all types of videos and proxy servers which is subset of multimedia sever. The Videos added/deleted according to its rank calculated using its online popularity.

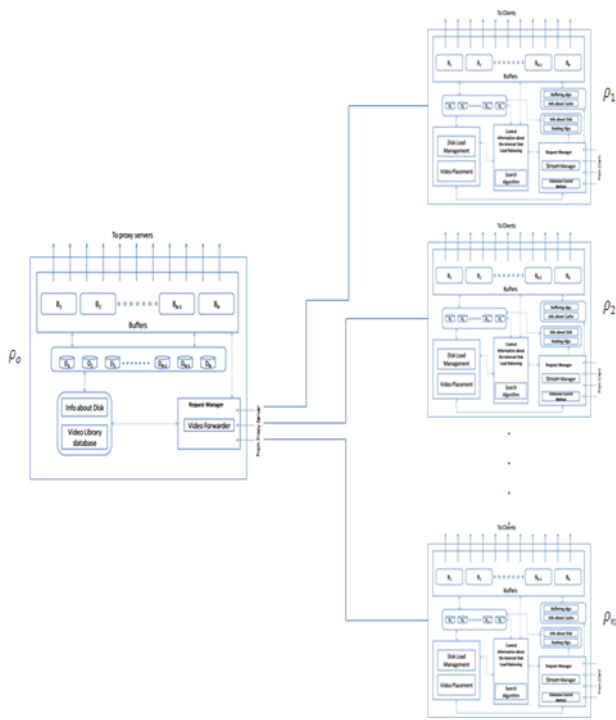


Fig. 4: Proposed System Architecture of VoD System

4.1. Mathematical Model

Table I shows the summary of the parameters and their definitions.

N^i	No. of videos
N_j^i	No of videos on disk j
n^i	No of disks
DB_j^i	Bandwidth capacity of disk j
DS_j^i	Size of disk j
B_j	Bandwidth of video j
S_j	Size of video j
R_j^i	Rank of video j
A_j^i	Access Request of video j
S_{avg}^i	Avg size of all videos
B_{avg}^i	Avg bandwidth of all videos
α_j^i	No of copies of video j
AVB_j^i	Actual B/W of video j

EVB_j^i	Expected B/W of video j
β_j^i	BSR of disk j (Original)
$A\beta_j^i$	Actual BSR of disk j
$E\beta_j^i$	Expected BSR of disk j
$T\beta_j^i$	To-be BSR of disk j
ρ_j^i	Avg Difference in BSR
$A\delta_j^i$	Actual BSR deviation of disk j
$T\delta_j^i$	To-be BSR deviation of disk j
τ_j^i	Change deviation of disk j

Table 1: Parameters and Definitions

4.2 Data Structures Used

AM (0,1) → Assignment Matrix

AM_{ij}	u_1	u_2	u_n
δ_1^i	1	0	0
δ_2^i	0	1	0
δ_n^i	0	1	1

Table 2: Assignment Matrix

AVB → Actual Video Bandwidth

AVB_j^i
AVB_1^i
AVB_2^i
:
:
AVB_n^i

Table 3: Actual Video bandwidth

EVB → Expected Video Bandwidth

EVB_j^i
EVB_1^i
EVB_2^i
:
:
EVB_n^i

Table 4 Expected Video Bandwidth

5. ISSUES IN VIDEO PLACEMENT

The following are the issues that should be addressed by the placement policy:

- i. The expected load on a video object can be projected by external agents on the basis of historical data together with human input. Discounting price, review may also affect the expected load. Such unpredictability can be better handled by the use of an external agent that monitors the demand for the video objects in the system and initiates on-line changes in placement of video objects when needed.
- ii. Placement policy should also consider quickness with which changes in placement have to be placed.

The expected load on a replica is maintained by the system only for the purpose of placement. The current load, in fact, interferes with the operation of the placement policy since it prevents creation of replicas on a device that is currently heavily loaded but its expected future load will subside. A change in expected load on more than one video object can happen at the same time. The Changes to the expected load for multiple video objects can be accomplished by repeatedly executing the same algorithm for different videos in a certain order. First, the BSR policy is invoked for changing the expected load for video objects whose demand has decreased. After this, the change in expected load for other video objects, i.e., for those whose expected load has increased are performed in decreasing order of expected load. Initial placement is done by placing the video objects arbitrarily.

5.1 PROPOSED VIDEO PLACEMENT ALGORITHM

The proposed algorithm consists of two modules:

- i. Offline Video Placement: Videos placed manually depending on the rank of the video.
- ii. Online Video Placement: Videos added/deleted according to its rank calculated using its popularity online.

Assumptions for Video Placement:

1. A video is placed if space is available irrespective of the availability of B/W on the disk.

2. A video is removed if and only if : No. of copies of the new video to be placed is greater than **and** Lack of storage space

5.1.1 Determining Rank of Videos for each Proxy Server

1. Start
2. For each video,
 - 2.1. Calculate

$$A_j^i * 100 / \sum_{j=1} A_j^i$$

- 2.2. Arrange the above values in descending order and assign R_j^i as index of each video starting from 1.

- 2.3. In case of same index check the following condition

$$S_{avg}^i = \sum_{j=1}^{N_j^i} S_j^i / N_j^i$$

$$B_{avg}^i = \sum_{j=1}^{N_j^i} B_j^i / N_j^i$$

$$B_j^i * S_j^i * 100 / (S_{avg}^i * B_{avg}^i)$$

- 2.4. Arrange the index in ascending order to remove conflict of same rank.

3. End.

5.1.2 Addition of Video in specified Proxy Server

1. Start
2. In proxy server, for each disk, Calculate

1. $\beta_j^i = DB_j^i / DS_j^i$

2. $A\beta_j^i = \sum_{j=1}^{N_j^i} AVB_j^i / \sum_{j=1}^{N_j^i} S_j^i$

3. $E\beta_j^i = \sum_{j=1}^{N_j^i} EVB_j^i / \sum_{j=1}^{N_j^i} S_j^i$

4. $\rho_j^i = (\sum_{j=1}^{N_j^i} (EVB_j^i - AVB_j^i)) / N_j^i$

5. $T\beta_j^i = (\sum_{j=1}^{N_j^i} AVB_j^i - \rho_j^i + EVB_m^i) / (\sum_{j=1}^{N_j^i} S_j^i + S_m)$

6. $A\delta_j^i = |A\beta_j^i - \beta_j^i|$

7. $T\delta_j^i = |T\beta_j^i - \beta_j^i|$

8. $\tau_j^i = T\delta_j^i - A\delta_j^i$

- 2.1. Select the disk with minimum value of τ_j^i for placing the new video.

3. End.

5.1.3 Dynamic Video Placement Algorithm in a Proxy Server

1. Start
2. For each video j ,
 - 2.1. If $EVB_j^i - AVB_j^i < 0$ then,
 - 2.1.1. If $R_j^i < 30\%$ of total ranks then Increase no of copies of video j by 1 using BSR
 - 2.2. Else
 - 2.2.1. If $R_j^i > 30\%$ of total ranks then $EVB_j^i - AVB_j^i = (EVB_j^i - AVB_j^i) / 2$
3. End.

5.1.4 Determining the number of copies of Video in a Proxy Server

- 1) First we set the ranges depending upon the no. of disks on the proxy server.
- 2) Now according to the percentage of the total access requests for a video, we place it in an appropriate range.
- 3) The range in which a video is placed decides the no. of copies of that video.
- 4) The videos in the range with maximum range boundary values will be placed on all the disks.
- 5) The no. of copies decreases by 1 with the next maximum range boundary values.
- 6) Thus the no. of copies in the range with lowest range boundary values will be 1.

5.2 Distributed Load Balancing Algorithm for VoD Servers

1. Start
2. Accept video V_l through Admission Control for proxy server ρ_i
 - 2.1. If video is found on the native proxy server ρ_i run the Load Balancing Algorithm else $\psi_i := 1$
3. Check the links with the neighboring proxy servers ρ_{i-1} and ρ_{i+1} for sufficiency of bandwidth required by the video V_l
 - 3.1. If $\lambda_{i-1} > \nu_l$ send video request to ρ_{i-1} , run the Request Response Algorithm and wait for its normalized utilization detail ψ_{i-1} else $\psi_{i-1} = 1$
 - 3.2. If $\lambda_{i+1} > \nu_l$ send video request to ρ_{i+1} , run the Request Response Algorithm and wait for its normalized utilization detail ψ_{i+1} else $\psi_{i+1} = 1$
4. If $\psi_i = 1$ then continue, else calculate the normalized utilization detail ψ_i for native proxy server ρ_i and compare with ψ_{i-1} and ψ_{i+1}
 - 4.1. If $\psi_i = \psi_{i-1} = \psi_{i+1} = 1$ then

- 4.1.1. Forward request to Main Server ρ_0 to upload video V_l to ρ_i
- 4.1.2. Run Load Balancing Algorithm
- 4.2. Else If ψ_i is minimum then send a negative acknowledgement to both ρ_{i-1} and ρ_{i+1}
- 4.3. Else If ψ_{i-1} is minimum then
 - 4.3.1. Send acknowledgement to ρ_{i-1} to start streaming through ρ_i
 - 4.3.2. $\lambda_{i-1} := \lambda_{i-1} - \nu_l$
 - 4.3.3. Send a negative acknowledgement to ρ_{i+1}
- 4.4. Else If ψ_{i+1} is minimum then
 - 4.4.1. Send acknowledgement to ρ_{i+1} to start streaming through ρ_i
 - 4.4.2. $\lambda_{i+1} := \lambda_{i+1} - \nu_l$
 - 4.4.3. Send a negative acknowledgement to ρ_{i-1}
5. After the video stream is complete
 - 5.1. if video is being streamed from the native proxy server ρ_i then
 - 5.1.1. $\sigma_\alpha^i := \sigma_\alpha^i + \nu_l$
 - 5.1.2. $DBAM_{\alpha i} := DBAM_{\alpha i} - \nu_l$
 - 5.2. Else if video is being streamed from proxy server ρ_{i-1}
 - 5.2.1. Send request to ρ_{i-1} to stop the streaming
 - 5.2.2. $\lambda_{i-1} := \lambda_{i-1} + \nu_l$
 - 5.3. Else if video is being streamed from proxy server ρ_{i+1}
 - 5.3.1. Send request to ρ_{i+1} to stop the streaming
 - 5.3.2. $\lambda_{i+1} := \lambda_{i+1} + \nu_l$
6. Stop

5.3 Algorithm Buffer Management Interval Caching

- ```
// S1, S2 arriving streams
// t1, t2 time at which they arrive
```
1. Start
  2. If no preceding stream then S1 gets data from disk;
  3. If next stream S2 arrives at  $t_2$  then interval =  $t_2 - t_1$ ;
  4. If space available in cache then cache interval;
  5. Else if cache is full then
  6. For each( interval  $j$  in cache)
  7. If ( $j > \text{interval}$ ) then replace  $j$  with interval;
  8. Else Serve from disk;
  9. End

## 6. SYSTEM IMPLEMENTATION

Main Server contains database of all the videos at any given point of time. Used when none of the required proxy servers are able to serve a video to the user. Also all the videos placed on any proxy server during initial placement phase is taken from the main server. Thus more and more proxy servers can be added later on at any point of time without disturbing the system configuration. Thus, the system becomes distributed.

### 6.1 System Implementation at Main Server

The main server contains files of all the videos available to the entire system. All the files are stored in the hard disk of the main server. Thus videos can added later on as and when required. Again, this property satisfies the distributed nature of the system. The module that runs on the main server is a C program that accepts connection request from a proxy server and transfers required video files to the proxy server. Moreover, the module is able to accept multiple proxy server requests simultaneously at one time. Also, the data transfer takes place at a high speed since we have used a modified FTP technique to deliver greater performance.

In the implementation of this module we have used a various concepts such as:

1. Socket programming –The most important and core of our file socket programming which deals with main server–proxy server connections and communications.
2. Multiple proxy server support –The main server is capable of supporting multiple proxy server requests at a time and catering to them successfully.
3. Modified FTP technique –The default buffer size is increased and the number of data packets streamed at one time is increased to provide high transfer speed.

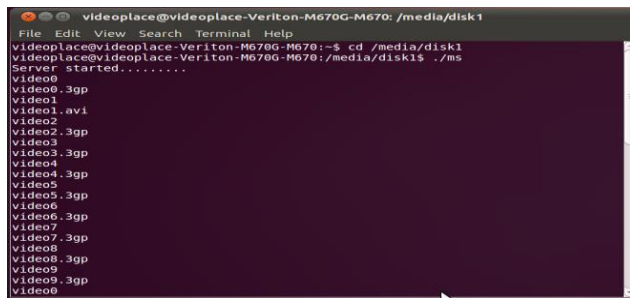


Fig. 5: Video Placement in Proxy Server

### 6.2 System Implementation at Proxy Server

Proxy Server contains files of popular videos on its storage according to its own calculation of accesses to the videos and also the metadata of each video in the system. It also has its own ranking database and access database for videos placed on it. The functions are performed by the proxy server are:

- Initial placement of videos on its storage in a normalized way using Modified BSR Algorithm.
- Providing access to video files to multiple users at the same time.
- Updating the shared variable such as video bandwidth matrix, assignment matrix, AVB matrix, access matrix at run-time.
- Dynamically calculating the rank of videos according to popularity.
- Dynamic addition and removal of videos files according to the Modified BSR Algorithm.

- Increasing and decreasing the number of copies for a video according to its run-time popularity.
- Maintaining metadata files for each video as well as for the primary memory.
- Creating log files for maintenance and error detection in the proxy server.

### 6.3 Video Placement Module

This module is responsible for initial and dynamic placement of videos. It implements the Modified BSR Algorithm, the initial placement technique, the dynamic placement policy, rank calculations, storage of metadata and creation of log files. Thus, acts as the main engine for video placement in the proxy server. There are four programs which run in order to complete the above tasks.

**Videoplace.c** –This program is backbone of the video placement module. It includes two c files used as header files which have different function declared and defined in them for video placement according to the Modified BSR Algorithm. It also defines all the shared memory variables which are used by all the modules in the video placement environment as well as the dynamic load balancing environment. Thus, this is the module which is run first while setting up the system for the VoD server.

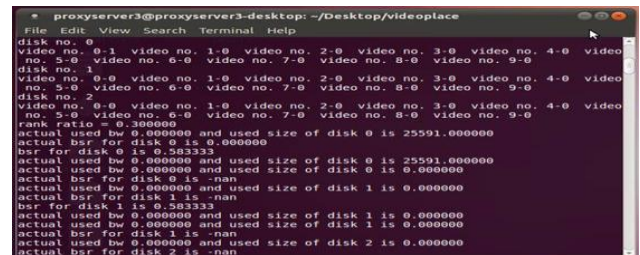
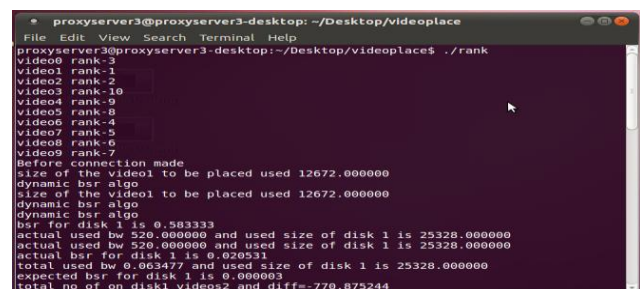
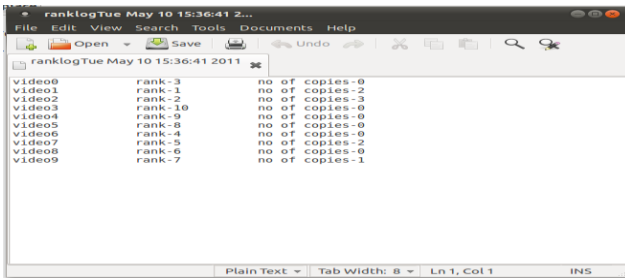


Fig 6: Dynamic Video Placement

**Rankcal.c**–This program is run independently using a Scheduler. It is used for calculation of rank of videos after a specified amount of time. The rankcal.c file also includes some shared memory variables like number of copies matrix, access different matrix, video bandwidth matrix, total access matrix to calculate the rank of a video according to the formula defined in the algorithm. It then calculates the number of copies of each video that should be placed in the proxy server and starts their placement accordingly using Modified BSR Algorithm. Also, the rankcal.c program created a rank log file with timestamp after its completion for maintenance and analysis purposes.



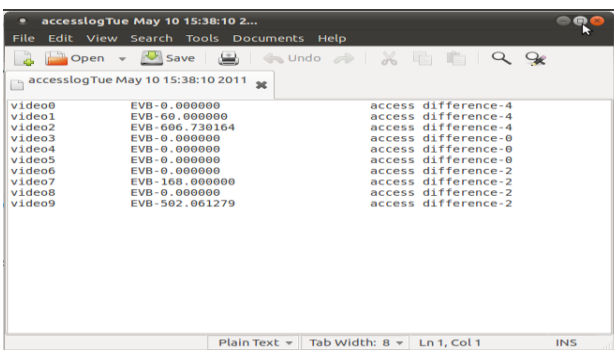
**Fig 7: Video Ranks Calculation**



| Video ID | Rank    | no of copies   |
|----------|---------|----------------|
| video0   | rank-3  | no of copies-0 |
| video1   | rank-1  | no of copies-2 |
| video2   | rank-2  | no of copies-3 |
| video3   | rank-10 | no of copies-0 |
| video4   | rank-9  | no of copies-0 |
| video5   | rank-8  | no of copies-0 |
| video6   | rank-4  | no of copies-0 |
| video7   | rank-5  | no of copies-2 |
| video8   | rank-6  | no of copies-0 |
| video9   | rank-7  | no of copies-1 |

**Fig 8: Placement of copies as per ranks**

**Access.c** –This is a very small program but important nevertheless. It keeps track of the accesses to each video and then according changes the values of the EVB matrix which is included in the shared memory variables. The correction in the EVB matrix values plays an important role in keeping the system optimized. Finally, the access.c file creates an access log file with timestamp for maintenance and analyzing results.



| Video ID | EVB            | access difference   |
|----------|----------------|---------------------|
| video0   | EVB-0.000000   | access difference-4 |
| video1   | EVB-60.000000  | access difference-4 |
| video2   | EVB-606.730164 | access difference-4 |
| video3   | EVB-0.000000   | access difference-0 |
| video4   | EVB-0.000000   | access difference-0 |
| video5   | EVB-0.000000   | access difference-0 |
| video6   | EVB-0.000000   | access difference-2 |
| video7   | EVB-168.000000 | access difference-2 |
| video8   | EVB-0.000000   | access difference-2 |
| video9   | EVB-502.061279 | access difference-2 |

**Fig 9: Access different of Video file**

**Different.c** –This program is used for recording the changes in the access of a video file for the last 48 hours, last 24 hours and the current access values. In short its acts as a database manager for access variables of a video.

## 7. SYSTEM PERFORMANCE

For optimizing the system performance, two objectives are defined:

1. *Minimizing the total capacity usage of storage disk arrays* –taken care by the Ranking Policy and Modified BSR Algorithm as proposed by our system.
2. *Optimizing the load balancing amongst storage disk arrays* –taken care by the Load Balancing Techniques.

The placement problem is also subjected to a number of constraints like server capacity constraints, constraints of bandwidth and reliability.

1. *Server Capacity* - The capacity of each disk array must not be exceeded.
2. *I/O Bandwidth* - The bandwidth of each disk array must not be excluded.
3. *Reliability* -In order to secure the accessibility of the user, each video is stored at least in two disk arrays.

Genetic Algorithm can be used to solve the above defined Objectives keeping the Constraints in place.

There are a number of advantages in using Genetic Algorithm for Video Placement:

1. It can handle problem with constraints.
2. It has the distinct advantage of solving multi-objective problems.
3. It processes an intrinsic parallelism architecture in which no extra effort is required to construct a parallel computational framework. Such property is essential for the achievement of fast system update.
4. It can solve multi-modal, non-differentiable, non-continuous or even NP-complete problems.

However, a major disadvantage of using Genetic Algorithm for video placement is that it requires extensive calculations to be performed which heavily load the processor speed and thus slows down the overall system which is unacceptable. Hence, the use of Genetic Algorithm for video placement is avoided in real-time systems where speed is critical for success.

The design of an optimal file placement is considered as being a multi-objective, highly constrained, and non-continuous problem. Although different heuristic approaches have been proposed, they are focused on a system with multiple servers in a VoD network, assuming a single disk per server. A popularity-based approach is developed for multiple disks based on the information of popularity. The algorithm is designed for a homogeneous system with identical disks, and assuming that all multimedia files have the same duration. A similar analysis of the storage cost for distributing movies across disks is available. However, different types of disks may coexist in the video server and the durations of the multimedia files usually vary from a few minutes for short video clips to more than two hours for movies.

An optimal condition of load sharing for minimum blocking probability is constructed numerically in such a heterogeneous system while the solution is obtained effectively by an iterative approach. A genetic algorithm, together with a heuristic bin-packing algorithm is designed such that the optimal number of copies of the multimedia files and their corresponding disk locations are determined.

There are two major objectives in order to optimize the system performance and provide a specified service quality:

- i. To minimize the total capacity usage of the server.
- ii. To minimize the blocking probability of the VoD system.

The problem is also subjected to the constraint on the disk capacity. In other words, the capacity of each disk must not be exceeded.

The Genetic Algorithm(GA) together with a modified bin-packing algorithm is known as Highest Load First (HLF). The gene of the chromosome in the GA is to reflect the number of copies of each multimedia file, while the HLF algorithm will search for the optimal file placement scheme of that chromosome such that a minimum blocking probability is found.

The objective is to allocate the elements such that all the bins are “just full”. Similarly to the conventional bin-packing problem, the sequence of the elements to be put into the bin may affect the effectiveness of the algorithm. In general, there are four different reordering approaches:

- a. The descending approach arranges the elements from largest size to the smallest.
- b. The ascending approach arranges the elements from



smallest size to the largest.

- c. The random approach arranges the elements randomly.
- d. The regular approach maximizes the distances between similarly sized elements.

Due to the discrete effect of the element's size, it is expected that the descending approach will be more effective as the element with larger size is allocated first. However, a major disadvantage of using Genetic Algorithm for file placement is that it requires extensive calculations to be performed which heavily load the processor speed and thus slows down the overall system which is unacceptable. Instead, our proposed algorithm takes into account Static as well as Dynamic Popularity of the videos to decide their placement. Also, we consider The Modified BSR Algorithm to minimize the objective of blocking probability of the VoD system. There are two major problems involved in VoD systems. The first problem is the manual video allocation of multiple copies of videos to the disks, called video placement. The second problem is where to forward a newly arrived request to play a video so that the dynamic load amongst disks or arrays of disks is balanced. Formulation of the video placement as an INLP optimization problem and the video placement problem to distribute the number of copies of each video to disks or arrays of disks is NP-hard. Greedy video placement and disk load balancing algorithms can be used to minimize the static and dynamic loads of disks respectively.

There are two important issues in the design of VoD system. The first is a high network bandwidth requirement for the streaming of a video and the second is a high I/O bandwidth to pull the data from a disk. The former requirement of high network bandwidth is solved by using new telecommunication technologies e.g. ADSL, VDSL and compression methods. The later requirement of I/O bandwidth is solved by the various ways such as allocating the multiple copies of some or all of the videos called video placement (VP), disk load balancing algorithm (DLB) and a buffer management scheme to increase the effective bandwidth of the disks.

The allocation of the videos copies on the disks is called video placement problem. The existing algorithms for the video placement include the video placement using genetic algorithm (GA) and graph-theoretic based clique approach. The solution using genetic algorithm is generated either for the specific threshold objective or the number of iterations. The solution of the video placement algorithm using genetic algorithm and clique is not always optimum. Hence, we have proposed a method that takes into account all the constraints faced by VoD systems and achieves all the objectives necessary without overloading the servers with excessive calculations.

In a video-on-demand server, resource reservation is required to guarantee continuous delivery. Hence any given storage device (or a striping group treated as a single logical device) can serve only up to a fixed number of client access streams. Each storage device was also limited by the number of video files it can store. One or more copies of a particular video may be placed on each disk on the server. Access rates of different videos are not uniform so a dynamic placement policy that creates/deletes replica of a video and mixes hot and cold videos is necessary.

We consider a fast or a slow device as a device with relatively larger bandwidth or space, respectively. A popular video object may be viewed by many clients simultaneously, and hence, multiple replicas of that video may need to be

created on multiple storage devices to support a very high bandwidth requirement. Large and small size videos are considered same. Most frequently watched videos are considered as hot videos whereas less frequently watched videos as cold videos. Both the type of videos is to be mixed for better results. Each video is characterized by their bandwidth and space.

In order to reduce the storage costs of a multimedia system, videos may be stored on tape and staged into disk as necessary. Such systems require placement policy that specifies which video on disk to replace if there is no free space. For example, a heat-based policy that replaces segments of relatively unpopular videos with segments of currently referenced videos as needed. This policy exploits the fact that different videos may have different expected demands. However, staging videos in from tape incurs various overheads and may not be economical for all videos. First, while tape storage reduces the cost of storing videos, it increases the cost of playback since the video has to be read in from tape to disk and then played. For playing large objects expensive tape device need to be occupied for a long time. Even for short video objects, the overhead of mounting the tape and advancing the tape to the correct position may be very high compared to the time required to actually retrieve the object. In addition, the queuing delay for accessing the tape may be significant if enough tape storage are not available or if load imbalance occurs across tape devices. Therefore, if the system cost is dominated by the cost of the resources needed to play back streams, it may be cost-effective to pre-stage popular videos to disks.

A dynamic approach that uses partial replication of video files (referred to as the Dynamic Segment Replication (DSR) policy) for load-balancing in multimedia systems. It is based on the observation that if there were a number of consecutive requests for the same video and the blocks read in by the first request are copied to another device, it would be possible to switch the following requests to the partial replica just created. Such an approach should be viewed as a complementary runtime policy. The DSR policy creates a replica of a hot video only if enough space is available on a device with available bandwidth. In other words, a good initial placement based on BSR matching is required for the DSR policy to handle dynamic load fluctuations. In a real system, some space and bandwidth may be set aside for the purpose of load balancing.

## **8. CONCLUSION**

In this research proposed system architecture for VoD systems is presented in a novel way by doing the survey of existing VoD systems. The proposed algorithm calculates the rank of the videos in a novel way. It is very effective as it also calculates the access difference using access rate of 48hr and 24hr. Also the expected bandwidth of a video is changed in every 24 hr. The algorithm also considers size and bandwidth of a video while calculating its rank. This gives a higher rank for short and mostly viewed videos, The online placement considers the BSR policy as well as Zipf's law which makes it unique and effective. The algorithm is run in every 24hr so all the values are updated regularly. This makes the system to give maximum number of hits. Thus the proposed algorithm will be a benchmark in video placement's history. Finally Proposed the optimum distributed load balancing and buffer Management algorithms for getting faster access to multiple clients on video on demand(VoD) systems.

## 9. REFERENCES

- [1] Thouin F. and Coates M., "Video-on-Demand Networks: Design Approaches and Futures challenge", Pg. 42-48, Network, Proceedings IEEE, April 2007.
- [2] Thouin F., Coates M., and Goodwill D., "Video-on-Demand Equipment Allocation", Proceedings IEEE Network Computing and Applications (IEEE NCA), Boston, MA, July 2006.
- [3] Wang B., Sen S, Adler M and Towsley D, "Optimal proxy cache allocation for efficient streaming media distribution", Multimedia IEEE, Pg. 366-374, April 2004.
- [4] Kim K. and Park S., "Data Placement and Retrieval Policy for Serving Video Operations through Sequential Access in Video Server", Proceedings of the 5th International Computer Science Conference on Internet Applications, Pg. 373-378, 1999
- [5] Cheng C. Lee M. and Oyang Y., "Disk and file system design for MPEG-2 video-on-demand servers", Consumer Electronics, IEEE Transactions, Pg. 1220-1228, November 1997
- [6] Tsai C., Edward T., Chu H. and Huang T., "WRR-SCAN: A Rate-Based Real-Time Disk-Scheduling Algorithm", Fourth International Conference on Embedded Software (EMSOFT '04), Pisa, Italy, September 2004.
- [7] Huang Y. and Huang J., "Disk scheduling on multimedia storage servers", Computers, IEEE Transactions Pg.77-82, Jan 2004
- [8] Kim K., Hwang J., Lim S., Cho J. and Park K., "A Real-Time Disk Scheduler for Multimedia Integrated Server Considering the Disk Internal Scheduler", Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03), Pg 124.1, IEEE 2003
- [9] Venkatramani C. and Chiueh T., "Design, Implementation and evaluation of a software-based real-time Ethernet Protocol", Conference on Applications, Technologies, Architectures and Protocols for Computer Communication, ACM SIGCOMM'95, Year-1995
- [10] Chiueh T., Venkatramani C. and Vernick M., "Design and implementation of the Stony Brook Video Server", Software - Practice & Experience, Volume 27, Issues 2, Pg. 139-154, Year - 1997
- [11] Vernick M., "Design and Implementation of the Stony Brook Video Server", PhD Thesis, Computer Science Department, State University of New York at Stony, August 1995
- [12] Venkatramani C., "Design, Implementation and Evaluation of RETHER:A Real-Time Ethernet Protocol", Ph.D. Dissertation, Computer Science Department, State University of New York at Stony, December 1996.
- [13] Niranjana T., Chiueh T. and Schloss G., "Implementation and Evaluation of a Multimedia File System", IEEE International Conference on Multimedia Computing Systems, 1997, Ontario, Canada
- [14] Lee B., Cao P., Fan L., Phillips G. and Shenker S., "Web caching and zipf-like distributions: Evidence and implications", INFOCOM '99, IEEE Proceedings March 1999, New York, USA
- [15] Chakchai S., "A Survey of Proxy Caching Mechanisms for Multimedia Data Streams", Project Thesis, Department of Computer Science, Washington University, May 2005
- [16] Tang K., Chan S and Wong E, "Video Placement in VoD system using genetic algorithm", IT 2000, IEEE Proceedings, Pg.672 - 676, Volume 1,19-22 Jan. 2000
- [17] Tang K., Chan S. and Wong E., "Optimal File Placement in VoD System Using Genetic Algorithm", IEEE Transactions on Industrial Electronics, Vol. 48, No. 5, Oct 2001.
- [18] Wolf J., Philip S. and Shachnai H., "Disk load balancing for video-on-demand systems", Multimedia Systems, Pg.358 - 370, Volume 5, Issue 6,December 1997
- [19] Birk Y., "Deterministic Load-Balancing Schemes for Disk-Based Video-On-Demand Storage Servers", Fourteenth IEEE Symposium on Mass Storage Systems, IEEE 1995
- [20] Dan A. and Dinkar S., "An Online Video Placement Policy based on Bandwidth to Space Ratio",SIGMOD'95, ACM 1995
- [21] Xugang Y., "A Heuristic Method for A Rostering Problem with The Objective of Equal Accumulated Flying Time", MSc thesis, Florida state university college of arts and sciences, 2003.
- [22] Dan A. and Sitaram D., "Buffer Management Policy for an On-Demand Video Server", IBM Research Division, T.J. Watson Research Center York-town Heights, NY 10598,1993.
- [23] Wujuan L., Yong L. and Leong Y., "A Novel Interval Caching Strategy for Video-on-Demand Systems",Pg. 1-5, ICON'06, Networks, 2006.
- [24] Neil E., Neil P. and Weikum G., "The LRU-K page replacement algorithm for database disk buffering",Pg. 297-306 ,ACM SIGMOD,1993.
- [25] Ozden B., Rastogi R. and Silberschatz A., "Buffer replacement algorithms for multimedia storage systems", Pg. 172-180,IEEE, Multimedia Computing and Systems International Conference, 1996.
- [26] Boll S., Heinlein C., Klas W., and Jochen W., "MPEG-L/MRP: Adaptive Streaming of MPEG Videos for Interactive Internet Applications", MIS'2000, International Workshop on Multimedia Information Systems 2000.
- [27] Kim T., Bahn H. and Koh K., "Popularity-aware interval caching for multimedia streaming servers",Pg. 1555 - 1557, ELECTRONICS LETTERS, IEEE 2003.
- [28] Sheu S., Hua K., and Tavanapong W., "Dynamic grouping: An efficient buffer management scheme for video-on-demand servers", Proceedings of the 2nd International Conference on Multimedia Information Systems, 1997.
- [29] Andrews M. and Munagala K., "Online Algorithms for Caching Multimedia Streams", Pg. 64-75, European Symposium on Algorithms, 2000.

- [30] Almeida J., Eager D. and Vernon M., "A Hybrid Caching Strategy for Streaming Media Files", Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking, 2001.
- [31] Cho K., Ryu Y., Won Y. and Koh K., "A Novel Buffer Cache Scheme for Distributed Multimedia Streaming", Pg. 1702-1705, Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, USA, 2003.
- [32] Dan A., Dias D., Mukherjee R., Sitaram D. and Tewari R., "Buffer and Caching in Large-scale Video Servers", Pg. 217-224, Proceedings of IEEE Compcon, 1995.
- [33] Dan A. and Sitaram D., "A Generalized Interval Caching Policy for Mixed Interactive and Long Video Workloads", Pg. 344-351, Proceedings of Multimedia Computing and Networking, 1996.