# Chapter 2

# Literature Review

In this chapter, a review of the current literature on limitations of Video-on-Demand system and possible solutions to overcome them are presented. It includes video placement, disk load balancing and buffer management schemes.

## 2.1 Overview

The video-on-demand(VoD) system is broadly categorized as *near* VoD and *true (interactive)* VoD [1, 2]. 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 include play, pause, stop, rewind, fast-forward etc.

An Interactive Video-on-Demand (IVoD) system retrieves video pro-

grams from its storage and delivers them to the customers through an information network. The customers can select and watch video programs at their convenient time & place and they can interact with the programs via interactive operations. In particular, a network model has been developed to describe the usage of system resources [5, 6, 27]. The two parameters considered are network bandwidth and disk storage capacity. With these resources, users can interact with the VoD system. An IVoD system integrates entertainment and telecommunication facilities and provides services to the geographically distributed users. In this model, from the different videos placed at the front end, user can select any video to view. There are some major components in this model such as users, proxy servers and central media server. When video is selected by the user, a request is transferred from user to the local proxy server. When a request for service is received, local proxy will decide whether this request will be accepted or not. If request is accepted i.e. video is available at the proxy and the request queue is emptied, then resources are allocated to handle the request and reply is sent from the same. The resource allocation includes allocating a video buffer for request and bandwidth in the network to connect the targeted video server, video buffer and users [17, 31].

This chapter investigates various architectures of VoD systems and various existing solutions available for Video Placement, Disk Load Balancing and Buffer Management. The various existing algorithms available for Video Placement, Disk Load Balancing, Buffer management is analyzed. Various methodology of Proxy Server caching is investigated in this chapter.

## 2.2 Design of VoD Systems

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, 2, 10, 11].

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 latter mechanism, the request routing server keeps the list of videos and their location (where each of them is stored), so that the request is forwarded as soon as the request comes from the client.

*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 without interrupting the delivery of content [1, 13, 14].

In the literature, two types of system architectures are mentioned for the VoD system viz. *centralized* and *proxy-based* architecture as shown in Fig. 2.1 [1, 18, 23]. Centralized architecture of the VoD system has a content server and the users' requests are sent to it. Poxy- based 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 [9,

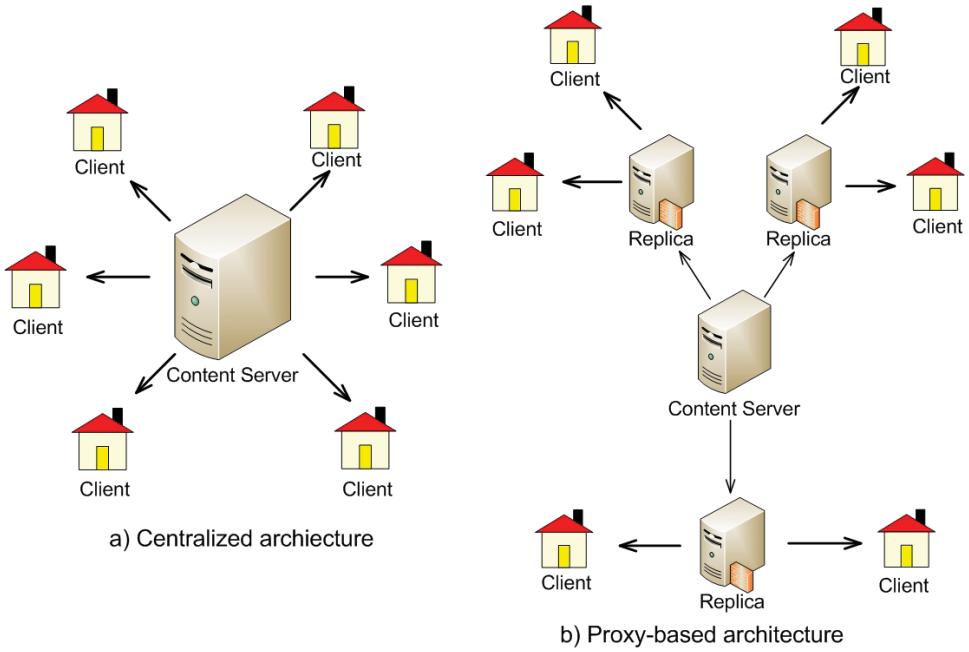


Figure 2.1: Types of VoD system architectures

52, 27]. 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. 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.

## 2.3 Video Placement in Distributed systems

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 [18, 31, 32]. 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* [31, 32, 35]. We will refer to offline video placement as simply video placement.

The number and location of replicas of distinct content objects in a P2P system have a strong impact on the systems performance [28, 30]. Indeed, along with the strategy for handling incoming requests, they determine whether such requests must either be delayed or served from an alternative, more expensive source such as a remote data center. Requests which cannot start service at once can either be enqueued or redirected.

Previous investigations of content placement for VoD systems were conducted by Suh, et al. [46]. The problem tackled in [46] differs from our current perspective, in particular no optimization of placement with respect to content popularity was attempted in that paper. Performance analysis of both queuing and loss models are considered.

Tewari and Kleinrock [47,48] advocated to tune the number of replicas in proportion to the request rate of the corresponding content, based on a simple queueing formula, for a waiting model and also from the standpoint of the load on network links. They further established via simulations that Least Recently Used (LRU) storage management policies at peers efficiently emulated their proposed allocation.

Wu et al.[49] considered a loss model, and a specific time slotted mode of operation whereby requests are submitted to randomly selected peers, who accommodate a randomly selected request. They showed that in this setup the optimal cache update strategy can be expressed as a dynamic program. Through experiments, they established that simple mechanisms such as LRU or Least Frequently Used (LFU) perform close to the optimal strategy they had previously characterized.

Kangasharju et al.[50] addressed file replication in an environment where peers are intermittently available, with the aim of maximizing the probabil-

ity of a requested file being present at an available peer. This differs from our present focus in that the bandwidth limitation of peers is not taken into account, while the emphasis is on their intermittent presence. They established optimality of content replication in proportion to the logarithm of its popularity, and identified simple heuristics approaching this.

Boufkhad et al.[51] considered P2P VoD from yet another viewpoint, looking at the number of contents that can be simultaneously served by a collection of peers. Content placement problem has also been addressed from the stand point of other different optimization objectives. For example, Almeida et al.[123] aim at minimizing total delivery cost in the network and Zhou et al.[52] target jointly maximizing the average encoding bit rate and average number of content replicas as well as minimizing the communication load imbalance of video servers.

Cache dimensional problem is considered in [36], where Laoutaris et al.[124] optimized the storage capacity allocation for content distribution networks under a limited total cache storage budget, so as to reduce average fetch distance for the request contents with consideration of load balancing and workload constraints on a given node.

Bo Tan et.al[9] has taken a different perspective, focusing on many-user asymptotics so the results show that the finite storage capacity per node is never a bottleneck. Even in the large catalogue model, it scales to infinity more slowly than the system size. However, none of these identify explicit content placement strategies at the level of the individual peers, which lead to minimal fraction of redirected (lost) requests in a setup with dynamic arrivals of requests.

## 2.4 Load Balancing in Distributed Systems

Load Balancing is a computer networking methodology to distribute workload across multiple computers or a computer cluster, network links, central processing units, disk drives, or other resources, to achieve optimal resource utilization, maximize throughput, minimize response time, and avoid overload [33, 99]. Using multiple components with load balancing, instead of a single component, may increase reliability through redundancy. It can therefore avoid situations where some nodes are heavily loaded while others are idle or doing little work. The load balancing service is usually provided by dedicated software or hardware, such as a multilayer switch or a system server. Using several hosts to provide services together provides high performance-price ratio, high reliability and scalability so as to reasonably assign the task to each cluster of servers and make each server work more and thus it raises the service quality of the whole server system. It is an effective mechanism in achieving high service rate by servicing the requests from the Proxy Server located near the client. The main objective of load balancing algorithms in a VoD distributed system is to minimize the waiting time for a service to begin and to maximize the service rate at the Proxy Server [116]. This is achieved by allowing the Proxy Servers with small retrieval bandwidth to help out Proxy Servers that are temporarily overloaded.

Distributed systems offer the potential for sharing and aggregation of different resources such as computers, storage systems and other specialized devices. These resources are distributed and possibly owned by different agents or organization [6, 27]. The users of a distributed system have different goals, objectives and strategies and their behavior is difficult to characterize. In such systems the management of resources and applications is a very complex task. A distributed system can be viewed as a collection of

computing and communication resources shared by active users. When the demand for computing power increases, the load balancing problem becomes important. The purpose of load balancing is to improve the performance of a distributed system through an appropriate distribution of the application load. [95, 96, 99]

A general formulation of this problem is as follows: given a large number of jobs, find the allocation of jobs to computers which optimizes a given objective function (e.g.total execution time). One way to deal with the management of distributed systems is to have an unique decision maker that will lead the system to its optimum. This is the approach used in most of the existing load balancing solutions [33]. Another way is to let the users to cooperate in making the decisions such that each user will operate at its optimum. The users have complete freedom of preplay communication to make joint agreements about their operating points.

There is a large body of literature on load balancing and all these studies can be broadly characterized as static and dynamic. Static load balancing uses a priori knowledge of the applications and statistical information about the system. Dynamic load balancing bases its decision making process on the current state of the system. A good load balancing scheme needs to be general, stable, scalable, and be able to add a small overhead to the system. These requirements are interdependent, for example a general load balancing scheme may add a large overhead to the system, while an application specific load balancing scheme may have a very small overhead.

### 2.4.1 Need of Load Balancing Algorithms

The primary aim of load balancing algorithms is to achieve an overall improvement in system performance at a reasonable cost. An algorithm that balances the load perfectly but uses a lot of computation or communication

is of not much use. However there may be priorities attached to the nodes or jobs which have to be taken care of by the algorithm. The scalability of the algorithm is very important as the distributed system in which it is implemented may change in size or topology. Hence the algorithm must be flexible enough to allow such changes to be handled easily [99, 100]. It needs to have a degree of fault tolerance i.e. it must be able to recover in case some nodes go down or give faulty results. It has to maintain system stability

### 2.4.2 Taxonomy of Load Balancing Algorithms

As shown in Fig. 2.2, Load balancing algorithms are classified as Dynamic Load Balancing and Static Load Balancing algorithms. Load balancing schemes can be distinguished depending on the knowledge about the application behavior [18, 33, 34]. The first one, Static load balancing, is used when the computational and communication requirements of a problem are a priori. In this case, the problem is partitioned into tasks and the assignment of the task-processor is performed once before the parallel application initiates its execution. The second approach, Dynamic load balancing schemes, is applied in situations where no priori estimations of load distribution are possible. It is only during the actual program execution that it becomes apparent how much work is being assigned to the individual processor. The third one is Hybrid load balancing condition when dynamic and static are merge together and perform to take the advantages of both conditions.

#### 2.4.2.1 Static Load Balancing Algorithms

As shown in Fig. 2.3, Static load balancing algorithms are **Golomb Recovery or Trapezium.** In mathematics, the term "Golomb Ruler" refers

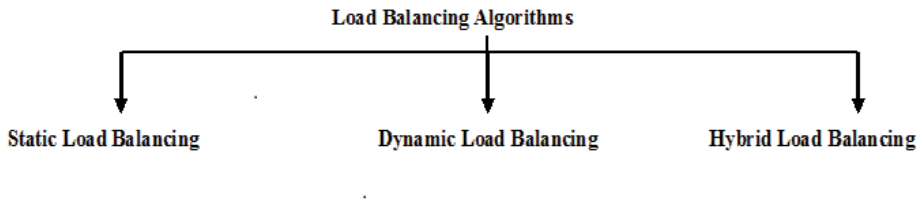


Figure 2.2: Classification of Load Balancing algorithms

to a set/sequence of non-negative integers such that no two distinct pairs of numbers from the set/sequence have the same difference. In Golomb ruler, no two pairs of integers (also called marks in the sequence) measure the same distance. Trapezium recovery scheme provides better optimality, performance and is simple to calculate as compare to Golomb recovery scheme. In Trapezium recovery scheme, first part is based on Golomb ruler and second part is based on new way.

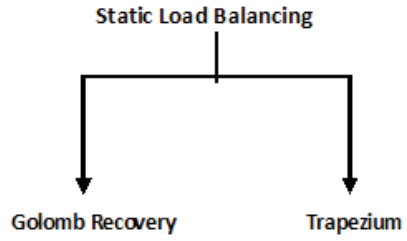


Figure 2.3: Classification of Static Load Balancing algorithms

In this method the performance of the processors is determined at the beginning of execution. Then depending upon their performance the work load is distributed in the start by the master processor. The slave processors calculate their allocated work and submit their result to the master. A task is always executed on the processor to which it is assigned that is static load balancing methods are non-preemptive [99]. The goal of static load balancing method is to reduce the overall execution time of a concurrent

program while minimizing the communication delays. It does not depend on the current state of the system. Prior knowledge of the system is needed. A general disadvantage of all static schemes is that the final selection of a host for process allocation is made when the process is created and cannot be changed during process execution to make changes in the system load. Some of the examples of static load balancing: -Round Robin algorithm, Randomized algorithm, Central Manager Algorithm, Threshold algorithm.

#### 2.4.2.2 Dynamic Load Balancing Algorithms

Dynamic Load balancing algorithms are classified into four main sub strategies Initiation, Location, Exchange and Selection.

**1. Initiation**

The initiation approach specifies the system which invokes the load balancing behavior. This may be a episodic/periodic or event-driven initiation. Episodic initiation is a timer based initiation in which load information is exchanged every preset time interval. The event-driven is a usually a load dependent policy based on the observation of the local load. Event-driven strategies are more reactive to load imbalances, while episodic policies are easier to implement [95]. However, episodic policies may result in extra overheads when the loads are balanced. Event-driven policies are sender-initiated, receiver-initiated or symmetric.

**2. Load-Balancer Location**

This approach specifies the location at which the algorithm itself is executed. The load balancing algorithm is said to be central if it is executed at a single processor, determining the necessary load transfers and informing the involved processors. On the other hand, distributed algorithms are further classified as synchronous and asynchronous. A synchronous load-balancing algorithm must be executed simultaneously at all the participat-

ing processors. Asynchronous algorithms, it can be executed at any moment in a given processor, with no dependency on what is being executed at the other processors.

**3. Information Exchange**

This specifies the information and load flow through the network. The information used by the dynamic load-balancing algorithm for decision-making can be local information on the processor or global information gathered from the surrounding neighborhood. The communication policy specifies the connection topology (network) of the processors in the system or the task-exchange by sending the messages to its neighboring processing elements. This network doesnt have to represent the actual physical topology of the processors. A uniform network indicates a fixed set of neighbors to communicate with, while in a randomized network the processor randomly chooses another processor to exchange information with.

**4. Load Selection**

Load selection is a very vital part of a system in which the processing elements decide from which node to exchange load. Apart from that, it specifies the appropriate load items (tasks) to be exchanged. Local averaging represents one of the common techniques. The overloaded processor sends load-packets to its neighbors until its own load drops to a specific threshold or the average load. Depending on the current state of the system, load balancing algorithms can be divided into two categories; Dynamic Load Balancing and Static Load Balancing as shown in fig 2.2. In order to define a Load-balancing algorithm completely, the main four sub-strategies are Initiation, Location, Information exchange and selection. Depending on who initiated the process, load balancing algorithms[3] can be of three categories as follows:

**Sender Initiated:** If the load balancing algorithm is initialized by the sender. In this type of algorithm the sender sends request messages till it

finds a receiver that can accept the load.

**Receiver Initiated:** If the load balancing algorithm is initiated by the receiver. In this type of description algorithm, the receiver sends request messages till it finds a sender that can get the load.

**Symmetric:** It is the combination of both sender initiated and receiver initiated algorithms.

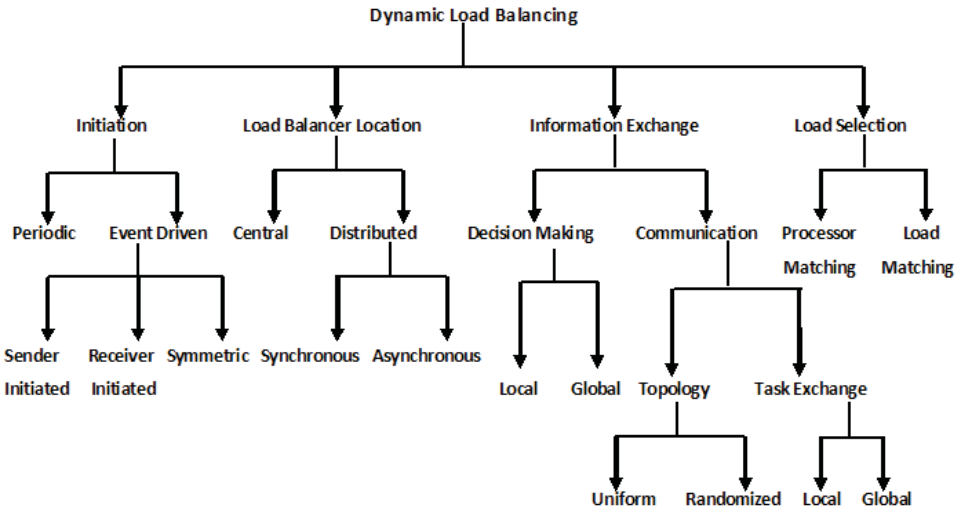


Figure 2.4: Classification of Dynamic Load Balancing algorithms

The main problem with Static Load balancing (SLB) algorithms was that they assume too much job information which may not be known in advance even if it is available moreover intensive computation may be involved in obtaining the optimal schedule. On account of this drawback much of the interest in load balancing research has shifted to Dynamic Load Balancing (DLB) algorithms that consider the current load conditions (i.e. at execution time) in making job transfer decisions. So here the workload is not assigned statically to the processing hosts as was being done in SLB but in-

stead of this workload is being be redistributed among hosts at the runtime as the circumstances changes i.e. transferring the tasks from heavily loaded processors to the lightly loaded ones[20]. Decisions on load balancing are based on current state of the system. No prior knowledge is needed. So it is a better than static approach.

In Dynamic load balancing approach, the decision on load balancing is based on the distributed system. Tasks are allowed to move from an overloaded system to under loaded system to receive faster service[23]. Dynamic approach is better as it considers the state of the system in making the load balanced. It is of two types: Distributed Load Balancing and Non-distributed Load Balancing

**i. Distributed Load Balancing:** In this system the Load Balancing Algorithm is executed by all the nodes in the system and load balancing responsibility is shared by all the nodes. Distributed Dynamic Load Balancing Algorithms generally tend to produce more message passing due to the distributed approach each node need to communicate with other nodes to make the decisions.

**ii. Non-Distributed Load Balancing:** In this system the nodes are not responsible for load balancing. Instead a load balancing master is chosen who does the load balancing for the entire system.

Algorithms can be classified Based on interaction between nodes:

**Cooperative:** In Cooperative systems, nodes work closely to achieve a global objective i.e. to improve the system's overall response time.

**Non-Cooperative:** In Non-Cooperative systems, nodes work independent of each other in order to obtain a local goal i.e. to improve the individual system's response time. A Dynamic Load Balancing (DyLB) algorithms[20] consider following issues:

(1) Load estimation policy, which determines how to estimate the workload of a particular node of the system. (2) Process transfer policy, which

determines whether to execute a process locally or remotely. (3) State information exchange policy, which determines how to exchange the system load information among the nodes. (4) Priority assignment policy, which determines the priority of execution of local and remote processes at a particular node. (5) Migration limiting policy, which determines the total number of times a process can migrate from one node to another.

Media caching in Soam Achary et.al[91] operates a collection of Proxy Servers as a scalable cache cluster media objects which are segmented into equal sized segments and stored across multiple proxies, where they can be replaced at a granularity of a segment. There are also several local proxies responsible to answer client requests by locating and relaying the segments.

To achieve better load balance and fault tolerance,a data layout is suggested in[76],which partitions a media object into segments of increasing sizes, stores more copies for popular segments, and yet guarantees at least one copy stored for each segment.

To achieve a much higher byte-hit ratio with less cache management overhead, Anna Satsiou, et al [92] focuses on an environment of more than one Proxy Server that serve homogeneous or even heterogeneous client preferences for streaming of video files. Under a hierarchical tree topology system of proxies, the prefixes of the videos are stored in small size proxy caches, each located very close to the corresponding client community, while larger caches located further away from the client communities are used to cache the latter segments of the videos requested by more than one client community. Frequency-Based Cache Management Policies are used to efficiently and dynamically cache the content of the most popular videos among the various proxies.

To minimize service delays and to reduce the loads placed on the network resources, Wallapak Tavanapong, et al. [93] have proposed a Video Caching

Network (VCN) that utilizes an aggregated cache space of distributed systems along the delivery path between the server and the users for caching popular videos. VCN is set up and adjusted dynamically according to users locations and request patterns.

Naoki Wakamiya et al.[94] have investigated mechanisms in which Proxy Servers cooperate with each other. The proxy is capable of adapting to incoming or cached video blocks at the users request by means of transcoders and filters. On receiving a request from a user, the proxy checks its own cache. If an appropriate block is not available, the proxy retrieves a block of a higher quality from the video server or a nearby proxy. The retrieved block is cached,its quality is adjusted to that which was requested as necessary, and is then sent to the user. Each proxy communicates with the others and takes the transfer delay and video quality into account in finding the appropriate block for retrieval.

Y.C Tay, et al [95], have explored the feasibility of linking up several small multimedia servers to a (limited-capacity) network, and allowing servers with idle retrieval bandwidth to help out servers that are temporarily overloaded. The goal is to minimize the waiting time for service to start. This work has introduced an algorithm called Global waiting Queue(GWQ)load balancing algorithm. It puts all pending requests in a global queue, from which a server with idle capacity obtains additional jobs.They propose an enhanced GWQ+L algorithm that allows a server to reclaim active local requests that are being serviced remotely. The main characteristics of the GWQ algorithm are: 1.A server serves a remote request only when it is fully loaded, if I receives a local request, and it does not have enough resources to attend to this request. 2.The first priority of every server is to service requests from its Local Queue. Remote Requests from the Remote Queue are attended only when its Local Queue is empty. 3. Each request is serviced by the same server throughout its lifetime. The

drawback of this algorithm are: 1. Each video object is replicated at every server. Actually, replication of complete video object at every server is not a prerequisite for this scheme, but the case of partial replication is not analyzed. All videos are requested with the same probability and hence the popularity of the videos is not considered. When there are several servers, the remote request is assigned to the first server that returns a positive answer. Thus, the current load of each applicant server is not considered. Hence,the assigned request may increase the load on the server and also increases the possibility of generating new remote requests for that server.

With the goal of minimizing the waiting time for a service to start, S. Gonzalez, et al.[96] considered a distributed VoD system in which only the most popular videos are replicated in all the servers, whereas the rest of them are distributed through the system, following some allocation scheme. This work also presents an algorithm to efficiently balance the load in the proposed system. This balancing achieved by assigning the jobs to other servers, increases the network traffic.

In order to accomplish, low-delay and high-quality video distribution without imposing extra load on the system, the video streaming system has been proposed by Yoshiaki Taniguchi,et al. [97]. This system consists of a video server and multiple Proxy Servers. In this mechanism, the proxies communicate with each other and retrieve missing video data from an appropriate server by taking into account transfer delay and affordable quality. In addition, the quality of cached video data is adapted appropriately at a proxy to scope with the client-to-client heterogeneity in terms of the available bandwidth, end-system performance, and user preferences on the perceived video quality. Minseok Song, et al.[98] proposes an adaptive data retrieval scheme for load sharing in clustered video servers. They have analyzed how the data retrieval period affects the utilization of disk bandwidth and buffer space, and then developed a robust period management policy to

satisfy the real-time requirements of video streams This work also has proposed a new data retrieval scheme in which the period can be dynamically adjusted so as to increase the disk bandwidth capacity of heavily loaded clusters and increase the number of clients admitted.

Yin-Fu Huang, et al.[99] focused on dynamic load balancing among the clustering servers in the VoD system. This is to support more clients with reduced average response time of requests.The load balancing among the servers can be achieved by the load mechanism Jun Guo, et al.[100] have established a conjecture on how to balance the movie traffic load among groups of disks to maximize the level of disk resource sharing. For a given file replication instance, the conjecture predicts in general an effective lower bound on the blocking performance of the system. The design of a numerical index hat measures quantitatively the goodness of disk resource sharing on allocation of multi-copy movie files is also proposed. This work has proposed the design of a greedy file allocation method that decides a good quality heuristic solution for each feasible file replication instance.

Jonathan Dukes, et al. [101] describe in detail the implementation of dynamic replication in a server cluster environment. They also describe the architecture of the Hammer Head multimedia server cluster. Hammer Head has been developed as a cluster-aware layer that can exist on top of existing commodity multimedia servers. This prototype takes the form of a plug-in for the multimedia server in Microsoft Windows Server 2003. Replicated state information is maintained using the Ensemble group communication toolkit. Xiabo Zhou, Cheng-Zhong xu [102] have proposed an algorithm for video replication and placement algorithm, which utilizes the information about Zipf-like video popularity distribution. They have replicated the complete videos evenly in all the servers, for which the storage capacity of individual Proxy Server should be very large to store all the videos. This may not allow each server to store replicas of more number of videos.