Semi-Continuous Transmission for Cluster-Based Video Servers

Sandy Irani and Nalini Venkatasubramanian
Department of Information and Computer Science
University of California, Irvine, CA 92697-3425
Email:{irani,nalini}@ics.uci.edu

Abstract

With advances in storage technology, the ability to provide client end storage for continuous media applications has become a possibility. Transmission of data in cluster based multimedia environments can be semi-continuous in conjunction with client side buffering and staging. Experiments indicate that a client buffer size (staging degree) of 20 percent (of object size) is near optimal for most objects. The work presented in this paper also addresses the implications of semi-continuous transmission to placement and admission control mechanisms in a cluster-based multimedia server. We improve admission control by introducing a technique called dynamic request migration in cluster-based multimedia servers that is enabled by client staging. Simulation studies demonstrate that close to maximum utilization can be achieved even if at most one migration within the server cluster is performed for each request arrival and each request is migrated at most once during its lifetime. Furthermore, our performance results reveal that with client staging and dynamic request migration, even naive placement techniques are tolerant to extreme variations in request patterns. In fact, our results indicate that under most circumstances one can be oblivious to request pattern variations during placement, eliminating the need to predict relative popularities of objects.

1. Motivation

Recent advances in networking technologies have enabled high bandwidth communication infrastructures and widespread availability of large bandwidth at clients and servers, making applications such as video-on-demand to the desktop feasible and affordable. Many of the original continuous media servers assume that clients accessing data from multimedia servers have very limited capability in terms of processing and storage. Current trends in storage and processing technology have caused a tremendous drop in storage cost. It is now reasonable to assume that there is sufficient storage to buffer limited data at the client end. We use the term *client staging* to refer to workahead transmission into (larger) client disk storage (the staging buffer) and *client buffering* to refer to transmission into a (smaller) client memory buffer. The ability to store data at the client side opens up possibilities of *semi-continuous transmission* of multimedia data where the server can download information to the client in fixed or variable sized segments.

In this paper, we study techniques to make effective utilization of semi-continuous transmission to further the overall performance of the system. We are especially interested in the implications of client side storage on admission control and placement policies at the server side of a cluster-based multimedia server. *The objective of these techniques is to improve utilization of system resources and request success ratios.* Specifically, we study the following

- Performance benefits obtained by client staging
- Admission control techniques at the server side to exploit the availability of client staging. We propose a mechanism called Dynamic Request Migration (DRM) and study the performance benefits of this technique.
- Performance impact of placement policies with client staging and dynamic request migration. We analyze how the ability to provide semi-continuous transmission impacts the performance of distributed object placement strategies.

The rest of this paper is organized as follows. We describe the overall architecture of the cluster-based multimedia server in Section 2. In Section 3, we describe issues that arise in the implementation of semi-continuous transmission of continuous media and propose some mechanisms for admission control and placement in cluster-based multimedia server environments. Performance evaluation of the proposed techniques is presented in Section 4. We address related work in Section 5 and conclude in Section 6 with future research directions.

2. System Architecture

Multimedia server environments vary widely based on the application domain,from tens/hundreds of servers for a nation-wide educational VOD facility to less than a dozen servers for small enterprise intranets. We assume the following architecture of a typical cluster-based video server (see Figure 1).

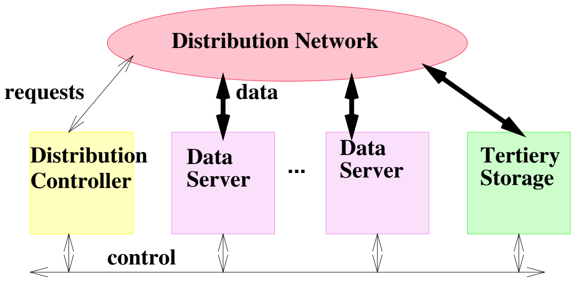

Figure 1. Architectural view of a cluster-based multimedia server

The video server consists of a cluster of multiple independent and distributed data sources (servers), with high capacity storage devices (e.g. hard-disks), processor, buffer memory, and high-speed network interfaces for real-time video retrieval and transmission [24]. To accommodate a large number of video objects, the video server cluster includes tertiary storage. A central distribution controller (DC) governs the operation of the data sources within the cluster. When a request to view a particular video arrives in the system, the distribution controller must decide whether or not to accept the incoming request based on current resource allocation. If the request is accepted, it must be allocated to a particular server (i.e. data source) within the cluster which holds a replica of the requested video and which also has the available resources to begin transmission immediately. In the typical mode of operation, the data sources guarantee to provide appropriate quality of service to each request until its completion. All the above components are interconnected via an external distribution network that provides client/subscriber connectivity. A lower speed back-channel conveys client commands back to the data sources via the DC. Note that in this architecture, scheduling a request on a data source implies that the request is serviced using storage on that data source. In contrast, other architectures [8, 10] assume the availability of shared storage among the servers. The scalability of our architecture is attributable to the ability to add additional data sources to the server in order to provide increased storage capacity and transfer bandwidth. We believe that assuming non-shared storage subsystems across servers will enable us to scale our management mechanisms to server complexes distributed over a wide area network easily.

Since storage is limited at each server and the video objects are large, it is not feasible to store a replica of each video on each server. Thus, a *video placement* strategy must be devised. The placement strategy decides when, where and how many replicas of a video object will need to be created to satisfy incoming requests. While a copy of a given video which resides on a server can be used to satisfy many requests for that video object, each request uses a portion of the bandwidth of the server. Naturally, the relative popularity of videos will differ, so researchers have focused on video placement strategies to balance the popularity of a video with its storage requirements so that both the storage and bandwidth of the servers can be used effectively.

3. Semi-continuous transmission

In continuous transmission, the video is transmitted in a continuous stream at the rate at which the video must be viewed (called the *view bandwidth*). This amount of bandwidth is reserved on the assigned server for the duration of the transmission. In semi-continuous transmission, the client staging buffer is used to transmit a portion of the video before it is actually viewed. The parameters of the workahead transmission into the client staging buffer is based on the capacity of the client staging buffer and the need to guarantee continuous playback at the client.

Semi-continuous transmission can be implemented in a variety of ways. It is possible to transmit the video in fixed size segments each of which are transmitted in a non-continuous manner some time before that particular segment will be viewed. Alternatively, it is possible to implement this by simply varying the bandwidth of each continuous transmission. Since the former approach is a special case of the latter, we will assume variable width transmission since this offers the most flexibility in transmission.

Semi-continuous transmission has the potential to offer performance benefits in different ways. First of all, it has the effect of smoothing out natural fluctuations in the arrival rate of requests to servers. If client staging is feasible, then spare bandwidth can be utilized when the number of requests in the system is below the average which will result in the availability of more bandwidth when the demand is above average. Variabilities are buffered permitting reallocation of resources on the fly. Secondly, client staging opens up the possibility for *dynamic request migration* (DRM) at the server during admission control.

3.1. Dynamic Request Migration

Suppose that a request for a video arrives and all servers which hold a copy of the requested video do not have any bandwidth available for the new request. A straightforward approach to deal with this would be to reject the request; more resource intensive solutions perform dynamic replication of the requested object on another server where resources can be made available. However, it is possible that some server with the requested video object may have another currently active request which can be migrated to another server. With DRM, the active request can be migrated to the other server, releasing bandwidth for the newly arrived request. This kind of dynamic request migration is difficult to implement without staging/buffering at the client since the request transfer can take some time which would otherwise result in jitter observed by the client. Jitter during stream switching can be a serious issue depending on the NW infrastructure (reestablishing QoS routes and network connections in a general Internet domain is time consuming). This can be handled by using special purpose software at switches and network components [3, 4]; however, this is not always possible. Sufficient client staging also opens up the ability to delay switching till resources occupied in the short term become available; this information is useful for adaptation purposes. Dynamic request migration can also be used to engineer a limited degree of fault tolerance into the server since the ability to dynamically switch servers for a single stream can help deal with node server failures.

We use the term *migration chain length* to refer to the number of requests that must be migrated to accommodate an incoming request. The term *hops per request* will denote the number of times a request can be migrated during the course of its lifetime. Throughout this paper, we will refer to the effects of *semi-continuous transmission* as the combined effect of both *client staging* and *dynamic request migration*.

3.2. Video Placement

Many sophisticated schemes have been devised for video placement as well as request assignment and migration [29, 26]. These schemes use statistics to predict the relative popularity of videos to guide initial video placement and then dynamic reallocation to adjust placement to discrepancies between predicted and actual demand. The results of our study indicate that with current technology, very simple schemes which take advantage of limited staging at the client are sufficient to provide close to 100% utilization. We explore two simplistic placement strategies:

• Even Allocation(placement): This strategy allocates the same number of copies to each video (with rounding done at random). Each copy is placed on a randomly chosen video server.

• Predictive Allocation(placement): The number of copies of each object is proportional to its predicted popularity. Once the number of copies of each video is decided, copies are distributed randomly to the servers.

We focus on the even allocation scheme in conjunction with client staging and buffering. The request assignment algorithm assigns each newly arrived request to the server which has a copy of the requested video and has the fewest current requests. A very limited amount of request migration is attempted if all servers which hold a copy of the requested video are full. If this fails, then the request is not accepted. Our results show that even these very simple schemes perform very well in most circumstances. In fact, the even allocation which is completely oblivious to which videos are the most popular achieves very good utilization except when the relative demand for the videos is highly uneven. The even allocation eliminates the need to predict the relative popularity of videos. It also enables the system to use a single video placement strategy even if the relative demand of different videos changes over the course of time. The reason for the success of such simple schemes is twofold:

•**Semi-Continuous Transmission.** The benefits of buffering/staging a portion of the video at the client and very limited dynamic request migration in most situations are sufficient to compensate for faulty video placement schemes.

• **Large Server to View Bandwidth Ratio.** A crucial parameter in server utilization is the ratio of the server bandwidth to the bandwidth at which a video is viewed. We call this the server-to-view bandwidth ratio (SVBR). Values for the SVBR which are consistent with current technology actually make it difficult to make a system perform poorly. Since the SVBR is likely to grow in the future, utilization will grow accordingly. Our experiments which show the impact of the SVBR on the utilization of the system are reported in the full version of this paper [15]. We also show an analytical expression which gives the expected utilization as a function of the SVBR for a one server system. The fact that the analytical results are very close to the empirical results, as shown in [15], also validates the accuracy of out experimental results.

3.3. Client Staging

More extensive use of client staging and buffering is what is needed to perform dynamic request migration and help improve system performance. Client Staging has the effect of smoothing out fluctuations in the arrival sequence which allows the system to accept more requests when the system receives more than the average number of requests. This happens because a server can transmit at a higher bandwidth when there are fewer requests in the system. The requests which are transmitted at a higher bandwidth finish earlier which frees up the system later when there may be more requests.

Let b_{view} be the bandwidth at which the data is viewed. The *projected finishing time* of a request is the time at which the entire data object will have been transmitted if data is sent at a rate of b_{view} from the current time on. A request is said to be *unfinished* if all the data has not yet been sent. The *deadline* of a request is the time by which all the data must be transmitted if it is to be viewed continuously by the client at a rate of b_{view}. We focus on a class of algorithms which will allocate at least a minimum bandwidth, say b_{view}, to any unfinished request. We call these algorithms *minimum-flow* algorithms. Note that this may not necessarily be the optimal choice since it may be preferable to stop transmission on a request which has a large amount of data stored on the client buffer in order to accept a new incoming request. We refer to the class of algorithms where a stream alternates between periods of transmission and no transmission as *intermittent* algorithms. In this paper, we restrict the discussion to minimum-flow algorithms. The reason for this restriction is that it makes the decision procedure to determine if a new request can be allocated to a given server very simple. For minimum-flow algorithms, a new request can be allocated to a given server if and only if the sum of the view bandwidths of all the unfinished requests allocated to that server plus the view bandwidth of the new request is at most the bandwidth capacity of the server. The decision procedure for the optimal intermittent algorithm is impractical to apply in real time.

We propose a scheduling algorithm which decides the best way to use extra available bandwidth in transmitting data ahead of time. The algorithm *Earliest Finishing Time First* (EFTF) picks the active request with the earliest projected finishing time whose client also has available buffer space and allocates as much bandwidth to that request as can be handled by the receiving client. We prove that the EFTF algorithm is optimal among minimum-flow algorithms when there is no bound on the bandwidth at which clients can receive data. Of course, it will not be the case in general that a typical client can receive data at the same rate at which a server can transmit data, so the theorem does not necessarily hold in our simulations with limits on the receiving bandwidth of clients, but empirically it does very well. It is, in fact, impossible that any algorithm can achieve optimality with limits on the receiving bandwidth of clients without knowledge of future request arrivals.

The EFTF procedure (See Figure 3.3) is called whenever a new request arrives, a request finishes transmission, a client's buffer becomes full or a client's buffer becomes empty.

Theorem 1 *If the videos are not paused and there are no limits on the bandwidth at which clients can receive data, then EFTF is optimal among minimum-flow algorithms, in that for any set σ of request arrivals which can all be accomodated by any scheduling algorithm, EFTF will accommodate σ.*

The proof of the above theorem is given in the full version of this paper [15]. What EFTF does not indicate is which set of requests should be accepted; it merely indicates the best way to allocate bandwidth for the set of accepted requests. We adopt a policy that will allow a request to be assigned to a server as long as the number of unfinished requests allocated to that server is less than the ratio of the server bandwidth to the view bandwidth, i.e. as long as there are resources to service the request. This may not necessarily be the optimal decision. It may be preferable to not accept a newly arrived request even if there is enough available bandwidth in order to have bandwidth available to accept future requests. It is, however, impossible that any algorithm can decide the optimal set of jobs to accept without knowledge of future request arrivals.

4. Performance Evaluation

4.1. Experimental Design

In all the simulations below, a static video placement is determined before any requests arrive. The number of copies of each video is first decided. Then a subset of the servers is chosen at random for each video and copies of that video are placed on the selected servers. In the even video placement, the same number of copies are chosen for each video and rounding is done at random. For example, in one system we consider, there is an average of 2.1 copies of each video. This means that each video has either two or three copies and those videos which have three copies are selected at random. In the predictive video placement policy, the algorithm has complete knowledge of the relative popularity of each video and makes copies of each video in proportion to this popularity. The algorithm, however, is required to make at least one copy of each video, assuming the availability of storage.

The arrival process of requests is Poisson. The arrival rate is chosen so that if all the requests are accepted, the utilization will be 100%. That is, the expected sum of the sizes of all requested videos is equal to the number of servers times the server bandwidth times the length of the simulation. The arrival rate is chosen so as to place as much stress as possible on the system and accentuate the differences between the different schemes.

We measure the performance of the system in terms of bandwidth utilization and request rejections. That is, we sum the size of all transmissions and divide that number by the total amount of data which could be sent if all servers were sending data at the maximum bandwidth for the duration of the simulation. This value is the ratio of data sent to the maximum amount that is possible to send. Note that

EarliestFinishTimeFirst
At each event, do the following:
Let R be the set of unfinished requests.
Let F be the set of requests whose clients no **not** have full buffers.
Let b_{server} be the bandwidth capacity of the server
Let $b_{remain} \leftarrow b_{server}$
Let b_r be the amount of bandwidth currently allocated to request r
for all requests r.
$b_r \leftarrow 0$.
for each request $r \in R$,
allocate b_{view} in bandwidth to r
$b_r \leftarrow b_{view}$
$b_{remain} = b_{remain} - b_{view}$
while ($b_{remain} > 0$ and $F \neq \emptyset$)
Pick the request r in F with the earliest projected finishing time
Let $b_{receive}$ be the maximum bandwidth at which the client who initiated request r can receive data
Allocate an additional $\min\{b_{remain}, b_{receive} - b_r\}$ in bandwidth to r
$b_{remain} \leftarrow b_{remain} - \min\{b_{remain}, b_{receive} - b_r\}$

Figure 2. The Earliest Finishing Time First Algorithm

there may be other resources which are consumed on a per request basis (for example, CPU cycles). However we assume that each of these resources is used in proportion to the bandwidth usage. When all servers are homogeneous, we can assume without loss of generality that bandwidth is the limiting resource which dictates the maximum number of requests which can be sustained at a time. If it were actually a different resource which proved to be the bottleneck, we could define utilization in terms of that resource and the results would be the same.

When a request arrives, it is placed on a server which currently has a copy of the requested video and has sufficient bandwidth to satisfy the request. If many such servers exist, the placement algorithm used here places the new request on the server with the minimum number of active requests. If such a server does not exist, then the request is not accepted (in the case when there is no dynamic request migration).

In all the experiments, the relative popularity of the videos is chosen according to a Zipf-like distribution [10]. A Zipf distribution takes two parameters, N and θ. N is the number of items in the distribution. c is a normalization parameter: $c = 1/(\sum_{i=1}^{N} 1/i^{1-\theta})$. The probability that a newly arrived request is for video i is: $p_i = c/i^{1-\theta}$. θ is a parameter which corresponds to the amount of skew in the demand. Typically, this number is varied in the range $0 \leq \theta \leq 1$. $\theta = 0$ corresponds to a highly skewed distribution and $\theta = 1$ corresponds to a completely uniform distribution. Values around $\theta = .27$ have been used in previous studies [10, 28]. N also effects the degree to which the distribution is skewed. In this study, we vary θ from $-.5$ to 1, giving an even wider range of parameters than is usually studied. Note that the distribution is well defined for negative values of θ. It just means that the distribution is highly skewed. For a fixed value of θ, a larger N results in a more highly skewed distribution.

Unless otherwise stated, a video server system consists of a collection of homogeneous servers. The rate at which videos are viewed is 3 Mb/s. Each data point is the result of 5 trials, each of which lasts 1000 hours. In order to focus our empirical study, we run all experiments on two systems with widely differing characteristics. The first system is a large video server which delivers feature length movies. The second system is a small video server which delivers shorter video clips. The parameters of each system are given in Figure 3. The length of each video is chosen uniformly at random from the ranges indicated. Further heterogeneity studies that represent a larger range of system configurations are reported in the full version of this paper [15].

4.2. Effect of Dynamic Request Migration

In this section, we examine the effect of dynamic request migration on system utilization. The even video allocation scheme is used throughout the experiments discussed in this section. Only enough staging at the client to allow for request migration is performed. The migration chain length which refers to the number of requests that are migrated to accommodate an incoming request is kept at one throughout

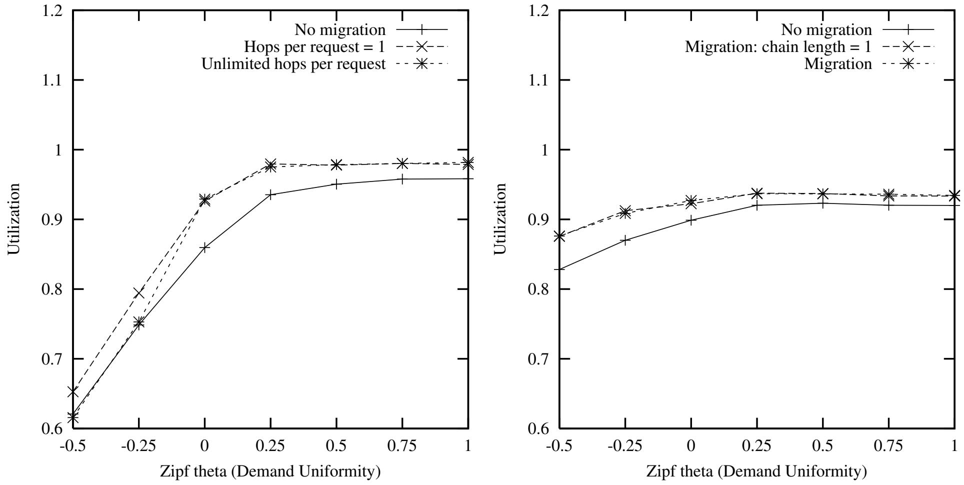

Figure 4. The effect of dynamic video migration. The results for the large system are to the left and the results for the small system are to the right.

System	Small	Large
Number of Servers	5	20
Bandwidth	100 Mb/s	300 Mb/s
Video Length	10-30 Min	1-2 Hrs
Number of Videos	500	700
Average Number of Copies Per Video	2.2	2.1
Disk Capacity	100 GB	150 GB

Figure 3. Parameters for the two video servers studied.

our experiments. The results below show that request migration even with this simple algorithm can significantly improve utilization. Furthermore, when migration is restricted so that each request is only migrated once over the course of its lifetime (i.e., hops per request = 1), the utilization is almost as good as when an arbitrary number of hops per request were allowed (unrestricted hops per request). The low utilization at negative values of Zipf θ is caused by the even allocation scheme for video placement. This will be discussed in more detail in Section 4.4.

4.3. Effect of Client Staging

In this section, we examine the effect of client staging on system utilization. Our experimental results are shown in Figure 5. Throughout the experiments, we use the even video placement scheme and do not migrate any jobs. As an additional restriction, we restrict the amount of bandwidth which can be used to send data to a single client to 30 Mb per second. Thus, we limit the rate at which a client can receive data to 30 Mb/s. This has the effect of limiting the benefits of semi-continuous transmission slightly. The amount of staging buffer is expressed as a percentage of the storage required to store an entire copy of the average sized video.

The most notable result is that almost the maximum amount of benefit from staging data at the client can be obtained with buffer space which is only 20% of the entire video object. This indicates that it is only worthwhile to allocate this amount of disk space to the staging of an incoming video transmission. The benefit from client staging is more pronounced for the smaller video server. In general, there are two primary factors which serve to cushion natural fluctuations in the arrival rate. These are client staging and a large server to view bandwidth ratio. Since the servers in the larger system already have a larger server to view bandwidth ratio, there is less room for improvement for client staging.

4.4. Effect of Video Placement

As we have observed, the performance of the even allocation scheme suffers under highly skewed distributions. This effect is more pronounced in the larger system. This is due to the fact that in both systems, the average number of copies of each video is roughly the same (around 2.2), but in the larger system, the copies are spread out among a larger number of servers. This means that there are a fewer number of videos on each server and there is less opportunity for variation in the popularity of videos stored on a

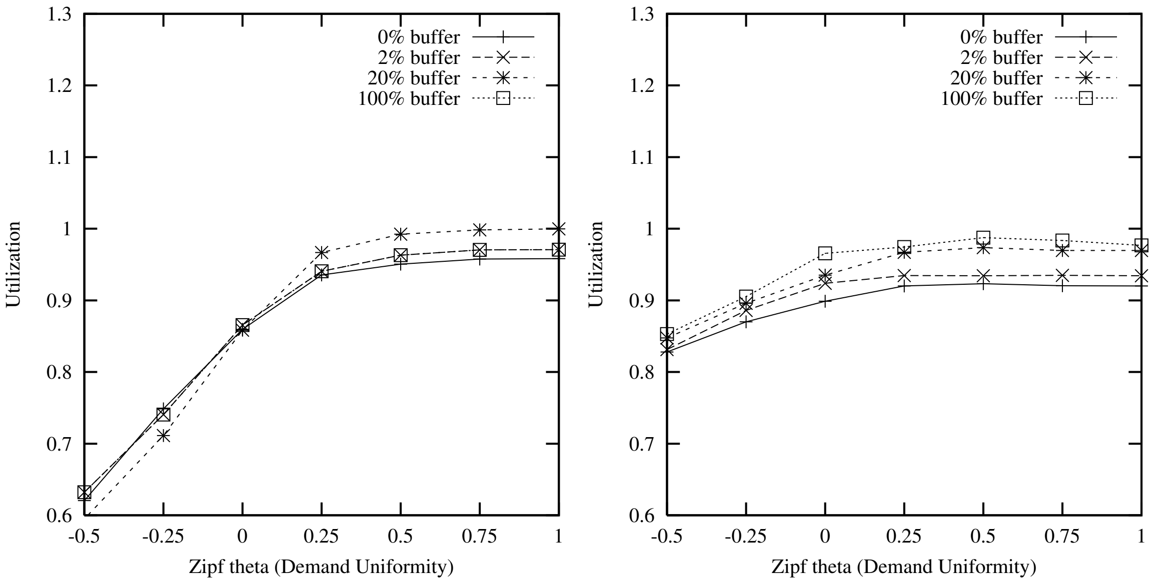

Figure 5. The effect of client staging. The results for the large system are to the left and the results for the small system are to the right. No dynamic request migration is performed in these experiments. The number of copies of each video is uniform.

single server to average out. That is, it becomes more likely that the average popularity of the videos stored on some given server will be higher than the average popularity of the videos over all servers.

A natural question then is how much does the video allocation have to be skewed in order to match the predictive scheme which can predict popularity perfectly. So far we have only discussed the two extremes in our ability to predict the relative popularity of videos. On one hand, we have the even allocation scheme which has no knowledge of the relative popularity of the videos. On the other hand, we have the popularity based allocation which assumes complete knowledge. A more practical scenario is that we have some, but not complete ability to predict how popular the videos will be. In order to explore the spectrum between these two extremes, we introduce a very mildly skewed allocation which makes a few extra copies of the most popular videos. We call this the partial predictive allocation scheme. Detailed experiments [15] show that even this mildly skewed allocation scheme in conjunction with dynamic request migration and client staging can achieve comparable utilization to a perfect predictive video allocation scheme. Thus, it is not necessary to know exactly how popular the most popular videos will be. It is only necessary to identify the ones that are likely to be more popular.

4.5. Semi-Continuous Transmission

In this section, we compare the performance benefit of the integration of the three factors described in this study: dynamic request migration, client staging and video placement. It should be noted that in order to actually implement the predictive scheme, it is necessary to monitor the relative popularity of all the videos and to dynamically replicate and de-replicate videos to adjust to changes in relative demand. In our study, we assume that the predictive scheme has perfectly predicted demand and that no adjustments to the number of replicas need to be made for the duration of the simulation.

Policy Number	Allocation Policy	Migration Policy	Client Staging
P1	Even	No Migr	0% Buffer
P2	Even	No Migr	20% Buffer
P3	Even	Migr	0% Buffer
P4	Even	Migr	20% Buffer
P5	Predictive	No Migr	0% Buffer
P6	Predictive	No Migr	20% Buffer
P7	Predictive	Migr	0% Buffer
P8	Predictive	Migr	20% Buffer

Figure 6. Policies evaluated

We compare the combinations of the different mechanisms using policies illustrated in Table 6. The first four policies do not use any knowledge about the relative popularity of the videos. The last four policies are assumed to have predicted the relative popularity of all the videos perfectly and this is reflected in the placement scheme accordingly. With both allocation policies, we determine the effect of migration, client staging and their combination on the placement schemes. Specifically, we study how migration and client staging can compensate for a faulty placement scheme. We have varied the skew parameter θ beyond the usual $[0, 1]$ interval in order to understand at which point the simpler schemes break down. The results show that for θ in the $[0, 1]$ range, the even allocation schemes perform quite well. In these cases, the use of dynamic request migration and client staging are the dominant factors in improving utilization. That is, policy P4 performs comparable to policy P8 and outperforms the other policies. For the highly skewed distributions in which the Zipf θ is negative, the allocation scheme is the dominant factor in determining performance. In these cases, dynamic request migration and client staging are not enough to compensate for the faulty placement scheme. As discussed in Section 4.4, this is because there are not enough copies of the most popular videos.

4.6. Effect of heterogeneity on semi-continuous transmission

We next studied the effect of heterogeneity of server resource configurations on the overall performance and the system. Our experiments were conducted on 3 classes of systems with 5, 10 and 20 servers within each distributed server configuration. Within each class, we studied the impact of bandwidth and storage heterogeneity on distributed server performance. In each case, we ensured that the number of requests submitted to the servers were sufficient to keep the servers fully saturated in terms of resource utilization. The results of these experiments are omitted here for lack of space but are reported in the full version of this paper [15]. The results show that the effect of heterogeneity (storage and bandwidth) is more pronounced with the smaller system. With a large number of servers, we obtain better overall performance since variabilities are spread out over a larger number of servers. In addition, the effect of storage heterogeneity on system performance seems to be much less pronounced than bandwidth heterogeneity. In fact, the statistical significance of any difference is questionable since in some cases, the systems which have heterogeneous storage actually outperforms the homogeneous system. Thus, our results indicate that the algorithms studied in this paper are robust to variations in the level of storage heterogeneity.

5. Related Work

Many of the initial efforts in designing video servers have focussed on continuous media transmission. Within this context, techniques have been proposed to address placement of media on disk to ensure real-time retrieval [2, 29], admission control procedures to maximize server throughput [27] and buffer management policies to minimize memory requirements [13, 17]. Replication and striping strategies for optimizing storage across disk arrays are described in [16, 25].

A number of algorithms for placement in distributed video servers have been proposed. Considering the storage subsystem alone, a two-stage DASD dancing scheme for load balancing is studied in [28]. In the initial static stage, a greedy assignment of videos to disk groups is obtained using a graph-theoretic approach. The dynamic phase that follows uses the static assignment to perform real-time disk scheduling effectively. The goal of the dynamic phase is the minimization of an objective function that optimally balances disk loads, again, using a graph theoretic approach. A *dynamic segment replication* scheme for partial replication of video objects is proposed in [9]. This mechanism permits the executing requests to be dynamically migrated to a replica on a less loaded device. The trade-offs between storage space and transfer bandwidth are highlighted in [10]. This work proposes a multi-phase online video placement policy that attempts to match the bandwidth to space ratios of videos with that of storage devices in a video server, in order to increase server throughput. A revenue based greedy matrix algorithm that attempts to maximize revenue generated by the service provider is discussed in [26]. More recent work studies the impact of dynamic replication techniques for continuous media servers [7].

More recently, a number of techniques have been proposed for managing video transmission at the proxies and the client [6]. Proxies are situated between servers and clients and provide services that lead to enhancement of QoS perceived by the client. Proxy cache management techniques take into account workload characteristics and make decisions on when, where and how to store and replace objects [18]. Video staging exploits disk storage space of proxy servers at the network edges to reduce bandwidth consumption. Video smoothing [20] techniques reduce rate variability of video transmission by using in-memory buffers in the delivery path, e.g. at a proxy server, to perform workahead transmission into a client buffer and reduce burstiness. *Prefix caching* is another strategy employed at the proxy where the first segment of many popular continuous media objects are cached [22]. Patching techniques reduce the overhead of communication by allowing a client to obtain a portion of a multimedia stream by listening(snooping) into ongoing transmissions of the required

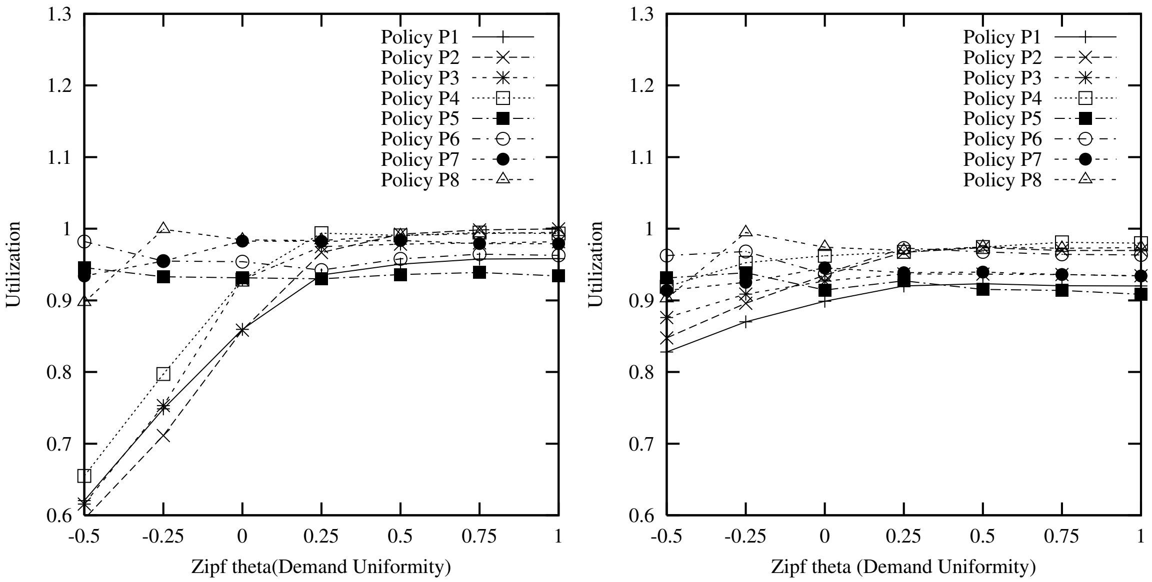

Figure 7. Comparing the effects of adaptive video placement on sever utilization. The results for the large system are to the left and the results for the small system are to the right. In these experiments, dynamic request migration and client staging is used. The clients have buffer capacity which is 20% of the size of a video.

streams to other clients. Efficient implementations of patching require sophisticated buffer management strategies [21] and controlled multicast protocols [12].

A number of broadcast and multicast techniques have been studied for effective utilization of network bandwidth [5, 14, 1] for video-on-demand systems. Multicast smoothing techniques [23] attempt to reduce network bandwidth and provide efficient delivery of services to heterogeneous clients by integrating video smoothing with differential caching at intermediate nodes. [11] presents a greedy disk conserving broadcast based scheme for optimizing the utilization of resources in the network, client and server; the work specifically addresses the issues of client resource requirements both in terms of I/O and storage. Recent work suggests the need for composing services such as workahead smoothing, prefix caching, patching etc. to provide a integrated proxy management environment [19].

Our work complements recent work in this area on proxy server based management techniques for MM transmission. The work presented in this paper addresses the implications of client side buffering and staging techniques to placement and admission control mechanisms at the server end. Specifically, we address server side techniques enabled by the introduction of client side buffering on disk. We present improved admission control mechanisms that exploit the ability to perform dynamic request migration when there are no available replicas to schedule a request to. We also demonstrate how replica placement and management policies in a cluster-based server can be greatly simplified without any degradation in performance. Traditional client buffering schemes have considered buffering at the client of 1-2 minutes of stored video (about 100Mbyte buffers). Our work differs from this in that we assume the availability of disk storage at the client that can buffer larger segments of video (e.g. 20 percent of the total video object size).

6. Future Research Directions

We plan to continue further performance evaluations with more server configurations and request types. Specifically, we would like to analyze the impact of non-continuous media such as text and images on semi-continuous transmission, e.g. in multimedia Web scenarios. Our work has so far focused on clusters of servers with non-shared storage, some of the techniques proposed in this paper are applicable to shared storage clusters as well. Scalability is an important issue in designing wide-area distributed multimedia services. We intend to study the performance of the proposed technique as the number of servers and requests scale. Integration of the proposed admission control and placement schemes with proxy servers is another area of future research. Implementing multicast in the context of semi-continuous transmission can be complicated since client resource capabilities can vary and appropriate synchronization protocols are required. We will study the applicability of techniques such as controlled multicasting, differential caching, stream merging and selective catching to semi-continuous transmission. Other issues include the study of specific request migration policies, interactivity in semi-continuous transmission and segment based real-time scheduling policies. Finally, we are studying the utility of semi-continuous transmission in mobile and fault tolerant multimedia environments.

Acknowledgements: The authors would like to thank Nilay Doshi, Javid Huseynov and Preethi Srinagesh for help with testing the mechanisms described during the course this work and for several helpful suggestions.

References

[1] K. Almeroth and M. Ammar. The use of multicast delivery to provide a scalable and interactive video-on-demand service. *IEEE Journal on Selected Areas in Communications*, Aug. 1996.

[2] D. Anderson, Y. Osawa, and R. Govindan. A file system for continuous media. *ACM Transactions on Computer Systems*, 10(4):311–337, Nov. 1992.

[3] W. Bolosky, J. Draves, R. Fitzgerald, G. Gibson, M. Jones, S. Levi, N. Myhrvold, and R. Rashid. The tiger video fileserver. In *Proc. of NOSSDAV'96*, 1996.

[4] W. J. Bolosky, R. P. Fitzgerald, and J. R. Douceur. Distributed schedule management in the tiger video fileserver. In *Symposium on Operating Systems Principles*, pages 212–223, 1997.

[5] J. W. C.C. Aggarwal and P.S.Yu. A permuation based pyramid broadcasting scheme for video-on-demand systems. In *Proceedings of IEEE Multimedia Computing Systems 1996*, June 1996.

[6] E. Chang. pidtv: A client based interactive dtv architecture. In *Proceedings of ACM Multimedia '99, Orlando, Florida*, Nov. 1999.

[7] C. Chou, L. Golubchik, and J. Lui. A performance study of dynamic replication techniques in continuous media servers. Technical report, University of Maryland, 1998.

[8] A. Dan, D. Dias, R. Mukherjee, D. Sitaram, and T. R. Buffering and caching in large scale video servers. In *IEEE Compcon*, pages 217–224, 1995.

[9] A. Dan, M. Kienzle, and D. Sitaram. Dynamic policy of segment replication for load-balancing in video-on-demand servers. *ACM Multimedia Systems*, 3(3):93–103, July 1995.

[10] A. Dan and D. Sitaram. An online video placement policy based on bandwidth to space ratio (bsr). In *SIGMOD '95*, pages 376–385, 1995.

[11] L. Gao, J. Kurose, and D. Towsley. Efficient schemes for broadcasting popular videos. In *Proceedings of NOSSDAV'98*, July 1998.

[12] L. Gao and D. Towsley. Supplying instantaneous video-on demand services using controlled multicast. In *Proceedings of IEEE Multimedia Computing Systems 1999*, June 1999.

[13] J. Gemmell and S. Christodoulakis. Principles of delay sensitive multimedia data storage and retrieval. *ACM Transactions on Information Systems*, 10(1):51–90, 1992.

[14] K. Hua and S. Sheu. Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems. In *Proceedings of SIGCOMM'97*, September 1997.

[15] S. Irani and N. Venkatasubramanian. An evaluation of semi-continuous transmission for video servers. Technical Report UCI-ICS 01-47, University of California, Irvine, 2001.

[16] K. Keeton and R. Katz. The evaluation of video layout strategies on a high-bandwidth file server. In *Proceedings of the Fourth International Workshop on Network and Operating System Support for Digital Audio and Video, Lancaster, UK*, pages 237–250, Nov. 1993.

[17] P. Lougher and D. Shepherd. The design of a storage server for continuous media. *The Computer Journal - Special Issue on Distributed Multimedia Systems*, 36(1):32–42, February 1993.

[18] P. G. Mohammad S. Raunak, Prashant Shenoy and K. Ramamritham. Implications of proxy caching for provisioning servers and networks. In *Proceedings of ACM Sigmetrics 2000, Santa Clara, CA*, June 2000.

[19] S. Sahu, P. Shenoy, and D. Towsley. Design considerations for integrated proxy servers. In *Proceedings of NOSSDAV'99*, 1999.

[20] J. Salehi, Z. Zhang, J. Kurose, and D. Towsley. Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing. *IEEE/ACM Transactions on Networking*, September 1998.

[21] S. Sen, L. Gao, J. Rexford, and D. Towsley. Optimal patching schemes for efficient multimedia streaming. In *Proceedings of NOSSDAV'99*, 1999.

[22] S. .Sen, J. Rexford, and D. Towsley. Proxy prefix caching for multimedia streams. In *Proceedings of INFOCOM'99*, March 1999.

[23] S.Sen, D. Towsley, Z. Zhang, and J. Dey. Optimal multicast smoothing of streaming video over an internetwork. In *Proceedings of INFOCOM'99*, March 1999.

[24] M. Thapar and B. Koerner. Architecture for video servers. In *Proceedings of the 43rd Annual NCTA Convention and Exposition, New Orleans, Louisiana*, pages 141–148, 1994.

[25] F. Tobagi, J. Pang, R. Baird, and M. Gang. Streaming raid: A disk storage system for video and audio files. In *Proceedings of ACM Multimedia'93, Anaheim, CA*, pages 393–400, August 1993.

[26] N. Venkatasubramanian and S. Ramanathan. Effective load management for scalable video servers. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS97)*, May 1997.

[27] H. M. Vin and P. V. Rangan. Designing a multi-user hdtv storage server. *IEEE Journal on Selected Areas in Communications*, 11(1):153–164, Jan. 1993.

[28] J. L. Wolf, P. S. Yu, and H. Shachnai. Dasd dancing: A disk load balancing optimization scheme for video-on-demand computer systems. In *Proceedings of ACM SIGMETRICS '95, Performance Evaluation Review*, pages 157–166, May 1995.

[29] P. Yu, M. Chen, and D. Kandlur. Design and analysis of a grouped sweeping scheme for multimedia storage management. *Proceedings of Third International Workshop on Network and Operating System Support for Digital Audio and Video, San Diego*, pages 38–49, November 1992.