Hierarchical Storage Systems for On-Demand Video Servers*†

Shueng-Han Gary Chan   Fouad A. Tobagi‡

Department of Electrical Engineering
Stanford University
Stanford, CA 94305

ABSTRACT

On-demand video servers based on hierarchical storage system offer high-capacity and low-cost video storage. In such a system, video files are stored in the tertiary level and transferred to the secondary level to be displayed. We have studied the architecture and performance of a hierarchical storage system for an on-demand video server. Our objectives are to understand its performance characteristics, and to design such a server to meet specific application requirements.

The secondary level in the hierarchical storage system is characterized by its bandwidth and storage capacity, while the tertiary level is characterized by its number of drives, drive bandwidth and exchange latency. The performance measure we mainly consider here is user delay. We show that, given a certain delay requirement, non-uniform video popularity can reduce the secondary storage requirement tremendously compared with the uniform-popularity case. We also found that secondary bandwidth, secondary storage capacity and tertiary bandwidth can generally be traded with each other to achieve the same average delay performance. However, there is a limit in such "trade-off." Furthermore, our study indicates that increasing a system resource (e.g., secondary bandwidth, secondary storage capacity or tertiary bandwidth) in the hierarchical storage system does not always lead to better performance. Therefore, in designing a server based on hierarchical storage system, balancing various system resources is very important. We finally provide some methodologies in designing a server given a certain delay requirement, by taking into account current storage technologies.

Keywords: video-on-demand, video server, hierarchical storage system, scheduling policies, resource allocation, distributed VOD system, multimedia system

1 Introduction

Most video servers reported upon in the literature are based on magnetic disks. Magnetic disk technology is appropriate for video presentation due to its high throughput and low access delay.[1–3] However, it is still not ideal to store large numbers of video files (e.g., > 1000s video files). This is because magnetic disks are expensive. Storing a single 90-minutes movie ($\simeq$ 1 GB) would cost about 500 dollars. As video files can differ markedly in their popularities and hence some of them are rarely accessed, it is not very cost-effective to store all video files

*SPIE High-Density Data Recording and Retrieval Technologies, SPIE'95, Oct 22-26, Philadelphia, PA.
†This work was supported in part by NSF under grant NCR-9016032, and by Pacific Bell.
‡Also at Starlight Networks, Inc., Mountain View, CA 94043.

on-line regardless of their popularities.

Video servers based on a hierarchical storage system is proposed to provide a cost-effective solution. The system uses tertiary storage, commonly known as library or "jukebox," to store all the video files. Video files which need to be displayed are "staged" onto the secondary level on the demand basis. The secondary level in the hierarchical storage system therefore acts as a "caching" or "staging" platform for video display.[4]

Such a hierarchical storage system offers several advantages. As the media cost of tertiary storage can be orders of magnitude lower than that of secondary storage, storing videos in the tertiary level is much cheaper. Furthermore, as tapes or optical disks offer high storage capacity ($\sim 10 - 100$ GB), and a tertiary library can hold hundreds to thousands of such tapes or disks, a tertiary system offers large storage capacity in the range of terabytes.

A server based on a hierarchical storage system actually involves many system parameters such as secondary bandwidth, secondary storage capacity, tertiary bandwidth, various scheduling policies, and admission control schemes. Designing such a server therefore requires a good understanding of the inter-dependence of these parameters. We have studied the performance of such an on-demand video server. Our objective is to understand its performance characteristics, so that we can design a video server to meet specific application requirements. These application requirements include user delay, streaming capacity, cost-effectiveness, etc.

The paper is organized as follows. In Section 2, we describe the characteristics of Video-on-Demand (VOD) applications. In Section 3, we describe the server architecture and its operation, and specify the operational procedures used in this study. In Section 4, we address the performance and design issues of the hierarchical storage system, based on the results obtained by simulation.

2 VOD Applications Characteristics

Video-on-demand (VOD) refers to the service in which a user, after requesting a video stored in a server, is able to see the video in a short time. VOD encompasses many applications such as movie-on-demand (MOD), distance learning, interactive news, catalogue shopping/home advertising, interactive/distributed video games, etc.[5–7] For a given VOD application, a user first selects a video such as a movie, an advertisement, or a video game. The request is then routed to the relevant video server. At this time, the server may either deny or accept the request. Requests may be denied due to a lack of resources such as streaming capacity, storage capacity and bandwidth, or according to some priority schemes. For the accepted request, if there are enough resources, the user is able to see the video immediately. Otherwise, the user may have to wait until enough resources are available. In any case, so long as the users do not cancel their requests, all the accepted requests will be eventually served. After serving a user, the system updates various information, such as video database, accounting information, and, in case of a video authoring system, writing-back of the modified video files to the relevant server.

In VOD, users usually can interact with the videos through fast-forward, rewind, pause, etc. However, generally they are not allowed to modify the video files, i.e., the video files are read-only. Read-only applications are different from authoring applications in which users can, besides interacting with the videos, change the video contents.

In order to design a practical and useful video server, we need to understand the characteristics of VOD applications. We discuss here two VOD applications characteristics: traffic characteristics and video files characteristics.

Traffic characteristics: Traffic refers to the arrival process of video requests. It has three attributes: arrival rate, size of each arrival, and request holding time.

- Arrival rate — An arrival consists of a single or multiple video requests, each of which asks for the display of a certain video.

  For large number of users (e.g., in the wide-area environment), the arrival process is generally modeled as a Poisson process with rate λ, $\mathcal{P}(\lambda)$, i.e., the inter-arrival time is exponentially distributed with mean $1/\lambda$. For small number of users (e.g., local services), the inter-arrival time may be modeled as exponentially distributed, with its rate depending on the number of "active" users in the system. Here, "active" users are the users who are either seeing or waiting to see their videos. General arrival processes can be modeled as series-parallel combination of exponential processes, such as hyperexponential distribution, Erlangian distribution, Cox-phased networks, etc.[8,9]

- Size of each arrival — The size of an arrival refers to the number of video requests in each arrival. It may be one-at-a-time or multiple requests at a time. Multiple requests per arrival is also called "bulk arrival." Though some applications are characterized by one-at-a-time arrival, some others, such as distance learning or multiuser video games, are more likely characterized by bulk arrivals.

- Request holding time — Request holding time is the total time a user sees or interacts with the video. It may be somewhat random (e.g., as in interactive video games), or relatively deterministic (e.g., as in MOD).

Video files characteristics: The file-related characteristics of a VOD application are streaming bandwidth, size of the files, number of video titles and video popularity.

- Streaming bandwidth — The video encoding scheme (e.g., MPEG-I, MPEG-II, motion-JPEG, etc.) determines the streaming bandwidth of a video. The bandwidth may range from lower than 1 Mbps to more than 10 Mbps.

- Size of the video files — The size of a video file depends on the duration of the video and its data rate. It may range from $\sim$ 10 MB (advertisement) to more than $\sim$ 1 GB (movies). For a VOD application, all the files may not be of the same size. In MOD, for example, the file size is likely to be almost "homogeneous," with each file of about 90 minutes. File size can be more "heterogeneous" in the video games environment, in which the complexity of graphics and rules determines the size of a file. Somewhere between the "extremes" maybe home advertising, in which file size may spread from 30 seconds to 2 minutes.

  Note that the size of a video file does not necessarily relate to request holding time. This is especially true in interactive video games, as some parts of the video can be repeated many times, while some others may not be played at all.

- Number of video titles — The total number of video titles depends on the users of the application. Applications for specialized users is likely to have fewer titles than the applications for general public. Furthermore, an application with a large number of users probably has more titles compared with an application with small number of users.

- Video popularity — Different videos have different access frequency. The popularity of a video is defined as the probability for the video to be accessed or chosen by any incoming request. The popularity index of a video is a number proportional to video popularity or access frequency.

  Uniform video popularity means that all videos are equally likely to be chosen. Non-uniform video popularity is commonly modeled using Zipf or geometric distribution. To explain these models, we first arrange the popularity of all N_v videos in the system in decreasing order. Let the popularity of the ith video be p_i, where $1 \leq i \leq N_v$.

  - Zipf distribution[10–12]: In this model, the popularity of the ith video is proportional to $1/i^{\zeta}$, where ζ is a (real) parameter. Most literature take ζ to be 1.0, though according to actual rental data, $\zeta = 0.271$ may be more likely.[11] Given ζ, we can then express p_i as:

$$p_i = C/i^{\zeta}, \tag{1}$$

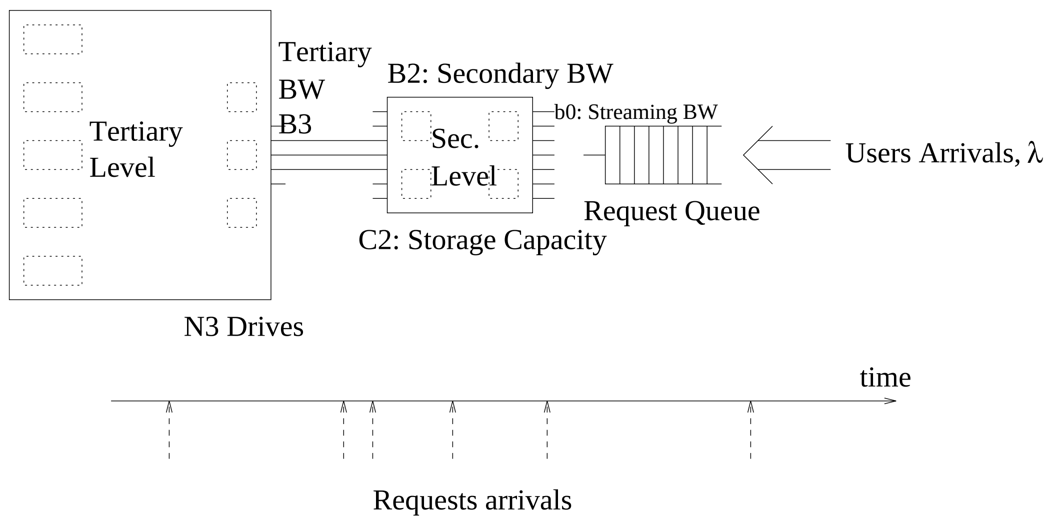

Figure 1: A hierarchical storage system for an on-demand video server. The system consists of three components: request queue, secondary level, and tertiary level.

where

$$C = \left(\sum_{i=1}^{N_v} 1/i^{\zeta} \right)^{-1}. \tag{2}$$

– Geometric distribution[13,14]: In this model, the ratio of p_i to p_{i-1} is equal to a constant called "skew parameter," σ $(0 \leq \sigma \leq 1)$, i.e., $p_i/p_{i-1} = \sigma$. We can therefore express p_i, where $1 \leq i \leq N_v$, as

$$p_i = \frac{(1-\sigma)\sigma^{i-1}}{(1-\sigma^{N_v})}. \tag{3}$$

3 The Architecture and Operation of a Hierarchical Storage System

3.1 System architecture

Figure 1 shows the hierarchical storage system we consider in the paper. The system consists of three components:

- Request queue — The request queue is the waiting room for the users whose requested videos have not been displayed yet. It is characterized by a queue size equal to the maximum number of outstanding requests that the system can handle.
- Secondary level — The secondary level is used for video presentation. It is characterized by a bandwidth, B_2, used for two purposes: i) video streaming, and ii) transferring video files from tertiary level to secondary level, the so-called "video staging." Therefore, the secondary level can be thought as a "staging platform" for video presentation.

  The secondary level has limited storage capacity, C_2. Therefore, not all video titles can be stored in the level at once. Through staging, video files are able to "share" the limited storage space in order to be displayed to the users.

- Tertiary level — All video files are stored in a tertiary repository called tertiary "library" or "jukebox." Tertiary library shelves many removable storage medium, such as tapes or optical disks, which share N_3 tertiary drives. Each drive has a bandwidth, b_3, and hence the maximum tertiary bandwidth is $B_3 = N_3 b_3$. A number of tertiary drives can be configured so that their bandwidths can be used in parallel to deliver a file (e.g., through fine-grained striping[15]).

  In the library, robotic mechanisms swap the tapes/disks in and out of the drives. The time it takes in such exchange is called "exchange latency," T_{ex}.

3.2 System operation

The operation of the hierarchical storage system can be described as follows: A user comes in, requesting a video to be displayed. According to an *Admission Control Scheme*, the request is either accepted or denied. Request admission is controlled and regulated by the request queue, which can only hold a maximum number of requests (and hence the "overflown" requests are lost). The accepted requests are put into the queue to be served. Requests are served according to a *request queue scheduling algorithm.* Requests having their videos displayed are taken out of the request queue.

If the requested video is neither in the secondary level nor being transferred to the secondary level, it has to be staged from the tertiary level in order to be displayed. A request whose video needs to be staged is called "staging request." A staging request may be encountered in the process of request queue scheduling. In this case, we say that we have a "staging event" to be scheduled. For each staging event, a video file has to be chosen to be staged. The video file is chosen according to a *staging-requests scheduling algorithm.* Once the file is chosen, the corresponding tape/disk is loaded into an idle drive.

For the secondary level, video staging actually involves two requirements. Before video staging takes place, not only has the file been already loaded in a tertiary drive, the secondary level must first have some bandwidth available — The bandwidth will be used for staging and hence it is called "staging bandwidth." Then there must be enough storage space for the in-coming file. This space is called "staging room." Therefore, only upon the availability of two resources does staging proceed – i) availability of staging bandwidth, and then ii) availability of staging room.

The secondary level gets its staging room by deleting one of the video blocks in the level. The block is chosen according to a *block-deletion algorithm.* Staging begins after deleting a block. After a certain "staging time," the video block will be staged and the video can be displayed to the user. Note that once a video block has occupied a tertiary drive and started staging, it has to be completely transferred onto the secondary storage before another video block can swap it out of the tertiary drive.

A request is said to be "delayed" if it cannot be served immediately. A delayed request may be lost due to: i) User balking — User leaves the system right after knowing that he/she cannot be served immediately, i.e., requests are lost on their arrivals; or ii) User reneging — The waiting user gives up waiting in the queue and leaves the system.

A video file in the hierarchical storage system can be divided into one or more blocks called "display-blocks." A display-block has to be completely present in the secondary level before it can be displayed. The size of the display-block is determined by the degree of user interaction, e.g., fast-forward, random seek, etc., the system wants to offer. Generally, the larger the display-block, the more freedom a user has in interacting with the video. Therefore, if the display-block size is chosen to be as large as the file size, user would have complete control with the entire video. One disadvantage of picking a large display-block size, however, is that a user may have to wait longer for the display-block to be completely staged before the user can see the portion of the video.

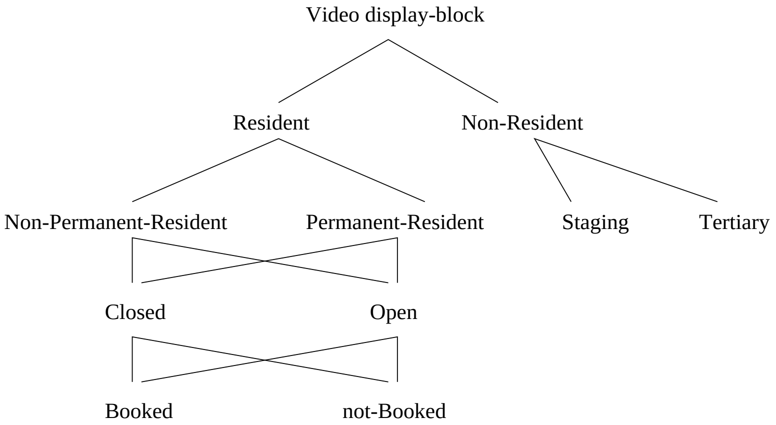

Figure 2: Classification of video display-blocks in the hierarchy.

Video display-blocks in the hierarchical storage system can be classified according to Figure 2. There are two types of video display-blocks: i) Non-resident display-block, and ii) Resident display-block.

Non-resident display-blocks are the display-blocks not entirely in the secondary level. There are two types of non-resident blocks: i) "Staging" blocks, blocks being staged and hence only partially present in the secondary level; or ii) "Tertiary" blocks, blocks entirely absent in the secondary level.

Resident display-blocks are the blocks entirely present in the secondary level. Such blocks can be divided into two categories: i) Permanent-resident blocks, blocks which reside permanently in the secondary level and hence can never be deleted; and ii) Non-permanent-resident blocks, blocks which may be deleted, and hence whose space can be re-used for staging purpose.

Resident display-blocks can also be classified as "open" or "closed." When a video is being displayed, the corresponding display-block is said to be "open." Analogously, when a video block is not being displayed, it is said to be "closed."

Besides "open" or "closed," a resident display-block can also be called "booked" or "not-booked." A "booked" video block is a display-block which is asked by one or more requests in the request queue, while a "not-booked" video block is a display-block which is not asked by any request in the request queue.

In the hierarchical storage system, there are two types of requests — hit requests and miss requests. A hit request is a request which asks for a resident video display-block, while a miss request is a request which asks for a non-resident video display-block. There are two types of misses: if the block requested is a staging block, we call the miss a "staging miss." On the other hand, if the block requested is a tertiary block, we call the miss an "absence miss." Obviously, only video files corresponding to absence misses are needed to be staged.

A request is said to be "blocked" if it cannot be served immediately. A video request is defined to be "served" if: i) the requested video is being displayed to the user; or ii) the requested video block is currently in the process of being staged from the tertiary level onto the secondary level.

3.3 Operational procedures

We now elaborate further on the system operation and specify the operational procedures chosen in our study. We will consider the case where the size of video display-block is equal to the size of video file, so that a user has complete interaction with the video from its beginning to the end.

In designing the hierarchical storage system, we can generally divide the operational procedures into two categories:

- Procedures related to request handling — These include i) Admission control scheme; ii) Request queue scheduling algorithm; and iii) Staging-requests scheduling algorithm;
- Procedure related to the secondary level — Block-deletion algorithm.

3.3.1 Request handling procedures

- Admission control schemes — In this paper, we consider Blocked-call-held (BCH) algorithm, in which we accept in-discriminantly all the incoming requests and put them into the request queue to be served. This policy therefore assumes infinite request queue size.
- Request queue scheduling — Secondary bandwidth can be either "split" or "shared" between staging and streaming. In the "split" policy, a certain amount of secondary bandwidth is permanently laid aside for either staging or streaming purpose. The bandwidth will be wasted if it is not used for that purpose. On the other hand, in the "shared" policy, the secondary bandwidth is not split for either purpose. Staging and streaming can freely use whatever bandwidth is available. In this paper, we will assume "shared" policy. We further assume in our study that the tertiary library is capable of delivering bandwidth equal to $N_3 b_3 = B_3$, i.e., the bandwidths of all drives can be used in parallel to deliver a file (and hence no more than one file can be staged at one time).

  Each request is served by assigning a certain secondary bandwidth to it. The assignment can be either a streaming bandwidth or a staging bandwidth. It is streaming bandwidth if a video is to be displayed to the user; otherwise, it is staging bandwidth, in which case the requested video block is to be staged. At a given point in time, if B_2^{stg} and B_2^{strm} are the total staging and streaming bandwidth used in the secondary level respectively, and B_3^{stg} is the total bandwidth used in the tertiary level in transferring video blocks to the secondary level, then the following relationships must hold:

$$\begin{aligned} B_2^{stg} + B_2^{strm} &\leq B_2, & (4)\\ B_3^{stg} &\leq B_3, & (5)\\ B_3^{stg} &= B_2^{stg}. & (6) \end{aligned}$$

  The request queue scheduling algorithm is called whenever there is a new incoming request, or a status change in the secondary bandwidth. The change may be caused by two circumstances: i) a user currently holding a stream leaves the system, hence freeing a streaming bandwidth; or ii) a video block has just finished staging onto secondary level, hence freeing the staging bandwidth.

  In this paper, we consider First-Come-First-Serve (FCFS) request queue scheduling algorithm. This algorithm tries to assign secondary bandwidth according to the arrival order of the requests, until all the secondary bandwidth is used up. Note that this is not a strictly-FCFS algorithm, i.e., requests may not always be served in strict order, as this algorithm puts bandwidth utilization at a higher priority than order preservation. This can be illustrated by two cases: i) Due to the limited number of drives in the tertiary level, all tertiary drives may be busy at one time. If this happens, staging requests cannot be served anymore; ii) In preparing staging room for an in-coming block, secondary level may not be able to find a deletable block. As a result, staging cannot proceed and hence no more staging request can be served. In both cases above, our FCFS algorithm "skips" all the staging requests in the queue and tries to serve only the hit requests, even the hit requests may arrive later than the staging requests.

- Staging-requests scheduling — We consider First-Come-First-Serve (FCFS) staging-requests scheduling algorithm, which stages requests according to their order of arrival.

3.3.2 Secondary level procedures

- Block-deletion algorithm — The secondary level obtains its staging room by deleting one of its display-blocks according to a block-deletion algorithm. Block-deletion consists of two steps:

  1. Selection of deletable blocks: From all the non-permanent-resident blocks, the secondary level selects a set of deletable candidates according to a *deletable-blocks-selection* algorithm.
  2. Replacement of a deletable block: Among all the deletable blocks, the secondary level selects one to be deleted according to a *block-replacement* algorithm. In case of a tie, one of the blocks is deleted according to *tie-block resolution.*

  In this paper, we consider Closed-not-Booked-Deletable (CnBD) algorithm as our deletable-blocks-selection algorithm. According to this algorithm, a video display-block is deletable only if it is closed and not being requested by any request in the request queue (i.e., not booked). This algorithm therefore tries to decrease the number of staging, and hence tries to minimize the staging overhead (in terms of staging time, staging bandwidth and exchange latency).

  Note that according to this algorithm, a staging request may not be served even though we have an idle tertiary drive and available secondary bandwidth. This is because it may happen that all the resident blocks are open and/or booked at that time, and hence a deletable block cannot be found. This "all-block-undeletable" phenomenon may persist for a while before a deletable block is identified. This waiting time for a deletable block can be rather long under high traffic condition, especially with long request holding time and small secondary storage.

  For a block-replacement algorithm, we will consider Least-Recently-Used (LRU) algorithm. A block is being "used" if it is being open. The algorithm deletes the block which has not been used or accessed (i.e., has been closed) for the longest time. This is based on the observation that least-recently-used block is most unlikely to be accessed in the near future. In case of tie-blocks, we delete the block we found first.

4 Performance Evaluation and System Design

4.1 Performance measure

The performance we mainly consider here is the "start-up delay," D_{st}, defined as the waiting time from the moment when a user initially submits a video request until the user begins to see the video. In this study, we focus on average start-up delay over all requests in the steady state, $\overline{D}_{st}$. We concern ourselves with the trade-off and sensitivity issues in designing the hierarchical storage system. Trade-off study is the study of the inter-dependence of two or more design choices to see how one parameter can "trade" with the others in order to achieve the same performance. Sensitivity study is the study of the influence of one or more design choices on a performance measure. The study allows one to understand the marginal benefit in altering the value of a design choice.

4.2 Simulation results

We have investigated the performance of the hierarchical storage system based on the following specifications:

- Traffic characteristics:
  - Poisson arrivals with rate λ (arrivals/hr) and one request per arrival;
  - Constant request holding time, $T_h = 90$ minutes;
  - No user balking or reneging, i.e., users, once requesting their videos, wait until their videos are displayed.
- Video characteristics:
  - Constant streaming bandwidth, b_0, for all video files at all times: $b_0 = 1.5$ Mbps;
  - Homogeneous file size, i.e., $C_f = 1$ GB for all video files;
  - No permanent-resident videos; and
  - Unmodifiable files, i.e., we assume read-only applications.

We have implemented an event-driven simulation to study the performance of the hierarchical storage system. We collect statistics of the user delay after the transient response has passed. Through "batch mean procedures,"[16,17] most of our 95% confidence intervals fall within 5% of the simulation values, with the majority falling within 3%. For plot clarity, we do not show the confidence intervals here.

We assume that tertiary exchange latency is small compared with staging time, and hence it is ignored. We present below the sensitivity and trade-off issues of the hierarchical storage system.

4.2.1 Sensitivity study:

In this section, we will consider uniform video popularity, i.e., $p_i = 1/N_v$ for $1 \leq i \leq N_v$, where N_v is the total number of videos in the server. In Figure 3, we consider a hierarchical storage system with $B_2 = 20$ MB/s, $B_3 = 10$ MB/s, and $N_v = 500$, with three values of C_2: 50 GB, 100 GB, and 200 GB. We plot here the total secondary bandwidth utilization as the traffic, λ, is increased. Total secondary bandwidth utilization, ρ_2, is defined as the time average of the fraction of B_2 used to serve requests. Therefore, ρ_2 indicates how well B_2 is used in both streaming and staging, i.e., if $\rho_2 = 100\%$, no B_2 is idle. Also shown here are the bandwidth utilization for streaming, ρ_2^{strm}, and the bandwidth utilization for staging, ρ_2^{stg}, which are defined as the time average of the fraction of B_2 used for streaming and staging respectively. Obviously, we have:

$$\rho_2 = \rho_2^{strm} + \rho_2^{stg}. \tag{7}$$

As traffic is increased, ρ_2 increases rather "linearly" until it reaches 100%. When ρ_2 approaches 100% (and hence ρ_2^{stg} achieves its maximum), we observe from our simulation traces that the queue for staging requests quickly develops and delay increases rapidly. Figure 3 shows that $\rho_2 = 1$ does not necessarily imply instability, i.e., infinite delay. In fact, according to our simulation data, instability occurs at ~ 70 requests/hr. λ can be increased much beyond the point when $\rho_2 = 1$ before the server becomes unstable. This is because staging bandwidth can be decreased (and hence the staging time is correspondingly increased) in order to accommodate more streaming traffic.

Figure 4 shows the average start-up delay, $\overline{D}_{st}$, with increasing traffic intensity. We again show three values of C_2: 50 GB, 100 GB, and 200 GB, with $B_2 = 20$ MB/s, $B_3 = 10$ MB/s, and $N_v = 500$. We observe from

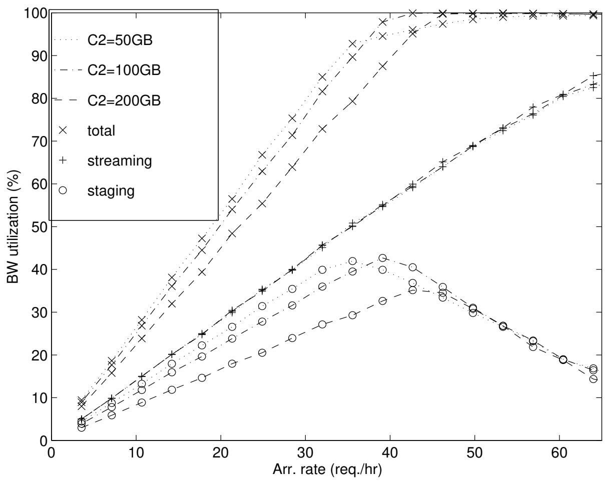

Figure 3: Total secondary bandwidth utilization, ρ_2, along with the utilizations for streaming (ρ_2^{strm}) and staging (ρ_2^{stg}), as λ is increased. We show three values of C_2. We see that $\rho_2 \simeq 100\%$ when $\lambda > \sim 40$ requests/hr. ($B_2 = 20$ MB/s, $B_3 = 10$ MB/s, $N_v = 500$, and uniform video popularity.)

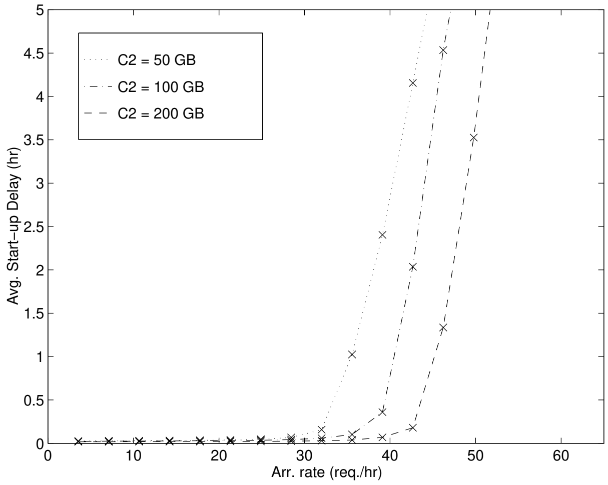

Figure 4: $\overline{D}_{st}$ as λ is increased. The onset of high delay indicates tertiary level "congestion." To design a low delay server, we need $\lambda \leq 38$ requests/hr for $C_2 = 100$ GB. ($B_2 = 20$ MB/s, $B_3 = 10$ MB/s, $N_v = 500$, and uniform video popularity.)

the figure that the "knee" of each curve occurs when $\rho_2 \simeq 1$. At this point, the queue for the staging requests becomes long, and hence the tertiary level becomes "congested." Therefore, in designing a server with low delay performance, we should limit the traffic to $\lambda \leq 32$ requests/hr, 36 requests/hr and 43 requests/hr for $C_2 = 50$ GB, 100 GB and 200 GB respectively. The increase in C_2 is large in order to accommodate a relatively small increase in traffic rate.

Figure 5 plots the average start-up delay, $\overline{D}_{st}$ as secondary bandwidth increases. We assume here $\lambda = 20$ req./hr, $B_3 = 10$ MB/s, and $N_v = 500$. We show three values of secondary capacity, $C_2 = 50$ GB, 100 GB, and 200 GB. All curves eventually settle to their limiting values when $B_2 > \sim 16$ MB/s. The vertical asymptote of each curve indicates the minimum bandwidth required to handle the traffic rate (20 requests/hr).

We show in Figure 6 the sensitivity of $\overline{D}_{st}$ when secondary storage capacity increases, with arrival rates $\lambda = 17.78$ requests/hr, 35.56 requests/hr, and 42.67 requests/hr. All curves monotonically decreases to zero. For $\lambda = 17.78$ requests/hr, $\overline{D}_{st}$ already achieves its low value at $C_2 > \sim 50$ GB. Therefore, to handle such traffic, a small secondary storage ($C_2 = 50$ GB) is required. With $\lambda = 35.56$ requests/hr and $C_2 < \sim 70$ GB, there is frequent "all-blocks-undeletable" phenomenon and the queue for staging requests is long, and hence the delay is high. Therefore, to accommodate $\lambda = 35.56$ requests/hr, we have to double C_2 to 100 GB. For a slightly higher traffic rate such as $\lambda = 42.67$ requests/hr, C_2 has to be ≥ 150 GB in order to achieve a reasonably low average delay. This is another 50 GB increase in storage requirement in order to accommodate a small increase in traffic.

Figure 7 shows the influence of tertiary bandwidth on delay, with $B_2 = 10$ MB/s, 20 MB/s, and 30 MB/s. We assume $\lambda = 20$ req./hr, $C_2 = 100$ GB, and $N_v = 500$. With $B_2 = 10$ MB/s, average start-up delay settles to its limiting value at $B_3 \approx 4.5$ MB/s. We observe from our simulation traces that when $B_3 < \approx 4.5$ MB/s, each staging event generally makes full use of B_3. Further increase in B_3 beyond 4.5 MB/s does not lead to higher performance, because the secondary level does not have enough bandwidth to take advantage of it. All curves

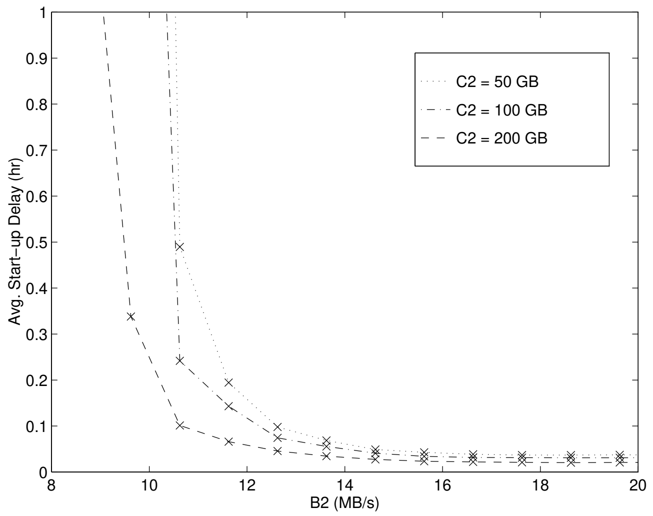

Figure 5: Sensitivity of $\overline{D}_{st}$ with respect to B_2, with three values of C_2. Each line shows a vertical asymptote and a limiting value (at $B_2 > \sim 16$ MB/s). ($\lambda = 20$ req./hr, $B_3 = 10$ MB/s, $N_v = 500$, and uniform video popularity.)

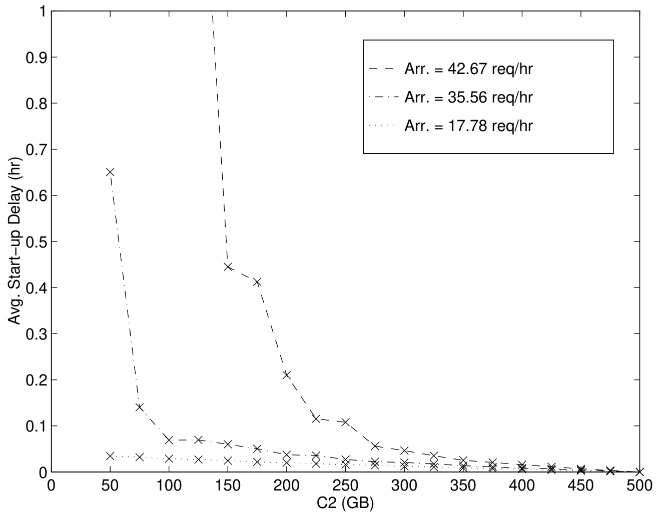

Figure 6: Sensitivity of $\overline{D}_{st}$ with respect to C_2, with three different λ's. The delay approaches zero as C_2 increases. ($B_2 = 20$ MB/s, $B_3 = 10$ MB/s, $N_v = 500$, and uniform video popularity.)

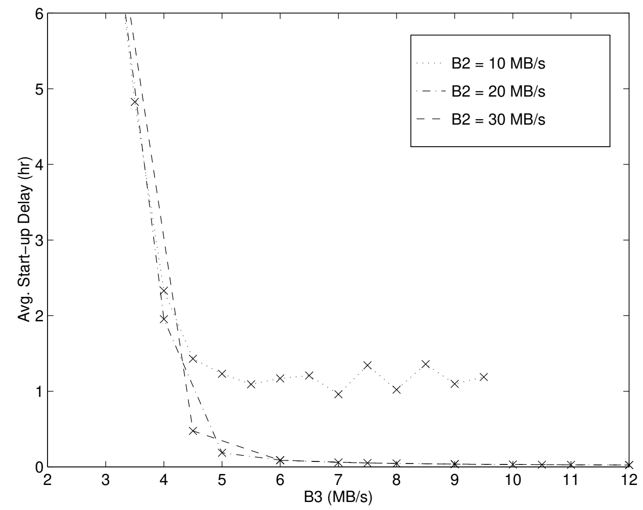

Figure 7: Sensitivity of $\overline{D}_{st}$ with respect to B_3, with three values of B_2. All lines achieve relatively low delay at $B_3 \simeq 5$ MB/s. ($\lambda = 20$ req./hr, $C_2 = 100$ GB, $N_v = 500$, and uniform video popularity.)

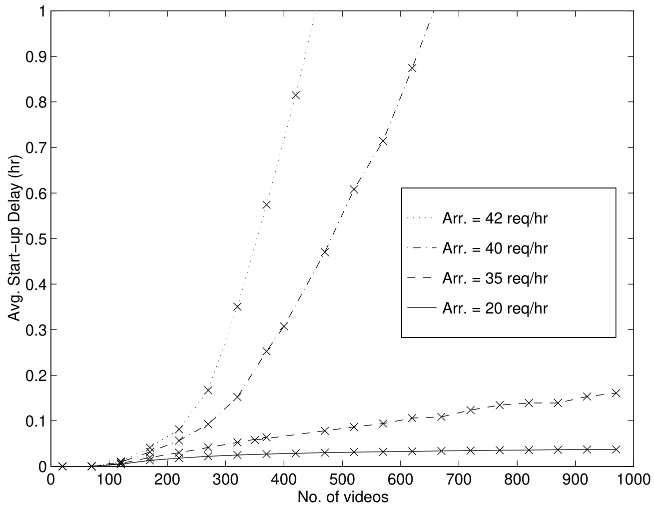

Figure 8: Sensitivity of $\overline{D}_{st}$ with respect to N_v. The delay performance of the server scales well at low ρ_2 ($\lambda = 20$ req./hr), but not so at high ρ_2 ($\lambda = 42$ req./hr). ($B_2 = 20$ MB/s, $C_2 = 100$ GB, $B_3 = 10$ MB/s, $N_v = 500$, and uniform video popularity.)

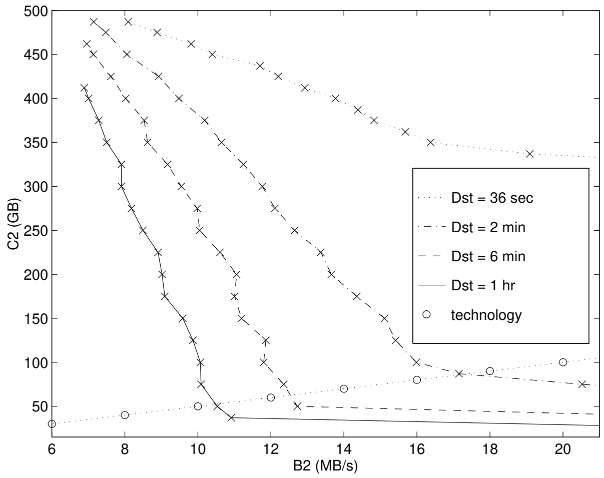

Figure 9: Trade-off between C_2 and B_2, with four values of $\overline{D}_{st}$. All delay lines show an initial rather linear "trade-off" region followed by a horizontal asymptote (no trade-off). ($\lambda = 20$ req./hr, $B_3 = 10$ MB/s, $N_v = 500$, and uniform video popularity.)

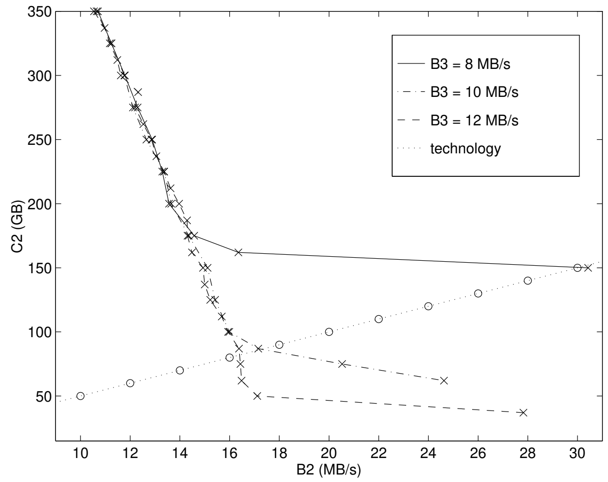

Figure 10: The design of a hierarchical storage system with $\overline{D}_{st} = 2$ min. For $B_3 = 10$ MB/s, we need $C_2 = 90$ GB and $B_2 = 18$ MB/s. ($\lambda = 20$ req./hr, $N_v = 500$, and uniform video popularity.)

show "knees" at $B_3 \approx 5$ MB/s, where they achieves relatively low delay.

We show in Figure 8 the delay performance of the on-demand video server as the number of video titles increases, assuming $B_2 = 20$ MB/s, $C_2 = 100$ GB, $B_3 = 10$ MB/s, and $N_v = 500$. We show four arrival rates, $\lambda = 20$ requests/hr, 35 requests/hr, 40 requests/hr and 42 requests/hr, which are equivalent to $\rho_2 \simeq 50\%$, 85%, 99% and 100% respectively. From the figure, we see that the server does not scale well in high secondary bandwidth utilization (i.e., $\rho_2 \geq 0.9$). Designing a server with expanding video titles therefore has to consider its bandwidth utilization carefully.

4.2.2 Trade-off study:

Figure 9 shows the trade-off between secondary storage capacity and secondary bandwidth, given a constant average start-up delay. The average delay shown are: $\overline{D}_{st} = 36$ sec., 2 min., 6 min., and 1 hr. The system parameters we consider here are $\lambda = 20$ req./hr, $B_3 = 10$ MB/s, $N_v = 500$, and uniform video popularity.

Each curve exhibits an initial rather linear region, showing flexible trade-off between C_2 and B_2, followed by a horizontal asymptote. The on-set of the asymptote indicates the limit that secondary bandwidth can be traded with secondary storage. We see that, to achieve a constant average delay, a higher secondary bandwidth can generally be traded off with a smaller storage capacity up to a certain point. Beyond this point, no such "trade-off" is possible.

For the 1-hr delay contour, we observe from our simulation traces that when $C_2 < \sim 30$ GB (30 videos), there is a long queue of staging-requests. This long "staging queue" is caused by frequent "all-blocks-undeletable" phenomenon and long waiting time (in the order of minutes) for a deletable block. As a result, staging of a block

# Magnetic Disks	GB/Disk
10	10
20	5
40	2.5

Table 1: Number of magnetic disks and their required capacities in order to satisfy an average delay requirement of 2 minutes. B_2 is chosen to be 16 MB/s.

takes a long time, and hence the staging queue is long.

According to Figure 9, we can find a set of (B_2, C_2) in order to satisfy a certain average delay requirement. For example, given a delay requirement of 2 min. and $B_2 = 16$ MB/s, we require $C_2 = 100$ GB. This can be satisfied by using different number of magnetic disks and disk capacity, as shown in Table 1.

Figure 9 shows that, for a given average delay requirement, an efficient system would be most probably operated at the knee of the curve, (B_2^*, C_2^*). This is because any B_2 higher than B_2^* cannot "trade with" less storage capacity, while, due to the rather steep linear region, there must be a great marginal storage benefit for any B_2 less than B_2^*. The latter point can be illustrated as follows: Suppose we have a delay requirement of 2 minutes (Hence, its knee is situated at $B_2^* = 16$ MB/s and $C_2^* = 100$ GB). By decreasing B_2 slightly to 15 MB/s, we need $C_2 = 140$ GB, a significant 40% increase in storage capacity. If we consider $\overline{D}_{st} = 3$ minutes, when B_2 is decreased from 13 MB/s to 12 MB/s, the storage requirement has to be doubled from 50 GB to 100 GB. This is a significant change in storage capacity for a slight change (less than than 10%) in B_2.

Note that a magnetic disk, besides its capacity C_{dsk}, comes with certain bandwidth B_{dsk}. Currently, the maximum capacity of a magnetic disk is $\simeq 10$ GB/disk. Several magnetic disks can be "pooled" together to give a higher total capacity, while at the same time increase the total bandwidth. If we take the effective bandwidth of a disk to be $\simeq 2$ MB/s, we have the following "disk technology" relationship:

$$C_2 = \left(\frac{B_2}{B_{dsk}}\right) C_{dsk} \tag{8}$$

$$\leq \left(\frac{B_2}{2}\right) 10. \tag{9}$$

Equation (9) is shown as the "technology" line in Figure 9. The region below the "technology" line is the current "technologically feasible" region. We show in the Figure 10 specifically the contours corresponding $\overline{D}_{st} = 2$ min., with three different values of B_3: 8 MB/s, 10 MB/s and 12 MB/s, assuming $\lambda = 20$ req./hr, $N_v = 500$, and uniform video popularity. From the interception between the "disk technology" line and the delay contour, we can deduce the configurations of the secondary level in order to satisfy the 2 min. average delay requirement. The configurations is shown in Table 2. As the "technology" line intercepts the delay contour of $B_3 = 8$ MB/s well into the horizontal asymptote region, we have a lot of excess B_2 in such configuration. We see that the requirements of both C_2 and B_2 increased by more than 50% as B_3 decreases from 10 MB/s to 8 MB/s.

Figure 11 shows a similar plot as Figure 10 ($\lambda = 20$ req./hr, $N_v = 500$, and uniform video popularity), but with $\overline{D}_{st} = 3$ min. Now, we have a different configuration as shown in Table 3. We see from the table that, for $\overline{D}_{st} = 3$ min., there is no benefit in increasing B_3 beyond 10 MB/s.

Figure 12 shows the influence of video popularity on the trade-off between C_2 and B_2, with $\lambda = 20$ req./hr, $B_3 = 10$ MB/s, and $N_v = 500$. We show four values of average delay: $\overline{D}_{st} = 36$ sec., 2 min., 6 min., and 1 hr. We have used the geometric video popularity model given in Section 2. We assume 80/20 popularity model,

B_3 (MB/s)	C_2 (GB)	B_2 (MB/s)	(# disks)	GB/disk
8	150	30	15	10
10	90	18	9	10
12	80	16	8	10

Table 2: Design values of the hierarchical storage system in order to satisfy a 2 min. average delay requirement, taking into account current magnetic disk technology.

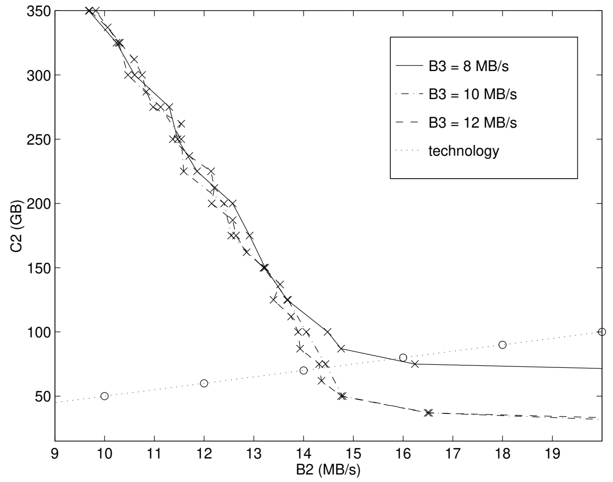

Figure 11: The design of a hierarchical storage system with $\overline{D}_{st} = 3$ min. For $B_3 = 10$ MB/s, we need $C_2 = 40$ GB and $B_2 = 16$ MB/s. ($\lambda = 20$ req./hr, $N_v = 500$, and uniform video popularity.)

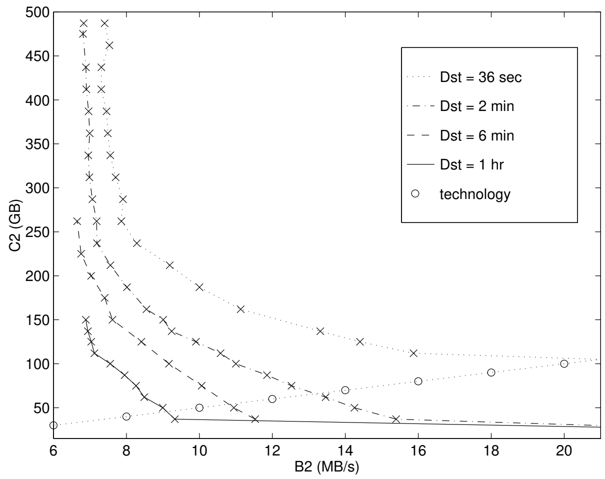

Figure 12: Trade-off between C_2 and B_2, with 80/20 video popularity model. We show four values of $\overline{D}_{st}$. Each curve shows an initial linear region, followed by a horizontal asymptote. Given a certain average delay, we see a great decrease in storage and/or bandwidth requirement. ($\lambda = 20$ req./hr, $B_3 = 10$ MB/s, and $N_v = 500$.)

B_3 (MB/s)	C_2 (GB)	B_2 (MB/s)	(# disks)	GB/disk
8	80	16	8	10
10	40	16	8	5
12	40	16	8	5

Table 3: Parameters of the secondary level in order to satisfy a 3 min. average delay requirement.

$\overline{D}_{st}$	B_2 (MB/s)	C_2 for unif. pop. (GB)	C_2 for 80/20 pop. (GB)
36 sec.	18	350	100
2 min.	16	100	30

Table 4: Influence of video popularity on the storage requirement for a certain $\overline{D}_{st}$ requirement.

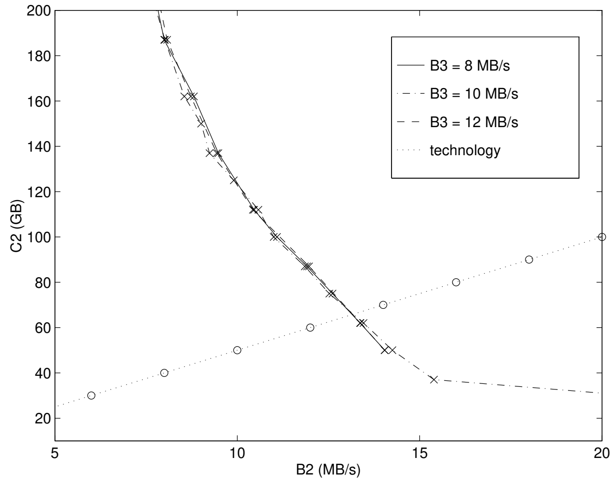

Figure 13: The design of a hierarchical storage system with $\overline{D}_{st} = 2$ min. For all the values of B_3 considered, we need $C_2 = 50$ GB and $B_2 = 14$ MB/s. ($\lambda = 20$ req./hr, $N_v = 500$, and 80/20 video popularity.)

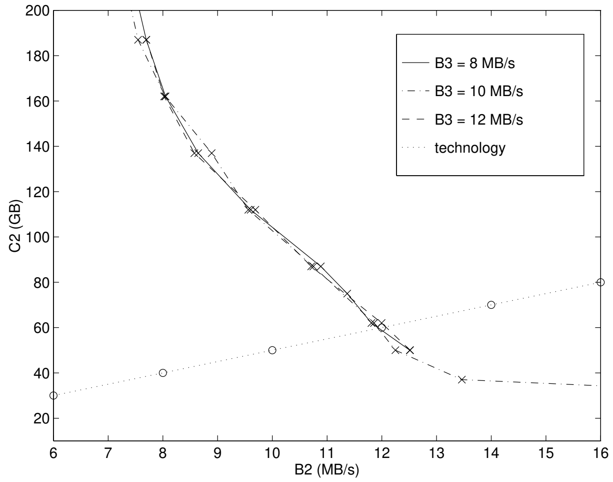

Figure 14: The design of a hierarchical storage system with $\overline{D}_{st} = 3$ min. For the values of B_3 shown here, we need $C_2 = 60$ GB and $B_2 = 12$ MB/s. ($\lambda = 20$ req./hr, $N_v = 500$, and 80/20 video popularity.)

i.e., 80% of the requests ask for 20% of the videos. This gives $\sigma = 0.984$ in Equation (3). The trade-off between C_2 and B_2 in Figure 12 is similar to that of Figure 9. Given a certain average delay performance, non-uniform video popularity markedly decreases the storage and/or bandwidth requirements. This is illustrated in Table 4. When $B_2 = 18$ MB/s, $\overline{D}_{st} = 36$ sec. and uniform video popularity, we need $C_2 = 350$ GB, which is equivalent to 70% resident videos! However, such big storage requirement is decreased tremendously to only $C_2 = 100$ GB if the popularity of the video file is non-uniform. If we require $\overline{D}_{st} = 2$ min., we just need $C_2 = 30$ GB, less than one-third of that required with the uniform-popularity case. Note that 30 GB means that only 6% of the videos are resident in the secondary level!

Figures 13 and 14 show the design of the hierarchical storage storage system for the non-uniform video popularity. Here, we show $\overline{D}_{st} = 2$ and 3 minutes respectively (with $\lambda = 20$ req./hr, and $N_v = 500$). We see that for $\overline{D}_{st} = 2$ minutes, we only need $B_2 = 14$ MB/s and $C_2 = 50$ GB for 8 MB/s $\leq B_3 \leq$ 12 MB/s. Comparing with table 2, this is a great reduction in both B_2 and C_2. For $\overline{D}_{st} = 2$ minutes, we just need $B_2 = 12$ MB/s and $C_2 = 60$ GB. Non-uniform video popularity can therefore reduce the requirement in B_2 and/or C_2.

In Figure 15, we show the trade-off between secondary capacity and tertiary bandwidth, with $\overline{D}_{st} = 36$ sec., 2 min., 6 min., and 1 hr. We have used $\lambda = 20$ req./hr, $B_2 = 20$ MB/s, $N_v = 500$, and uniform video popularity. The trend of each curve is similar to Figure 9: a linear region followed by a horizontal asymptote (settle with $B_2 > \sim 14$ MB/s). The linear region indicates that C_2 can flexibly trade with B_3, while the horizontal asymptote

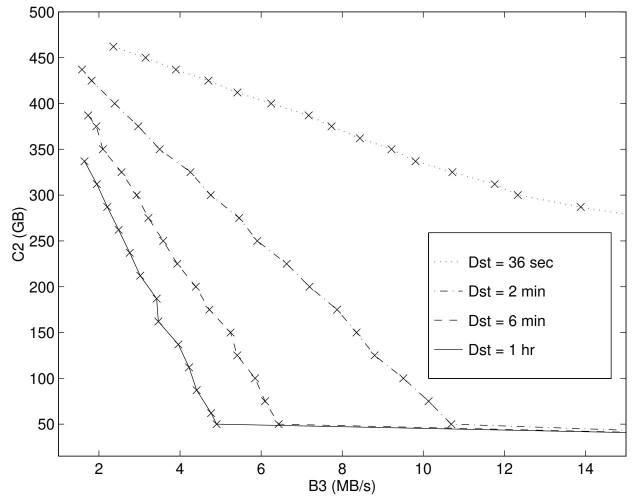

Figure 15: Trade-off between C_2 and B_3, with four values of $\overline{D}_{st}$. All lines show an initial rather linear region (indicating flexible trade-off), followed by a horizontal asymptote (indicating no trade-off). ($\lambda = 20$ req./hr, $B_2 = 20$ MB/s, $N_v = 500$, and uniform video popularity.)

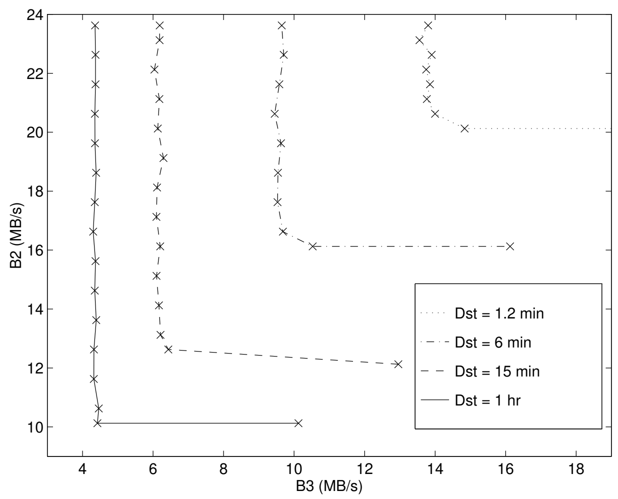

Figure 16: Trade-off between B_2 and B_3, with four values of $\overline{D}_{st}$. All lines show a rather sharp vertical and horizontal asymptotes, indicating that B_2 cannot flexibly trade with B_3. ($\lambda = 20$ req./hr, $C_2 = 100$ GB, $N_v = 500$, and uniform video popularity.)

again indicates the limit tertiary bandwidth can trade with the secondary storage capacity. Therefore, for a given delay of, say, 2 min., it is probably not worthwhile to increase tertiary bandwidth beyond 11 MB/s. As in Figure 9, we observe low performance with $C_2 < \sim 30$ GB. This is again due to frequent "all-blocks-undeletable" phenomenon and long staging queue.

Besides trading off capacity with bandwidth, we think it may be possible to trade B_2 with B_3. After all, if we increase B_2, we may serve more streams at a time, and hence decrease the delay. On the other hand, if we decrease B_3 at the same time, the staging process will be slower and hence the delay is likely to increase. Therefore, there may be some trade-offs between B_2 and B_3 in order to give a certain average delay. Figure 16 shows such trade-off between the bandwidths, with delay ranging from 1.2 min. to 1 hr. We have used $\lambda = 20$ req./hr, $C_2 = 100$ GB, $N_v = 500$, and uniform video popularity. Surprisingly, there is not much trade-off between the bandwidths for the parameters we consider here. We see that there exists only a small region around the "knee" of the graph that tertiary and secondary bandwidth can trade with each other to achieve a certain delay performance. As the delay is increased, the flexibility in such trade-off decreases. Figures 9, 15, and 16 show that B_2 and B_3 can generally be flexibly traded with C_2 but not with each other.

5 Conclusions

A video server based on a hierarchical storage system is proposed to offer high-capacity and low-cost video storage. In this paper, we detail the system architecture and operation of such a server, and present its performance characteristics and design methodologies. These will provide guidelines in designing such a server to meet specific application requirements.

The hierarchical storage system consists of three components: 1) a request queue for admission control and request scheduling; 2) a secondary level for video presentation; and 3) a tertiary level for video staging. We have given the operation characteristics of such hierarchical storage system. These characteristics include hit and miss requests and the types of video blocks in the hierarchy.

The design choices of a hierarchical storage system can be divided into architectural parameters and operational procedures. Architectural parameters include request queue size, bandwidth and storage capacity in the secondary level, bandwidth of the tertiary level, etc. Operational procedures deal with various scheduling policies. We have studied a server based on First-Come-First-Serve scheduling algorithm, Least-Recently-Used block-replacement policy and Blocked-Call-Held admission control.

Using an event-driven simulation, we have addressed the trade-off and sensitivity issues of such a hierarchical storage system. The performance measure we mainly consider here is user delay. We found that the performance of the hierarchical storage system scales well in the number of video titles only under low bandwidth utilization. We have also studied the influence of video popularity on the performance of such a server. According to our model, non-uniform video popularity is able to reduce the storage requirement of the secondary level tremendously compared with the uniform-popularity case.

Our results also show that, to achieve a certain delay performance, secondary bandwidth, secondary storage capacity and tertiary bandwidth can generally be traded off with each other until a certain point is reached. There is no more beneficial "trade-off" beyond this point. Our study indicates that there may not be any marginal benefit in increasing a parameter (e.g., secondary bandwidth, secondary storage capacity and tertiary bandwidth) beyond a certain value. In designing an on-demand video server based on the hierarchical storage system, it is therefore very important to carefully balance various system resources. We finally provide some design methodologies in how such a server can be designed with a certain delay constraint. These methodologies take into account current storage technology and various trade-off conditions in the hierarchical storage system.

We are currently investigating the influence of scheduling policies and admission control on the performance of the system. Some analytic models are also being developed. Through these studies, we are able to provide comprehensive guidelines in designing an on-demand video server based on hierarchical storage system.

6 REFERENCES

[1] F. Tobagi and J. Pang, "StarWorks^{TM} – A Video Applications Server," in *Compcon*, (San Fransisco, CA), February 22-25 1993.

[2] F. Tobagi, J. Pang, R. Baird, and M. Gang, "Streaming RAID^{TM} – A Disk Array Management System for Video Files," in *First ACM International Conference on Multimedia*, (Anaheim, CA), August 1-6 1993.

[3] R. Lauriston, "Serving up Video," *Newmedia*, pp. 52–57, November 1993.

[4] P. N. Misra, "Capacity analysis of the Mass Storage System," *IBM System Journal*, vol. 20, no. 3, pp. 346–364, 1981.

[5] T. Little and D. Venkatesh, "Prospects for Interactive Video-on-Demand," *IEEE Multimedia Magazine*, pp. 14–24, Fall 1994.

[6] F. A. Tobagi, "Distance Learning with Digital Video," *IEEE Multimedia Magazine*, pp. 90–94, Spring 1995.

[7] B. Furht, D. Kalra, F. L. Kitson, A. A. Rodriguez, and W. E. Wall, "Design Issues for Interactive Television Systems," *IEEE Computer Magazine*, pp. 25–38, May 1995.

[8] H. Kobayashi, *Modeling and Analysis: An Introduction to System Performance Evaluation Methodology.* Addison Wesley, 1981.

[9] L. Kleinrock, *Queueing Systems: Theory*, vol. 1. Wiley Interscience, 1975.

[10] C. C. Bisdikian and B. V. Patel, "Issues on Movie Allocation in Distributed Video-on-Demand Systems," in *ICC'95*, pp. 250–255, 1995.

[11] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling Policies for an On-Demand Video Server with Batching," in *ACM Multimedia'94*, pp. 15–23, 1994.

[12] F. Schaffa and J.-P. Nussbaumer, "On Bandwidth and Storage Tradeoffs in Multimedia Distribution Networks," in *Infocom'95*, pp. 1020–1026, 1995.

[13] L. D. Giovanni, A. M. Langellotti, L. M. Patitucci, and L. Petrini, "Dimensioning of Hierarchical Storage for Video on Demand Services," in *ICC'94*, pp. 1739–1743, 1994.

[14] Y. N. Doğanata and A. N. Tantawi, "Making a Cost-Effective Video Server," *IEEE Multimedia Magazine*, pp. 22–30, Winter 1994.

[15] A. Drapeau and R. Katz, "Striped Tape Arrays," in *IEEE Symposium on Mass Storage Systems*, pp. 257–265, 1993.

[16] P. Bratley, B. L. Fox, and L. E. Schrage, *A Guide to Simulation.* New York: Springer-Verlag, 2nd ed., 1987.

[17] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis.* New York: McGraw-Hill, 2nd ed., 1991.