Modeling and Dimensioning Hierarchical Storage Systems for Low-Delay Video Services

S.-H. Gary Chan, *Member*, *IEEE*, and Fouad A. Tobagi, *Fellow*, *IEEE*

Abstract—In order to cost-effectively accommodate a large number of titles in a video system, a hierarchical storage system can be used. In this system, not-so-popular video files are stored in a tertiary level such as a disk/tape library. These files are transferred, or "staged," to a secondary level composed of magnetic disks before being streamed to the users. This system overcomes the current limitations in using disk/tape libraries to stream videos and resolves the bandwidth difference between staging and streaming. In this paper, we present, via analysis, a model of the system and determine the minimum storage and bandwidth required, at each level, to meet a given user delay goal. We also analyze a number of system operations pertaining to whether or not a file is played while it is being staged (i.e., stage-streaming) and whether or not the displayed segments are deleted (i.e., trail-deletion). We show that stage-streaming and trail-deletion can achieve substantially lower bandwidth and storage requirements. In order to further increase the streaming and storage scalability, a distributed storage system can be used where multiple local servers are put close to user pools and get their files from one of the libraries through a network. We extend the models developed to such a system and specify the resource requirements to meet a given delay goal.

Index Terms—Hierarchical storage systems, video services, tertiary and secondary levels, distributed storage systems, staging and replacement policies, user delay goal.

1 Introduction

On-demand video services encompass many important applications pertaining to entertainment, information, and education, such as movie-on-demand, news-on-demand, distance learning, etc. [1], [2], [3], [4]. In order to offer a wide variety of programs, a video system must accommodate a large number of video titles in a cost-effective manner.

Traditionally, magnetic disks have been used in video servers to stream videos because of their high throughput, low access latency, and random data access. This is the main reason why most of the previously proposed commercial video servers are based on magnetic disks [5], [6], [7], [8], [9]. Much research has been done regarding how to layout/stripe and access video data on disks in order to multiplex as many concurrent requests as possible and to achieve a certain level of fault tolerance. However, magnetic disks are still not cost-effective to store a large volume of video files due to their relatively high cost.

On the other hand, low-cost tertiary storage commonly known as library or "jukebox" is designed for mass storage in excess of terabytes. However, due to their relatively long access latency, tertiary systems are not yet suitable for video streaming. Therefore, a cost-effective approach is to make use of a hierarchical storage system consisting of a tertiary level and a secondary level. In this system, the tertiary storage stores all video files which are dynamically transferred or "staged" onto the secondary level for streaming according to user demand. This caching operation also resolves the mismatch between the drive bandwidth in the tertiary storage and the streaming rate of the videos, at the expense of some disk bandwidth due to staging.

Such a hierarchical system is attractive if many titles are not very popular because, in this case, the secondary storage space can be effectively shared among the titles. This is particularly true for video-on-demand systems, where the probability of a movie being accessed has been observed to match a Zipf distribution [10], [11]. That is, if P_i is the access probability of movie i, then $P_i \propto 1/i^{\zeta}$, where ζ is approximately 0.78 based on real video rental data. We show in Fig. 1 the typical distribution of P_i for a movie store of 600 movies. The access probability drops very fast, indicating only a small fraction of movies (about 20 percent) are popular. There is a rather long "flat" tail, showing that most of the movies are of low but rather uniform popularity. Because there are only a few popular movies,[1] it is justified to always keep them online in the secondary level so that they can be accessed with low delay and many requests can share a stream by means of multicasting and broadcasting techniques (see, for example, [12], [13], [14], [15], [16], [17] and references therein). The disk space at the secondary level can hence be partitioned into two parts: the part storing those popular movies and the part for staging the not-so-popular ones. In the remainder of this paper, we will consider only the part for staging and those many but not-so-popular movie archives. Obviously, the movies we

- *S.-H.G. Chan is with the Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. E-mail: gchan@cs.ust.hk.*
- *F.A. Tobagi is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305.*

Manuscript received 20 Feb. 2002; revised 12 July 2002; accepted 13 Feb. 2003.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 115921.

1. In this study, we refer to a "movie" as a video of reasonably long duration, say 30 minutes or more, so that a user would not mind being delayed by a few minutes before viewing the video.

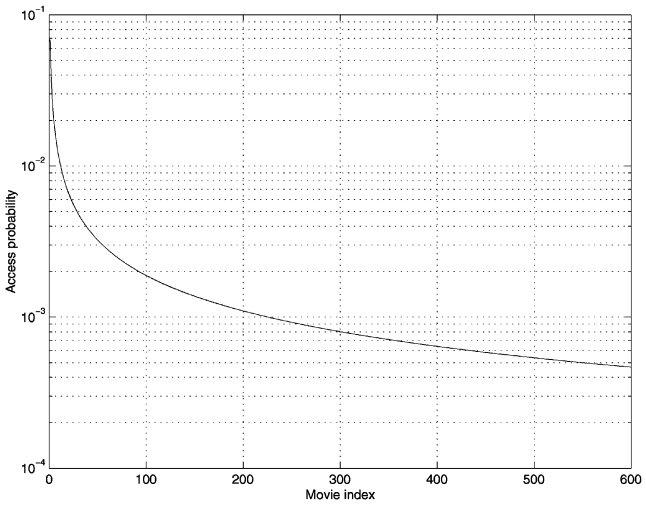

Fig. 1. The access probability of a video store of 600 movies ($\zeta = 0.78$).

focus on are those with the average number of concurrent users being less than one (i.e., no multicasting or broadcasting gain) so that the secondary storage space can be shared among the large pool of movies. For example, by again referencing Fig. 1. If the total arrival rate is 100 requests/hour and the movies are 90 minutes long, after some simple calculation, we may keep about 100 movies "online." In this way, the remaining 500 "offline" movies would have an aggregate request rate of 41 req/hr, with the average number of concurrent users in each of the movies being less than 1.[2]

For the offline movies, users may experience a delay due to queuing and file transfer. Therefore, a hierarchical storage system should be designed to meet a certain start-up delay goal. We present several system operations for the system based on file-by-file staging (file-by-file staging incurs fewer exchange overheads in the tertiary level when compared with block-by-block staging, where video blocks of different requests are staged in a multiplexed, round-robin manner [18]). These operations offer different degrees of user interactivity. If a system is to be designed for fully interactive applications where a user can interact with the displaying video from its beginning to end, movies must be completely staged before playback commences. For applications which do not require full user interactivity, a movie may be displayed *while* it is being staged (termed "stage-streaming" as discussed in [19], [18]) and the portion of the movie that has already been displayed may even be deleted ("trail-deletion") so as to reduce the storage requirement in the secondary level. In this paper, we address and compare, via modeling and analysis, the storage and bandwidth requirements in each level of the storage hierarchy using these staging mechanisms, given a certain user delay goal.

In order to further increase the storage scalability, a distributed storage system can be used, where a number of local servers are connected to a number of tertiary libraries in which files are stored. The local servers are placed close to the users to stream videos. They access movies from the libraries over a network or a switch. We show how our models can be extended to this case to dimension system parameters, in terms of the storage and bandwidth required in each level, to meet a certain user delay goal.

Our contributions are as follows: 1) We present an efficient method, via modeling, for choosing appropriate system parameters to meet a given user delay requirement; 2) we analyze and evaluate a number of system operations; and 3) we extend the models to design a distributed storage system. In modeling the system, our intent is not to introduce new mathematical techniques, but to make use of commonly known analytic/mathematical ones. Note that we are interested in determining the *minimum requirements* in bandwidths and storage to meet a delay goal. As disk performance and cost continue to improve every year, the actual values to use depends on the cost benefit of a hierarchical storage system. This issue has been investigated in [20], [21], [22], [23], [24].

Note that designing a hierarchical storage system is a two-step process. In the first step, we specify the minimum required delivery bandwidth and storage in each level. To meet these specifications, practitioners must follow up in the second step by designing, in detail, the respective level in terms of, for example, how data blocks are laid out and accessed in tapes and disks, and the scheduling of the disk arms and robotic arms. This paper focuses on the first step. Regarding the second step, there has been extensive research on designing the individual levels in isolation. Data layout and access issues in disk-based video servers have been reported in, for example, [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35]. Data layout for tertiary storage systems has been studied in [36], [37], [38]. We complement these bodies of work by considering a hierarchical storage system in which the two levels interact with each other.

Some other work on hierarchical storage systems includes the software design aspect [39], [40], [41], [42] and its admission control and user blocking performance [43], [44], [45]. Most of this work considers using the tertiary level to directly stream videos to the users, while ours considers the case where the tertiary drives cannot be used for video streaming (as for today's technology) and, hence, disks are used to resolve the bandwidth difference between tertiary staging and video streaming. As opposed to previous work in [46], [47], [48], [49], [50], we consider the interaction between the levels or dimensioning the system parameters to meet a certain user delay goal. Merchant et al. consider block-by-block staging in a hierarchical storage system [51], while we study file-by-file staging and distributed storage systems here. Though the hierarchical storage system appears to be similar to the traditional disk-memory caching system, previous work on disk-memory caching cannot be directly applied here. This is mainly due to the unique sequential access characteristics of video applications, which leads to some new operational variations (such as trail-deletion and stage-streaming) that are not possible in traditional caching systems.

This paper is organized as follows: In Section 2, we present the model, design framework, and some design examples of a baseline hierarchical storage for interactive

2. The hottest one among the 500 offline movies has arrival rate around $100 \times 2 \times 10^{-3} = 0.2$ req/hr. Given that the holding time is 90 minutes, the average number of concurrent users for the movie is $0.2 \times 90/60 = 0.3$, which is less than 1.

applications. In Section 3, we analyze and evaluate a number of system operations so as to offer different levels of user interactivity. In Section 4, we extend the models developed to address the design issues of distributed storage systems. We conclude in Section 5.

2 Analytic Model and Dimensioning of a Baseline Hierarchical Storage System

In this section, we first describe the operation of a simple baseline hierarchical storage system. Since the design of the system involves multiple parameters pertaining to storage and bandwidth, we have used analysis to facilitate the process. After presenting the model for the system, we show some illustrative design examples. Table 1 lists some of the important symbols used in the paper.

2.1 The Operation of a Baseline Hierarchical System

The baseline hierarchical storage system is composed of a single tertiary library and a secondary level. The secondary storage level consists of disks and has a certain aggregate effective bandwidth, B_2 MB/s, which can be either "partitioned" or "shared" between staging and streaming, depending on whether or not a certain amount of bandwidth is permanently set aside for staging or streaming purpose. ("Effective" bandwidth refers to net data delivery rate after taking into account the access overheads and scheduling policies.) The level has a limited storage capacity C_2 in MB. This abstraction of secondary level is reasonable given that video servers nowadays comes with a certain maximum number of streams and a certain storage capacity.

The tertiary storage system has many removable storage media such as optical disks or tapes and $N_{dr}^{(3)}$ number of drives, where $N_{dr}^{(3)}$ typically ranges from 1 to 16. Each drive has a certain effective delivery bandwidth b_3 MB/min; hence, the maximum bandwidth the tertiary storage level can deliver at any time is given by $B_3 = b_3 N_{dr}^{(3)}$. Typically, $B_3 \leq B_2$. The tertiary drives can be operated independently, in which case, multiple requests can be served at the same time. They can also be configured as if they form a single drive by means of striping, in which case, only a single request is served at any one time with rate B_3 (we define $N_{dr}^{(3)} = 1$ in this case).

In the baseline system, videos are displayed after they are completely staged and no portion of the videos can be deleted while they are being displayed. In this way, users can have full freedom in interacting with the videos. Variations in the system operation and their analysis and comparisons will be addressed in later sections.

As mentioned before, we only consider the set of not-so-popular movies, which are stored in the tertiary level; let the number of such movies be N_v. We define an arrival as a hit if it requests a video that has already been staged to secondary storage and, hence, is ready to be streamed; otherwise, it is a miss. Let D_{st}, D_{st}^{Hit}, and D^{Miss} be the *st*art-up delay of a random user, of a hit, and of a miss, respectively, and let $\overline{D}_{st}$, $\overline{D}_{st}^{Hit}$, and $\overline{D}^{Miss}$ be their respective averages. Since all movies must be staged to secondary storage before they are displayed, the start-up delay of a miss on arrival is given by $D^{Miss} + D_{st}^{Hit}$. Therefore,

$$\begin{aligned} \overline{D}_{st} &= \alpha_2 \overline{D}_{st}^{Hit} + \alpha_3 \left(\overline{D}^{Miss} + \overline{D}_{st}^{Hit} \right) \\ &= \overline{D}_{st}^{Hit} + \alpha_3 \overline{D}^{Miss}, \end{aligned} \tag{1}$$

where α_2 is the hit rate at arrival and $\alpha_3 \equiv 1 - \alpha_2$ is the miss rate.

We design a hierarchical storage system operating under a certain "target" arrival rate λ_0. In reality, λ_0 may be the maximum arrival rate that the server is designed for. The case of interest is $1 \ll \lambda_0 T_h \ll N_v$, where T_h is the average holding time of the users in the system. We want to find C_2, B_2, and B_3 so as to meet a certain (low) user delay requirement, such as $\overline{D}_{st} \leq \hat{D}$ (average delay requirement), $\overline{D}^{Miss} \leq \hat{D}^{Miss}$ (statistical delay guarantee for misses), or $P(D_{st} > \hat{D}) \leq \varepsilon$ (statistical delay guarantee).

We further note that, since the parameters C_2, B_2, and B_3 can be traded off with each other, there are many sets of values to meet the delay requirement. Specific values can be chosen with some additional constraints such as: 1) if there is a cost associated with each set of parameters and the total cost of the system is to be minimized, 2) given today's technology, some sets may be feasible while some others may not, e.g., a disk may come with a certain storage and bandwidth and, hence, C_2 and B_2 must be related accordingly (this has been studied in [52]), or 3) there is a "knee" in the trade off curve, making this set a likely design choice. We will adopt this in our design. We next present our analytic model, which is later validated by simulation.

2.2 Analytic Model

Regarding user requests, we assume, as in other literature, a large population and that requests arrive according to a Poisson process with rate λ req/min; the design point is hence given by $\lambda = \lambda_0$. All users are admitted and they wait until their videos are displayed. Each stream is held for a certain holding time; for the sake of simplicity, we consider it to be a constant T_h minutes. The streaming rate is b_0 MB/min for all movies at all times.[3] The maximum number of concurrent streams in the secondary level is therefore $s_2 = \lfloor B_2/b_0 \rfloor$. We also consider that video files are of the same size C_f MB; therefore, the maximum number of files that can be stored in the secondary level is $N_2 = C_2/C_f$. Note that, since users may interact with the videos, C_f is not necessarily equal to $b_0 T_h$. We further let

$$N_3 \triangleq N_v - N_2 \tag{2}$$

be the number of video files that cannot be held by the secondary storage.

We show in Fig. 2 a model of the hierarchical storage system. Each arrival independently chooses movie i with probability p_i, where $1 \leq i \leq N_v$ and $\sum_i p_i = 1$. All the hits are put into a hit queue waiting for available streams. If a

3. Using simulations, we find that the performance does not change much if arbitrary distributions for holding times and streaming rates are used.

TABLE 1
Symbols or Variables Used in the Paper

Variables	Remarks
λ	External arrival rate to the hierarchical storage system (req/min)
λ_0	Target arrival rate at which the hierarchical system is designed (req/min)
λ_3	Request rate to the tertiary level (req/min)
D_{st}	Start-up delay for a random user in the system (minutes)
D_{st}^{Hit}	Start-up delay for a hit in the secondary level (minutes)
D^{Miss}	Delay for a miss in the secondary level (minutes)
N_v	Total number of video files in the system
N_2	Total number of video files that can be stored in the secondary level
N_3	$= N_v - N_2$
n_2	Additional number of streams (or storage space) beyond $\lambda_0 T_h$ in the secondary level in order to achieve negligible hit delay in the level ($\approx$ 8–18)
α_2	Hit probability for an incoming request $= N_2/N_v$
α_3	Miss probability for an incoming request $= 1 - \alpha_2$
b_0	Streaming rate for a video file (MB/min)
C_f	Size of a video file (MB)
T_h	Average user's holding time (minutes)
s_2	Maximum number of streams in the secondary level $= \lfloor B_2/b_0 \rfloor$
C_2	Total storage capacity in the secondary level (MB) $= N_2 C_f$
B_2	Total bandwidth in the secondary level (MB/min)
B_3	Total bandwidth in the tertiary level (MB/min)
$1/\mu_3$	Average service time of the drives (staging time + exchange time) in the tertiary level (minutes)
$N_{dr}^{(dr)}$	Number of independent drives in the tertiary level
b_3	Bandwidth of each drive in the tertiary level (MB/min)

request is a miss, it joins the staging-movie queue, which stages the movies according to First-Come-First-Served. Note that, in addition to the staging time, misses may have to wait for some storage space in the secondary level to be available before staging can begin and, hence, the gate corresponding to the waiting time for a "deletable" file in the figure. We use Least-Recently-Used (LRU) as our file replacement policy. Once a file is completely staged, the miss(es) corresponding to the movie becomes hit(s) and join the hit queue. Given the above system model and a user delay goal, the problem is to determine the parameters B_2, C_2, and B_3.

We begin by specifying B_2, which is used for streaming and staging. Since each user holds a stream for T_h minutes, for stability we must have $s_2 \geq \lambda_0 T_h$. Writing $s_2 = \lambda_0 T_h + n_2$, where $n_2 \geq 0$, we plot in Fig. 3 the average waiting time for a stream against n_2, based on simulating the secondary level as an M/D queue. We see that n_2 can be very small, in the range of

$$n_2 \approx \text{8-18}, \tag{3}$$

to achieve virtually no waiting. Therefore, we need

$$B_2 \geq (\lambda_0 T_h + n_2) b_0 \tag{4}$$

for streaming purposes.

Since the bandwidth in the secondary level is also used for file staging, we need to add B_3 to it. If B_2 is partitioned between staging and streaming, the total B_2 required is

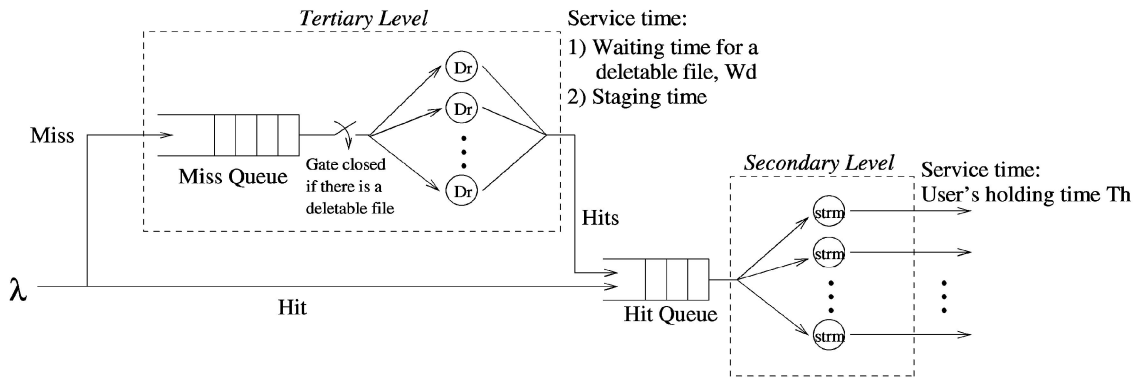

Fig. 2. A model for the hierarchical storage system.

$$B_2 = (\lambda_0 T_h + n_2) b_0 + B_3. \tag{5}$$

Even if the bandwidth is shared, the required B_2 is still roughly the same, as B_3 is typically well-utilized (with utilization ≥ 0.6 from simulation and analysis).

We next obtain the minimum C_2 required so that the misses do not experience additional delay due to a lack of storage space. We call this waiting for a deletable file. Note that each request holds a storage space of C_f for a time $T_h + T_{stg}$ (because concurrent access to the same file is not likely given our assumption), where T_{stg} is the staging time of a file given by C_f/B_3. Typically, $T_{stg} \ll T_h$ and, hence, on average, $\lambda_0 T_h$ concurrent files are open for streaming. Therefore, we can use the same argument of obtaining B_2 above to obtain the minimum storage requirement, i.e.,

$$C_2 = (\lambda_0 T_h + n_2) C_f. \tag{6}$$

Note that if one uses n_2 and, hence, the C_2 and B_2, specified above, the hit delay would be negligible and, hence, we can approximate the average delay for all users $\overline{D}_{st}$ in (1) as

$$\overline{D}_{st} \simeq \alpha_3 \overline{D}^{Miss}, \tag{7}$$

i.e., user delay is mainly due to misses.

What remains is B_3, which is obtained by designing the tertiary subsystem to meet the user delay requirement. The arrival process to the tertiary level is Poisson with rate

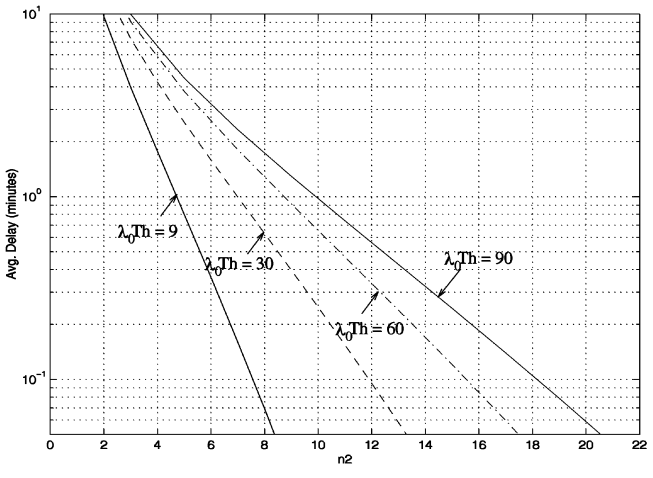

Fig. 3. Average waiting time for a stream versus n_2, given $\lambda_0 T_h$.

$$\lambda_3 \triangleq \alpha_3 \lambda, \tag{8}$$

where, for uniform video popularity,

$$\alpha_3 = 1 - N_2/N_v. \tag{9}$$

Due to uniform file size, the tertiary subsystem can be modeled as an $M/D/N_{dr}^{(3)}$ system with service time $1/\mu_3$ given by

$$\frac{1}{\mu_3} = \frac{C_f}{b_3}. \tag{10}$$

For simplicity, we will use an M/M model. As compared to the actual M/D case, the performance obtained this way is, hence, slightly pessimistic.

As shown above, the user delay distribution in the hierarchical storage system depends on the delay performance of the tertiary level. The delay distribution of an M/D or M/M system is well-known in the literature. For concreteness and as an example, we will consider average delay requirement in the following. Given an overall average delay requirement $\hat{D}$, the miss delay requirement, $\hat{D}^{Miss}$, can be obtained as

$$\hat{D}^{Miss} \simeq \hat{D}/\alpha_3. \tag{11}$$

Given $\hat{D}^{Miss}$, we numerically solve for μ_3 of the following equation:

$$\hat{D}^{Miss} = \frac{1}{\mu_3} + \frac{P_Q}{N_{dr}{}^{(3)}\mu_3 - \lambda_3}, \tag{12}$$

where P_Q is the well-known probability of queuing in the tertiary level and is a function of μ_3 (see, for example, [53] for its expression). With μ_3, the required B_3 is then given by

$$B_3 = N_{dr}^{(3)} C_f \mu_3. \tag{13}$$

For the special case when $N_{dr}^{(3)} = 1$, we have $P_Q = \lambda_3/\mu_3$ and, hence,

$$B_3\left(N_{dr}^{(3)} = 1\right) = C_f\left(\lambda_3 + \frac{\alpha_3}{\hat{D}}\right). \tag{14}$$

The design for nonuniform popularity follows the same procedure as above, except that the expression of α_3 is different. We assume that the distribution of the movie popularity can be characterized by multiple popularity

TABLE 2
System Parameters Required to Achieve $\hat{D} = 2$ Minutes

	$N_v = 500$		$N_v = 5000$	
	$\lambda_0 = 20$/hr (Configuration I)	$\lambda_0 = 50$/hr (Configuration II)	$\lambda_0 = 20$/hr	$\lambda_0 = 50$/hr
C_2 (GB)	45	90	45	90
α_3	0.91	0.82	0.991	0.982
B_3 (MB/s)	12.63	18.21	13.76	21.82
B_2 (MB/s)	21.1	35.1	22.2	38.7

$C_f = 1$ *GB*, $b_0 = 1.5$ *Mbit/s*, $T_h = 90$ *minutes*, $N_{dr}^{(3)} = 1$, $n_2 = 15$.

classes, within each of which all videos are equally likely to be requested. Let A be the number of popularity classes, each of which containing $N_{v,a}$ videos, where $1 \le a \le A$. The total number of videos is hence given by $N_v = \sum_{a=1}^{A} N_{v,a}$. Let P_a be the probability of a random arrival choosing popularity class a, where $\sum_a P_a = 1$. The request rate for each class is hence given by $\lambda_a = P_a\lambda$. Further, let the popularity of the ith movie in class a be $p_{i,a} = P_a/N_{v,a}$, for $1 \le i \le N_{v,a}$. Because the extra n_2 space in the secondary level is shared among the popularity classes, the hit probability for class a can be estimated as

$$\alpha_{2,a} = (\lambda_a T_h + P_a n_2)/N_{v,a}. \tag{15}$$

Therefore,

$$\alpha_3 = \sum_{a=1}^{A} P_a(1 - \alpha_{2,a}). \tag{16}$$

The design procedure of hierarchical storage systems can be summarized as follows:

1. Obtain the minimum C_2 using (6).
2. Given C_2, find α_3 by means of (9) and (16) for uniform and nonuniform popularity, respectively.
3. After solving numerically for μ_3 in (12), obtain the minimum B_3 using (13). The special case of $N_{dr}^{(3)} = 1$ can be obtained directly using (14).
4. Determine the minimum B_2 with (5).

2.3 Illustrative Design Examples

We next present some design examples given the user delay requirement $\overline{D}_{st} \le \hat{D}$. Since video files have to be completely staged before they are displayed, it is important to keep the staging time as short as possible. It is well-known in queuing systems that a single "big" server achieves lower total system time than multiple "small" servers. Therefore, the drive bandwidth in the tertiary level for such interactive applications should be used in parallel, i.e., $N_{dr}^{(3)} = 1$.

We first consider uniform video popularity. Table 2 shows the design of a hierarchical server satisfying $\hat{D} = 2$ minutes, given a certain target arrival rate λ_0 for a medium-size system with $N_v = 500$ and a large-size system with $N_v = 5,000$ ($C_f = 1$ GB, $T_h = 1.5$ hr, $b_0 = 1.5$ Mb/s, $N_{dr}^{(3)} = 1$, $n_2 = 15$).

We see that the required C_2 is much less than $N_v C_f$, showing that the hierarchical storage system greatly reduces the system storage cost (only about 10-20 percent of the total storage is needed in the secondary level). There is also an "economy of scale" in terms of the bandwidth required—higher λ_0 does not lead to proportionally higher bandwidth requirement. For $N_v = 500$, the utilization of B_3 and B_2 for $\lambda_0 = 20$ req/hr are 0.4 and 0.51, respectively, while that of $\lambda_0 = 50$ req/hr are 0.69 and 0.8, respectively. These utilizations are reasonable given our relatively low delay requirement. Note that, as N_v increases, the system requirements do not increase proportionally. This is the advantage of a hierarchical storage system: It makes use of the low-cost tertiary level for storage and the high-performance secondary level for streaming. We reemphasize that the storage requirements specified here are for the set of not-so-popular movies. This storage is *in addition* to those popular movies being stored online. We note that, with a higher b_0, both the secondary and tertiary bandwidth requirements would be proportionally higher as we are dealing with a system with higher throughput and larger file size.

We show in Fig. 4 the system performance with $N_v = 500$ using the parameters specified in configurations I and II of Table 2 by means of simulation, with B_2 shared between staging and streaming. The delay at λ_0 is, as expected, slightly lower than $\hat{D}$ because we have used the simpler and slightly pessimistic M/M model. Though the server satisfies the delay requirement at λ_0, the delay increases rapidly when the arrival rate is more than λ_0. This is because we have designed our system at the "trade off knee." This is because we size our parameters just enough to meet the delay goal at the design point. When the arrival rate increases beyond the point, the system runs out of resources. There is no longer enough secondary storage and, hence, the waiting time for a deletable file increases. As each user stays in the system longer, the queue quickly builds up, leading to high delay. Since the deployed servers can be used at rates different from what they were designed for, the figure shows that it is important to consider the maximum arrival rate in order to prevent unacceptable delay should the arrival rate increase. It also indicates the

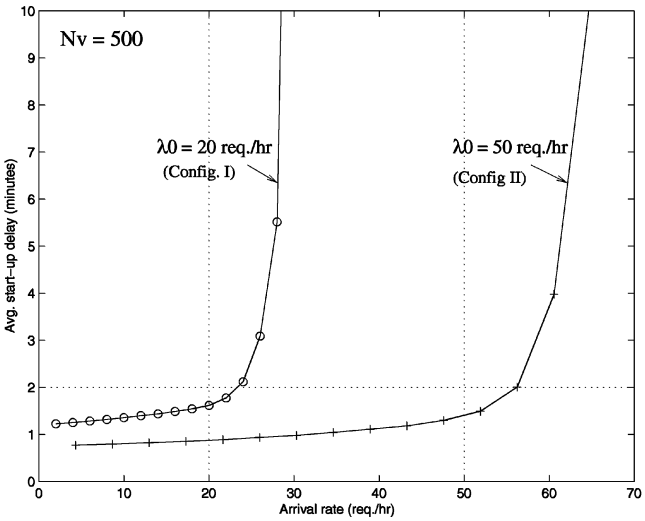

Fig. 4. Server performance with a design satisfying $\hat{D} = 2$ minutes at $\lambda_0 = 20$ req/hr or $\lambda_0 = 50$ req/hr ($N_v = 500$, $C_f = 1$ GB, $b_0 = 1.5$ Mb/s, $T_h = 90$ minutes, $N_{dr}^{(3)} = 1$, and $n_2 = 15$).

importance of admission control for arrival rate beyond the design point λ_0.

To validate our model based on nonuniform popularity as given by Fig. 1 in Section 1, we assume that the least popular $N_v = 500$ movies are stored offline. We approximate the distribution as uniform and, hence, use configuration II given in Table 2. In Fig. 5, we show the simulation result of this set of 500 movies. The delay requirement is met, showing that, in reality, we do not need to know the exact popularity distribution before designing the system, i.e., the performance of the system is not sensitive to the distribution.

We end this section by considering the case of nonuniform video popularity, with $\hat{D} = 2$ minutes and $A = 3$ popularity classes. The class popularities are $P_1 = 0.05$ (unpopular archives), $P_2 = 0.25$, and $P_3 = 0.7$ (more popular archives). For $N_v = 500$, the number of movies in each class is $N_{v,1} = 150$, $N_{v,2} = 200$, and $N_{v,3} = 150$, respectively,

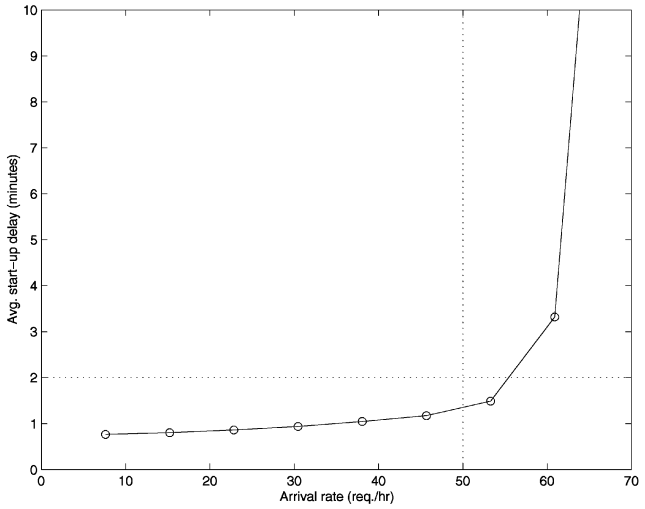

Fig. 5. The case with nonuniform video popularity approximated as uniform.

TABLE 3
System Requirements to Achieve $\hat{D} = 2$ Minutes with $A = 3$ Popularity Classes

	$N_v = 500$		$N_v = 5000$	
	$\lambda_0 = 20$/hr	$\lambda_0 = 50$/hr	$\lambda_0 = 20$/hr	$\lambda_0 = 50$/hr
C_2 (GB)	45	90	45	90
α_3	0.8382	0.6764	0.9856	0.9676
B_3 (MB/s)	11.64	15.03	13.689	21.503
B_2 (MB/s)	20.07	31.9	22.13	38.38

$C_f = 1$ *GB*, $b_0 = 1.5$ *Mbits/s*, $T_h = 90$ *minutes*, $N_{dr}^{(3)} = 1$, *and* $n_2 = 15$.

while, for $N_v = 5,000$, the number of movies in each class is $N_{v,1} = 1,500$, $N_{v,2} = 2,000$, and $N_{v,3} = 1,500$, respectively. Table 3 shows the bandwidth and storage requirements. For $N_v = 500$, the utilization of B_2 is given by 0.51 and 0.74 for $\lambda_0 =$ 20/hr and 50/hr, respectively. This is again reasonable given our low user delay requirement. Note that the requirements in Table 3 are slightly lower than that of the uniform popularity case (see Table 2). This is expected because the miss rate α_3 in this example is only slightly lower. We again see that, as the number of movies increases, the resources do not need to be increased substantially. As compared to Table 2, the system requirements for nonuniform popularity is slightly lower. Given the little difference between the uniform and nonuniform cases, we may design a hierarchical storage system using uniform popularity, i.e., the exact knowledge of the popularity distribution is not strictly required. If the actual popularity turns out to be nonuniform, the performance of the system would be better.

3 Variations in System Operation

So far, we have considered that a video is displayed after it is completely staged. To decrease the start-up delay, a video can be displayed *while* it is being staged; this is called *stage-streaming*. Since a file is displayed while it is being staged, the system will be less interactive until the file is completely staged. So far, we have also considered keeping the displayed portion of a video. To reduce the secondary storage requirement, we may "delete the trail" by keeping only the portion which is to be played later; in this case, the user cannot rewind the video. This *trail-deletion* saves storage at the cost of some user interactivity. In this section, we analyze and compare these operations.

3.1 Scheme Analysis

We first analyze stage-streaming without trail-deletion. For continuity, the data staging rate data must be at least equal to the data streaming rate, i.e., $b_3 \geq b_0$. If the staging rate is higher than the streaming rate, eventually the whole file would be staged before the video finishes. Arrivals finding their movies being or completely staged would enjoy no delay.

To specify the system requirements, we first note that the minimum C_2 is the same as that of complete staging before display (i.e., (6)) and $B_2 = (\lambda_0 T_h + n_2)b_0 + B_3$ as before.

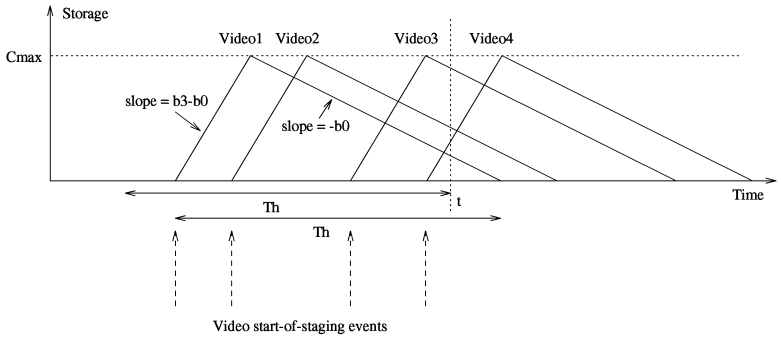

Fig. 6. Storage requirement of a video file for trail-deletion as a function of time. The total storage requirement at a time is the sum of the storage at that time.

Since misses suffer only queuing, as opposed to system, delay in the tertiary level, b_3 can be chosen to meet such delay requirement given our $M/M/N_{dr}^{(3)}$ approximation. Therefore, the only difference between this scheme and "complete staging before display" is the reduction in B_3 and, hence, accordingly B_2.

We next consider stage-streaming with trail deletion. The required secondary storage space can be very low if the staging bandwidth for a video file is comparable to its streaming bandwidth. Since portions of video files are deleted once they are displayed, all requests would be misses; i.e.,

$$\lambda_3 = \lambda_0. \tag{17}$$

Due to the higher miss rate, the required B_3 to meet a delay goal with trail-deletion would be slightly higher than without. In other words, trail deletion trades off the reduction in C_2 with B_3 and, accordingly, B_2, which is equal to $(\lambda_0 T_h + n_2)b_0 + B_3$.

The storage requirement of the secondary level can be found as follows. We show in Fig. 6 the storage occupied for a video file against time. Its storage requirement first increases with a rate $(b_3 - b_0)$ for a time C_f/b_3 and then decreases with the consumption rate b_0 for a period $(1 - b_0/b_3)T_h$. The maximum storage space occupied by a file is

$$C_{max} = \left(1 - \frac{b_0}{b_3}\right) C_f. \tag{18}$$

Note that the total secondary storage required at any time t is the aggregate sum of the storage of all open files at that time. To obtain the required C_2, observe that the "start-of-staging" process corresponds to the service-entrance process in the tertiary level, which is an $M/D/N_{dr}^{(3)}$ queue. By comparing the distributions using simulation, we find that such a process can be approximated by Poisson [54]. The storage requirement at time t is hence the sum of the storage of all the open files within the time interval $[t - T_h, t]$. Given a start-of-staging event within this interval, the time of its occurrence, T_s, is uniformly distributed within the range. If C is the storage space of an open file at time t, then

$$\begin{aligned} &P(C \le c) = \\ &P\left(T_s \in \left[t - \frac{c}{b_3 - b_0}, t\right] \bigcup T_s \in \left[t - T_h, t - T_h + \frac{c}{b_0}\right]\right) \\ &= \frac{c}{(b_3 - b_0)T_h} + \frac{c}{b_0 T_h} \\ &= \frac{c}{C_{max}}. \end{aligned} \tag{19}$$

Therefore, given a staging event in $[t - T_h, t]$, the storage requirement of the video file at time t is uniformly distributed between 0 and C_{max}.

Let X be the total secondary storage space required at any time t. Then, $X = \sum_{i=1}^{N} C_i$, , where $C_i \sim U[0, C_{max}]$ is the storage for the ith opened file and N is the random number of files opened at that time, which is a Poisson distribution. For reasonably large N, say $N \ge 5$, by central limit theorem, we have $\frac{1}{N}X \mid N \sim \mathcal{N}(\frac{C_{max}}{2}, \frac{C_{max}^2}{12N})$, where $\mathcal{N}(\mu, \sigma^2)$ is a Gaussian distribution with mean μ and variance σ^2. Therefore, if C_2 is the secondary storage capacity, the probability of running out of storage space is given by, after conditional on N,

$$\begin{aligned} P(X \ge C_2) &= \sum_{n=1}^{\infty} P\left(\sum_{i=1}^{n} C_i \ge C_2 \mid N = n\right) P(N = n) \\ &\approx \sum_{n=1}^{\infty} \left\{1 - \Phi\left(\frac{C_2/C_{max} - n/2}{\sqrt{n/12}}\right)\right\} \frac{(\lambda_0 T_h)^n}{n!} e^{-\lambda_0 T_h}, \end{aligned} \tag{20}$$

where $\Phi(x) \triangleq \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\xi^2/2} d\xi$ is the cumulative normal distribution function. In general, $P(X \ge C_2)$ should be kept low (say, $\le 10^{-3}$).

3.2 Illustrative Examples and Comparisons

In this section, we show some illustrative examples with average delay requirement and uniform video popularity. As shown in the previous section, the performance for nonuniform popularity is similar. We first consider stage-streaming without trail deletion. As opposed to the case of "complete staging before display," where all tertiary drives should be used in parallel to stage a file, the tertiary drives in the stage-streaming operation should be used independently in order to reduce the bandwidth requirement. This is because the misses experience only the queuing delay in the tertiary level. In Fig. 7, we compare B_3 required for

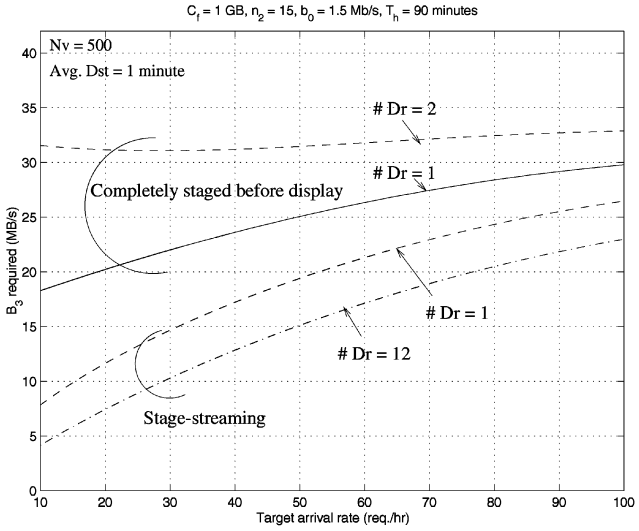

Fig. 7. Comparison of B_3 requirement between "complete staging before display" and stage-streaming given $N_{dr}^{(3)}$ ($N_v = 500$ and $\hat{D} = 1$ minute, $C_f = 1$ GB, $b_0 = 1.5$ Mb/s, $T_h = 90$ minutes, and n_2 = 15).

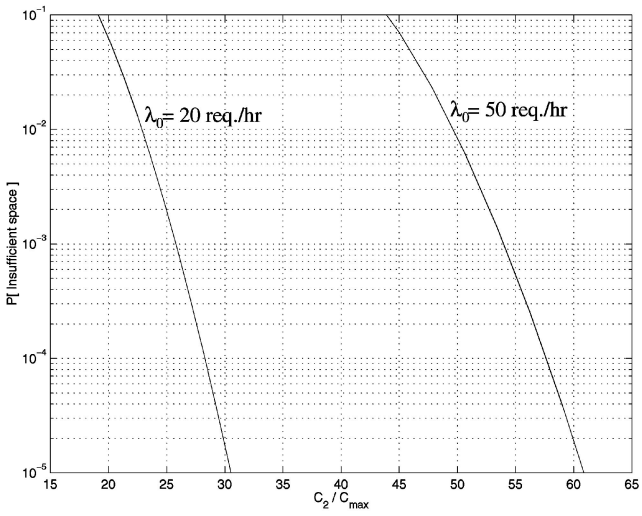

Fig. 9. Probability of insufficient storage for stage-streaming with trail-deletion versus C_2/C_{max}, given λ_0.

"stage-streaming" and "complete staging before display" as a function of λ_0, given $N_{dr}^{(3)}$ ($\hat{D} = 1$ minute, $N_v = 500$, $C_f = 1$ GB, $b_0 = 1.5$ Mb/s, and $T_h = 90$ minutes). Stage-streaming achieves a much lower bandwidth requirement and performs the best with independent tertiary drives. We note in passing that, given $\hat{D}$, the difference in system requirements between the two staging schemes decreases with increasing $\hat{D}$ (not shown). This is because, while queuing delay at the tertiary level is what users experience in stage-streaming, queuing delay *plus* staging time is the user delay experienced for "complete staging before display." As $\hat{D}$ increases, queuing delay at the tertiary level constitutes more and more of the total user delay, therefore narrowing the difference between the two schemes.

Let's now consider trail-deletion with stage-streaming. In Fig. 8, we compare the required B_3 with and without trail deletion ($\hat{D} = 1$ minute and $N_{dr}^{(3)} = 12$). Without trail-deletion, due to the increase in C_2, the hit rate increases slowly with λ_0. Therefore, B_3 increases sublinearly with λ_0. However, with trail deletion, all requests are misses and, hence, B_3 increases with a faster rate as λ_0 increases. Trail deletion hence trades off B_3 against C_2. This is shown in Fig. 9, where we plot the probability of running out of secondary storage against C_2/C_{max} (20), given λ_0 ($T_h = 1.5$ hour). In order to keep the probability of running out of storage low, say 10^{-2}-10^{-4}, we need $C_2 = \beta(\lambda_0 T_h)C_{max} = (\beta\lambda_0 T_h)(1 - b_0/b_3)C_f$,, where $\beta \simeq 0.7$-1. We see that, depending on the relative ratio between b_0 and b_3, substantial storage reduction is possible as compared to the case of not using trail-deletion, which requires storage $\approx \lambda_0 T_h C_f$.

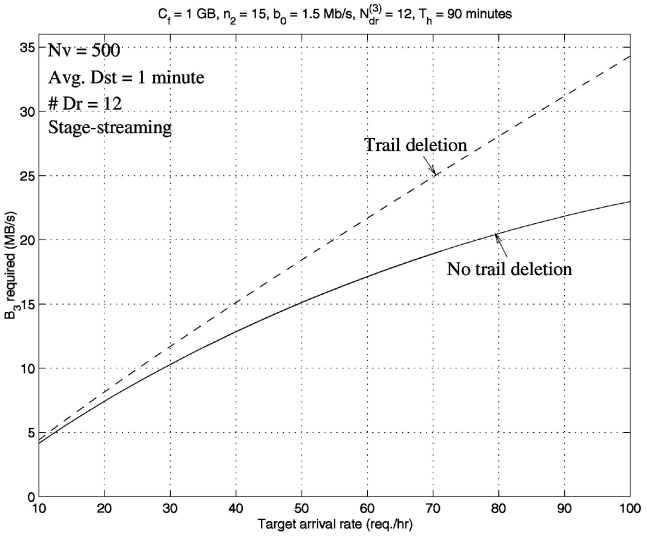

Fig. 8. Comparison of B_3 requirement with and without trail deletion ($\hat{D} = 1$ minute, $N_v = 500$, $C_f = 1$ GB, $b_0 = 1.5$ Mb/s, $T_h = 90$ minutes, $n_2 = 15$).

4 A Distributed Storage System

In this section, we consider the problem of dimensioning the parameters of distributed storage systems to meet a certain delay goal. We extend the models we developed in Section 2.2 to address this problem. We show in Fig. 10 a distributed storage system, where S local servers are connected to L tertiary libraries where files are stored. We will focus on "complete staging before display" without trail deletion in this section; our previous discussion can be used to extend the results to staging-streaming and "trail deletion." We first present the analysis of the system, and then provide a design example at the end of the section.

4.1 Analysis

We want to specify the bandwidth and storage requirements for each of the local servers and bandwidths for each of the libraries in order to meet a delay requirement, given a certain target aggregate arrival rate λ_0 for those not-so-popular movies. Note that, out of the total request rate in a library, a small fraction may come from a particular local server. Therefore, partitioning a library bandwidth for each local server would not be economical. By the same token, since it is unlikely for all the libraries to be staging files to a single local server at the same time, partitioning bandwidth

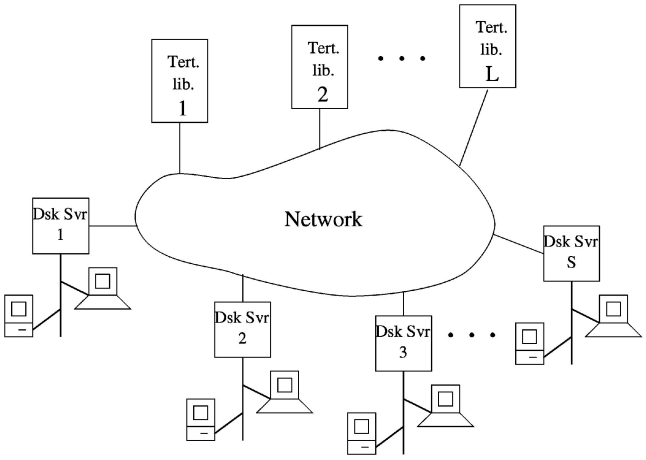

Fig. 10. A distributed storage system with multiple local servers accessing data from libraries.

in a local server for each library is also not economical. Hence, library bandwidth should be shared among the local servers, while the bandwidth of a local server should be shared among all the libraries.

We first present our notations, which are also summarized in Table 4. Let $B_2^{(s)}$ and $C_2^{(s)}$ be the secondary bandwidth and local storage for server s, $s = 1 \ldots S$, and $B_3^{(l)}$ be the bandwidth for library l, $l = 1 \ldots L$. Let $\nu_s \lambda_0$ be the request rate for server s, with $\sum_{s=1}^{S} \nu_s = 1$. The hit probability in server s, $\alpha_2^{(s)}$, is then given by

$$\alpha_2^{(s)} = \frac{n_2 + T_h \nu_s \lambda_0}{N_v}, \tag{21}$$

where n_2 has been given by (3). Let $\eta_l^{(s)}$ be the fraction of misses in server s requesting a file from library l, where $\sum_{l=1}^{L} \eta_l^{(s)} = 1$.

The design procedure of the system is as follows: Let's consider average delay requirement and let $\overline{D}_l^{Miss}$ be the average delay for library l. The average delay for all misses is given by

$$\overline{D}^{Miss} = \frac{\sum_{s=1}^{S} \sum_{l=1}^{L} \nu_s \left(1 - \alpha_2^{(s)}\right) \eta_l^{(s)} \overline{D}_l^{Miss}}{\sum_{s=1}^{S} \nu_s \left(1 - \alpha_2^{(s)}\right)}. \tag{22}$$

As hit delay is negligible, the average start-up delay in server s is given by $\overline{D}_{st}^{(s)} = (1 - \alpha_2^{(s)}) \overline{D}^{Miss}$ and the average overall delay is $\sum_{s=1}^{S} \sum_{l=1}^{L} \nu_s (1 - \alpha_2^{(s)}) \eta_l^{(s)} \overline{D}_l^{Miss}$.

Denote $\lambda_3^{(l)}$ the request rate in library l. $\lambda_3^{(l)}$ is given by

$$\lambda_3^{(l)} = \sum_{s=1}^{S} \eta_l^{(s)} \left(1 - \alpha_2^{(s)}\right) (\nu_s \lambda_0). \tag{23}$$

Define $\delta_l^{(s)}$ as the fraction of files staged from library l to server s. We have

$$\delta_l^{(s)} = \frac{\eta_l^{(s)} \left(1 - \alpha_2^{(s)}\right) (\nu_s \lambda_0)}{\lambda_3^{(l)}}. \tag{24}$$

We now state the procedures for designing the distributed storage system:

1. We first obtain $C_2^{(s)}$ by $(\lambda_0 \nu_s T_h + n_2) C_f$.
2. The bandwidth used for streaming at server s is given by $(\lambda_0 \nu_s T_h + n_2) b_0$.
3. Using (23), we have $\lambda_3^{(l)}$. Given $\lambda_3^{(l)}$, we then determine $B_3^{(l)}$ to satisfy the miss delay requirement.

TABLE 4
Important Variables Considered in the Distributed Storage Environment

Variables	Remarks
L	Number of libraries in the system
S	Number of local servers in the system
$C_2^{(s)}$	Total storage in server s (MB)
$B_2^{(s)}$	Total bandwidth in server s (MB/min)
$B_3^{(l)}$	Total bandwidth in library l (MB/min)
$\lambda_3^{(l)}$	Aggregate request rate for library l (req/min)
ν_s	Fraction of the total arrival directed to local server s
$\alpha_2^{(s)}$	Hit rate in the local server s
$\eta_l^{(s)}$	Fraction of misses in the local server s is directed to library l
$\delta_l^{(s)}$	Probability of a file in library l to be staged to local server s
$\sigma^{(l)}$	Bandwidth utilization in library l (MB/min)

We further let $\sigma^{(l)} = \lambda_3^{(l)} C_f / B_3^{(l)}$ be the resultant utilization of $B_3^{(l)}$ for library l and, hence, $\delta_l^{(s)}\sigma^{(l)}$ is the utilization of $B_3^{(l)}$ to deliver files to server s. Denote $B_3^a = \sum_{l=1}^{L} B_3^{(l)}$ as the maximum aggregate bandwidth used for staging in the system. We obtain $B_2^{(s)}$ as follows: We first find the bandwidth necessary in server s for staging, $B_*^{(s)}$. Obviously, $B_*^{(s)} \le B_3^a$. In fact, since the probability for all L libraries simultaneously staging to a certain local server is likely to be low, $B_*^{(s)}$ can be much less than B_3^a due to bandwidth sharing. We pick $B_2^{(s)}$ so that the probability for the total staging bandwidth exceeding its value is very low, e.g., less than 0.1.

$B_*^{(s)}$ is obtained as follows: At any arbitrary time t, library l is staging a file to server s with probability $\delta_l^{(s)}\sigma^{(l)}$. Consider the following generation function for server s:

$$\begin{aligned} g_s(y) &= \prod_{l=1}^{L}\left((1-\delta_l^{(s)}\sigma^{(l)}) + \delta_l^{(s)}\sigma^{(l)} y^{B_3^{(l)}}\right) \\ &\equiv \sum_{i=0}^{2^L} c_i^{(s)} y^{B(i)}, \end{aligned} \tag{25}$$

where we have arranged the exponent of y so that $0 = B(0) \le B(1) \le \ldots \le B(2^L) = B_3^a$. The coefficient $c_i^{(s)}$ is the probability that staging bandwidth in server s equals $B(i)$. We can therefore take $B_*^{(s)} = B(I)$, where $\sum_{i=I+1}^{2^L} c_i^{(s)} \ll 1$, i.e., the outage probability of B_2 is very low. $B_2^{(s)}$ is then obtained as $(\nu_s \lambda_0 T_h + n_2) b_0 + B_*^{(s)}$.

4.2 A Design Example

We illustrate the ideas presented in the previous section with a design example, where $\overline{D}_{st} \le \hat{D} = 2$ minutes, $\lambda_0 = 200$ req/hr (equally distributed among all the local servers), $L = 8$, $S = 10$, and $N_v = 4,000$ titles with each library storing 500 video titles ($C_f = 1,000$ MB, $b_0 = 1.5$ Mb/s and $T_h = 90$ minutes, and $N_{dr}^{(3)} = 1$). We consider only the uniform popularity case, as the nonuniform case is similar in performance, as shown in Section 2.3 above.

To show how much bandwidth can be saved using bandwidth sharing, let's first consider a design using bandwidth partitioning, where the bandwidth in the local server is partitioned for the libraries and the library bandwidth is partitioned for the local servers. Since each local server stores $(\lambda_0/S)T_h + n_2 = 45$ GB, the local miss request rate is $(1 - 45/N_v) \times \lambda_0/S = 19.78$ req/hr and, hence, the request rate to a library from a given local server is $19.78/L = 2.47$ req/hr. Since the average miss delay in a library should be $\hat{D}/(19.78/20) = 2.02$ minutes, the tertiary bandwidth needed for each local server is, using our M/M approximation, 8.93 MB/s and, hence, the total library bandwidth to serve all S local servers is $8.93 \times S = 89.3$ MB/s! This is very high and each partitioned bandwidth is only $2.47\ \text{req/hr}(C_f/8.93\ \text{MB/s})/3,600\ \text{s/hr} = 7.7\%$ utilized. For a local server, since it partitions its bandwidth for each library, the total bandwidth needed for staging is $8.93\ \text{MB/s} \times L = 71.4\ \text{MB/s}$, with each partitioned bandwidth again being only 7.7 percent utilized.

Let's consider now bandwidth sharing. We have $\nu_s = 0.1$ (each server serves 20 req/hr) and, hence, $C_2^{(s)} = 45$ GB, for $s = 1, \ldots, S$. Note that $\eta_l^{(s)} = 0.125$ for any server s, giving $\overline{D}_l^{Miss} = 2.0228$ minutes and $\lambda_3^{(l)} = 24.72$ req/hr. Given $\lambda_3^{(l)}$ and $\overline{D}_l^{Miss}$, we have $B_3^{(l)} = 15$ MB/s, for $l = 1, 2, \ldots, L$ and, henc, $\sigma^{(l)} = 0.4545$. The generator function for server s is therefore

$$\begin{aligned} g_s(y) &= (1 - 0.4545/10 + (0.4545/10)y^{15})^8 \\ &= 0.689 + 0.2626y^{15} + 0.04376y^{30} + 0.004167y^{72} + \ldots, \end{aligned}$$

from which we may take $B_*^{(s)} = 15$ MB/s to achieve less than 5 percent bandwidth outage probability (such a bandwidth can be reduced if a larger local storage is used). Using the $B_*^{(s)}$, we have $B_2^{(s)} = 23.44$ MB/s, for $s = 1, 2, \ldots, S$.

We see from the above that both $B_3^{(l)}$ (= 15 MB/s) and $B_2^{(s)}$ (= 23.44 MB/s) are substantially reduced when compared with the partitioned case.

5 Conclusions

A video system should offer scalable storage in order to accommodate a wide variety of video programs. Such scalability can be achieved by a hierarchical storage system in which the not-so-popular video files are stored in the tertiary level and staged onto the secondary level to be displayed. We have analyzed a number of system operations—a video may be displayed after it is completely staged or while it is being staged (i.e., the "stage-streaming" operation) and the displayed segments of a video may be deleted. By combining these different operations, different levels of user interactive capability can be offered. To achieve higher streaming scalability in the system, multiple storage nodes and streaming nodes can be connected by a network. Video files are staged from the storage nodes to the streaming nodes for display. In this paper, we have presented models of all these systems.

Because users in a hierarchical storage system experience some start-up delay due to queuing in the tertiary level, a hierarchical storage system should be designed to meet a certain (low) delay goal. The design parameters include storage in the secondary level and bandwidth in each level. In this paper, we have presented simple models for the hierarchical storage system for different operations. The models are used to dimension the minimum system requirements to meet a certain user delay goal.

We find that hits enjoy negligible delay and, therefore, the tertiary bandwidth should be dimensioned to meet user delay requirement. If files are displayed after they are completely staged, the drives in the tertiary level should be used in parallel; on the other hand, for the stage-streaming operation, independent tertiary drives should be used. As compared to the case of "complete staging before display," stage-streaming is able to reduce the tertiary bandwidth requirement substantially and trail-deletion can reduce the storage requirement substantially with some cost on bandwidth. We also note that user delay increases quickly as the arrival rate increases beyond the design point, indicating the importance of admission control at that point. Via simulation and by using a popularity model based on real rental data, we show that our design meets the delay requirement. The storage required in the secondary level can be much lower than the total storage of all the files in the system, hence achieving low storage cost with high

scalability. The system requirements also do not change much when the underlying video popularity is changed. We also show how our model can be extended to the design of a distributed storage system in which the library bandwidth is shared among the local servers and vice versa to achieve a great decrease in resource requirements.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their detailed and constructive comments, which greatly improve the content and clarity of the paper. This work was supported, in part, by the Competitive Earmarked Research Grant from the Hong Kong Research Grant Council (HKUST6014/01E and HKUST6199/02E).

REFERENCES

[1] A. Guha, "The Evolution to Network Storage Architecture for Multimedia Applications," *Proc. IEEE Int'l Conf. Multimedia Computing and Systems,* pp. 68-73, June 1999.
[2] T. Kurioka, H. Minami, H. Okuda, J. Numazawa, and A. Yanagimachi, "Television Home Server for Integrated Services—Toward the Realization of ISDB 'Anytime' Services," *IEEE Trans. Consumer Electronics,* vol. 44, pp. 1195-1200, Nov. 1998.
[3] V.O.K. Li and W. Liao, "Distributed Multimedia Systems," *Proc. IEEE,* vol. 85, pp. 1063-1108, July 1997.
[4] T. Little and D. Venkatesh, "Prospects for Interactive Video-on-Demand," *IEEE Multimedia,* pp. 14-24, Fall 1994.
[5] W.J. Bolosky, J.S. Barrera, R.P. Draves, R.P. Fitzgerald, G.A. Gibson, M.B. Jones, S.P. Levi, N.P. Myhrvold, and R.F. Rashid, "The Tiger Video Fileserver," *Proc. Sixth Int'l Workshop Network and Operating System Support for Digital Audio and Video (NOSSDAV '96),* Apr. 1996.
[6] R.L. Haskin and F.B. Schmuck, "The Tiger Shark File System," *Proc. COMPCON '96,* pp. 226-231, 1996.
[7] C.T. Johnson, "The IBM 3850: A Mass Storage System with Disk Characteristics," *Proc. IEEE,* vol. 63, pp. 1166-1170, Aug. 1975.
[8] A. Laursen, J. Olkin, and M. Porter, "Oracle Media Server: Providing Consumer Based Interactive Access to Multimedia Data," *Proc. ACM SIGMOD,* pp. 470-477, 1994.
[9] A. Laursen, J. Olkin, and M. Porter, "Oracle Media Server Framework," *Compcon: Digest of Papers,* pp. 203-208, 1995.
[10] K.C. Almeroth, A. Dan, D. Sitaram, and W.H. Tetzlaff, "Long Term Resource Allocation in Video Delivery Systems," *Proc. IEEE Infocom '97,* pp. 1333-1340, Apr. 1997.
[11] A. Dan, M. Kienzle, and D. Sitaram, "A Dynamic Policy of Segment Replication for Load-Balancing in Video-on-Demand Servers," *Multimedia Systems,* vol. 3, pp. 93-103, July 1995.
[12] C.C. Aggarwal, J.L. Wolf, and P.S. Yu, "The Maximum Factor Queue Length Batching Scheme for Video-on-Demand Systems," *IEEE Trans. Computers,* vol. 50, no. 2, pp. 97-110, Feb. 2001.
[13] S.-H. G. Chan and F. Tobagi, "Trade-Off between System Profit and User Delay/Loss in Providing Video Services with Request Batching," *IEEE Trans. Circuits and Systems for Video Technology,* vol. 11, pp. 916-927, Aug. 2001.
[14] S.-H.G. Chan and F. Tobagi, "Distributed Servers Architecture for Networked Video Services," *IEEE/ACM Trans. Networking,* vol. 9, pp. 125-136, Apr. 2001.
[15] L. Gao, Z.-L. Zhang, and D. Towsley, "Catching and Selective Catching: Efficient Latency Reduction Techniques for Delivering Continuous Multimedia Streams," *Proc. ACM Multimedia '99,* pp. 203-206, Oct.-Nov. 1999.
[16] C.C. Aggarwal, J.L. Wolf, and P.S. Yu, "Design and Analysis of Permutation-Based Pyramid Broadcasting," *ACM/Springer Multimedia Systems,* vol. 7, no. 6, pp. 439-448, 1999.
[17] S.-H.G. Chan and S.-H.I. Yeung, "Client Buffering Techniques for Scalable Video Broadcasting over Broadband Networks with Low User Delay," *IEEE Trans. Broadcasting,* vol. 48, pp. 19-26, Mar. 2002.
[18] S. Ghandeharizadeh, A. Dashti, and C. Shahabi, "A Pipelining Mechanism to Minimize the Latency Time in Hierarchical Multimedia Storage Managers," *Computer Comm.,* vol. 18, pp. 170-184, Mar. 1995.
[19] S. Ghandeharizadeh and C. Shahabi, "On Multimedia Repositories, Personal Computers, and Hierarchical Storage Systems," *Proc. ACM Multimedia,* pp. 407-416, 1994.
[20] R.-H. Hwang and P.-H. Chi, "A Low Cost Optical Storage Server for Near-on-Demand Systems," *IEEE Trans. Broadcasting,* vol. 47, pp. 357-366, Dec. 2001.
[21] M.G. Kienzle, A. Dan, D. Sitaram, and W. Tetzlaff, "Using Tertiary Storage in Video-on-Demand Servers," *Compcon: Digest of Papers,* pp. 225-232, 1995.
[22] A.L. Chervenak, D.A. Patterson, and R.H. Katz, "Choosing the Best Storage System for Video Service," *Proc. ACM Multimedia,* pp. 109-119, 1995.
[23] S.A. Barnett and G.J. Anido, "A Cost Comparison of Distributed and Centralized Approaches to Video-on-Demand," *IEEE J. Selected Areas in Comm.,* vol. 14, pp. 1173-1183, Aug. 1996.
[24] P. Triantafillou and T. Papadakis, "Ondemand Data Elevation in Hierarchical Multimedia Storage Servers," *Proc. 23rd Int'l Conf. Very Large Data Bases,* pp. 226-235, 1997.
[25] S.-L. Tsao, "A Low Cost Optical Storage Server for Near-on-Demand Systems," *IEEE Trans. Broadcasting,* vol. 47, pp. 56-65, Mar. 2001.
[26] P. Sumari, M. Merabti, and R. Pereira, "Video-on-Demand Server: Strategies for Improving Performance," *IEE Proc.-Software,* vol. 146, pp. 33-37, Feb. 1999.
[27] Z.D. Wu, "Design and Analysis of Video-on-Demand Servers," *Proc. IEEE Globecom,* pp. 773-778, Nov. 1998.
[28] S.-L. Tsao and Y.-M. Huang, "Making a Cost-Effective Storage Server for Broadcasting Digital Video Services," *IEEE Trans. Broadcasting,* vol. 44, pp. 300-307, Sept. 1998.
[29] E. Chang and H. Garcia-Molina, "Reducing Initial Latency in Media Servers," *IEEE Multimedia,* vol. 4, no. 3, pp. 50-61, July-Sept. 1997.
[30] S. Chen and M. Thapar, "A Novel Video Layout Strategy for Near-Video-on-Demand Servers," *Proc. IEEE Int'l Conf. Multimedia Computing and Systems '97,* pp. 37-45, June 1997.
[31] A.N. Mourad, "Issues in the Design of a Storage Server for Video-on-Demand," *ACM/Springer Multimedia Systems,* vol. 4, pp. 70-86, Apr. 1996.
[32] H.M. Vin, A. Goyal, and P. Goyal, "Algorithms for Designing Multimedia Servers," *Computer Comm.,* vol. 18, pp. 192-203, Mar. 1995.
[33] H.M. Vin and V. Rangan, "Designing a Multiuser HDTV Storage Server," *IEEE J. Selected Areas in Comm.,* vol. 11, pp. 153-164, Jan. 1993.
[34] J. Wilkes, R. Golding, C. Staelin, and T. Sullivan, "The HP Auto RAID Hierarchical Storage System," *ACM Trans. Computer Systems,* vol. 14, pp. 108-136, Feb. 1996.
[35] C. Ruemmler and J. Wilkes, "An Introduction to Disk Drive Modeling," *Computer,* pp. 17-29, Mar. 1994.
[36] L. Golubchik, R. Muntz, and R. Watson, "Analysis of Striping Techniques in Robotic Storage Lbraries," *Proc. IEEE 14th Symp. Mass Storage Systems,* pp. 225-238, Sept. 1995.
[37] D.A. Ford, R.J.T. Morris, and A.E. Bell, "Redundant Arrays of Independent Libraries (RAIL): A Tertiary Storage System," *Proc. COMPCON '96,* pp. 280-285, 1996.
[38] S.-W. Lau and J.C.S. Lui, "Scheduling and Data Layout Policies for a Near-Line Multimedia Storage Architecture," *ACM/Springer Multimedia Systems,* vol. 5, pp. 310-323, Sept. 1997.
[39] R. Boutaba and A. Hafid, "A Generic Platform for Scalable Access to Multimedia-on-Demand Systems," *IEEE J. Selected Areas in Comm.,* vol. 17, pp. 1599-1613, Sept. 1999.
[40] S.M. Poon, B.S. Lee, and C.K. Yeo, "A DAVIC-Based Video-on-Demand System over ATM Networks," *IEEE Trans. Consumer Electronics,* vol. 45, pp. 345-355, May 1999.
[41] T.-H. Wu, I. Korpeoglu, and B.-C. Cheng, "Distributed Interactive Video System Design and Analysis," *IEEE Comm. Magazine,* pp. 100-108, Mar. 1997.
[42] D.W. Brubeck and L.A. Rowe, "Hierarchical Storage Management in a Distributed VOD System," *IEEE Multimedia,* pp. 37-47, Fall 1996.

[43] P. Mundur, R. Simon, and A. Sood, "Integrated Admission Control in Hierarchical Video-on-Demand Systems," *Proc. IEEE Int'l Conf. Multimedia Computing and Systems,* pp. 220-225, June 1999.
[44] L.D. Giovanni, A.M. Langellotti, L.M. Patitucci, and L. Petrini, "Dimensioning of Hierarchical Storage for Video on Demand Services," *Proc. IEEE Int'l Computing Conf. (ICC '94),* pp. 1739-1743, 1994.
[45] Y. Won and J. Srivastava, "Minimizing Blocking Probability in a Hierarchical Storage Based VOD Server," *Proc. Int'l Workshop Multimedia Database Management Systems,* pp. 12-19, Aug. 1996.
[46] S.B. Jun and W.S. Lee, "Video Allocation Methods in a Multi-Level Server for Large-Scale VOD Services," *IEEE Trans. Consumer Electronics,* vol. 44, pp. 1309-1318, Nov. 1998.
[47] Y.-C. Lai, Y.-D. Lin, and H.-Z. Lai, "A Hierarchical Network Storage Architecture for Video-on-Demand Services," *IEEE Trans. Broadcasting,* vol. 43, pp. 145-154, June 1997.
[48] J.-P. Nussbaumer, B.V. Patel, F. Schaffa, and J.P.G. Sterbenz, "Networking Requirements for Interactive Video on Demand," *IEEE J. Selected Areas in Comm.,* vol. 13, pp. 779-787, June 1995.
[49] G. Bianchi and R. Melen, "The Role of Local Storage in Supporting Video Retrieval Services on ATM Networks," *IEEE/ACM Trans. Networking,* vol. 5, pp. 882-892, Dec. 1997.
[50] H. Suzuki and K. Nishimura, "Performance Analysis of a Storage Hierarchy for Video Servers," *Systems and Computers in Japan,* vol. 28, pp. 11-20, June 1997.
[51] A. Merchant, Q. Ren, and B. Sengupta, "Hierarchical Storage Servers for Video on Demand: Feasibility, Design and Sizing," *Proc. IEEE Globecom '96,* pp. 272-278, Nov. 1996.
[52] S.-H.G. Chan and F.A. Tobagi, "Hierarchical Storage Systems for On-Demand Video Servers," *Proc. SPIE High-Density Data Recording and Retrieval Technologies,* pp. 103-120, Oct. 1995.
[53] L. Kleinrock, *Queueing Systems: Theory,* vol. 1. Wiley Interscience, 1975.
[54] S.-H.G. Chan, "Scalable Services for Video-on-Demand," PhD dissertation, Dept. of Electrical Eng., Stanford Univ., Jan. 1999.

S.-H. Gary Chan (S'89-M'98) received the PhD degree in electrical engineering with a minor in business administration from Stanford University, Stanford, California, in 1999, and the BSE degree (highest honors) in electrical engineering from Princeton University, Princeton, New Jersey, in 1993. He is currently an assistant professor with the Department of Computer Science, The Hong Kong University of Science and Technology, Hong Kong, and an adjunct researcher with Microsoft Research Asia in Beijing. He was a visiting assistant professor in networking at the Department of Computer Science, University of California, Davis, from September 1998 to June 1999. During 1992-1993, he was a research intern at the NEC Research Institute, Princeton, New Jersey. His research interests include multimedia networking, peer-to-peer networks, high-speed and wireless communications networks, and Internet technologies and protocols. He was a William and Leila Fellow at Stanford University during 1993-1994. At Princeton, he was the recipient of the Charles Ira Young Memorial Tablet and Medal and the POEM Newport Award of Excellence in 1993. He is a member of Tau Beta Pi, Sigma Xi, Phi Beta Kappa, and the IEEE.

Fouad A. Tobagi (M'77-SM'83-F'85) received the engineering degree from the Ecole Centrale des Arts et Manufactures, Paris, France, in 1970 and the MS and PhD degrees in computer science from the University of California, Los Angeles, in 1971 and 1974, respectively. From 1974 to 1978, he was a research staff project manager with the ARPA project at the Computer Science Department, University of California, Los Angeles, and engaged in research in Packet Radio Networks, including protocol design and analysis and measurements of packet radio networks. In June 1978, he joined the faculty of the School of Engineering at Stanford University, Stanford, California, where he is a professor of electrical engineering and computer science. In 1991, he cofounded Starlight Networks, Inc., a venture concerned with multimedia networking, and served as Chief Technical Officer until August 1998. His research interests have comprised packet radio and satellite networks, local area networks, fast packet switching, multimedia networking and networked video services, and multimedia applications. His current research interests include voice and video communication over the Internet, wireless and mobile networks, network design and provisioning, and network resource management. He is a fellow of the IEEE for his contributions in computer communications and local area networks. He is the winner of the 1981 Leonard G. Abraham Prize Paper Award in the field of Communications Systems for his paper "Multiaccess Protocols in Packet Communications Networks" and cowinner of the IEEE Communications Society 1984 Magazine Prize Paper Award for the paper "Packet Radio and Satellite Networks." He served as associate editor for computer communications for the *IEEE Transactions on Communications* for the period 1984-1986, editor for packet radio and satellite networks for the *Journal of Telecommunications Networks* for the period 1981-1985, coeditor of the special issue on local area networks of the *IEEE Journal on Selelected Areas in Communications* (November 1983), coeditor of *Advances in Local Area Networks*, a book in the series Frontiers in Communications (New York: IEEE Press), coeditor of the special issue on packet radio networks of the *Proceedings of the IEEE* (January 1987), and coeditor of the special issue on large scale ATM switching systems for B-ISDN of the *IEEE Journal on Selected Areas in Communications* (October 1991). He is currently serving as an editor for a number of journals in high speed networks, wireless networks, multimedia, and optical communications. He is a member of the ACM and served as an ACM national lecturer for the period 1982-1983. He is corecipient of the 1998 Kuwait Prize in the field of applied sciences.