HIERARCHICAL STORAGE SYSTEMS FOR INTERACTIVE VIDEO-ON-DEMAND

Shueng-Han Gary Chan and Fouad A. Tobagi

Technical Report No. CSL-TR-97-723

April 1997

This work was supported in part by NSF under grant NCR-9016032, by Pacific Bell, and by the Center for Telecommunications at Stanford University.

Hierarchical Storage Systems for Interactive Video-On-Demand

Shueng-Han Gary Chan and Fouad A. Tobagi

Technical Report: CSL-TR-97-723

April 1997

Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
William Gates Computer Science Building, 4-A
Stanford, CA 94305-9040
pubs@shasta.stanford.edu

Abstract

On-demand video servers based on hierarchical storage systems are able to 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. The design of video servers allowing user interaction with the playbacked video is of great interest. We have conducted a comprehensive study on the architecture and operation of such servers based on hierarchical storage systems for interactive video-on-demand. Our objective is to understand its performance characteristics, so as to design a video server to meet specific application requirements. The applications of interest are many: distance-learning, movie-on-demand, interactive news, home-shopping, etc.

The design of such a server actually involves many design choices pertaining to both architecture and operational procedures. As far as architecture is concerned, we need to consider such system parameters as bandwidth, storage capacity, and the number of drives at both secondary and tertiary storage levels. As far as operational procedures are concerned, we need to consider policies for request admission and scheduling, video file replacement, and bandwidth allocation in the system.

We have addressed the performance and design issues of a video server based on a hierarchical storage system. Since there are so many design choices to be considered, we first study through simulation a baseline system using First-Come-First-Served scheduling policies and Least-Recently-Used replacement policy. The baseline allows us to capture the essential performance characteristics of a hierarchical storage system. Then we extend our study beyond the baseline to cover numerous other system variations in terms of architectural parameters and operational procedures. We have also examined various application characteristics, such as file size and video popularity, on the system performance. We demonstrate the usefulness of our results by applying them to the design of a hierarchical storage system, taking into account current state of storage technologies.

Key Words and Phrases: Video-on-demand, video server, hierarchical storage systems, I/O architecture, scheduling policies, multimedia systems

Contents

1 Introduction

On-demand video services, or video-on-demand (VOD) for short, refer to services in which a user is able to request a video on demand. VOD allows users tailor their own viewing programs according to their tastes and time schedules. It encompasses many applications such as movie-on-demand (MOD), distance learning, interactive news (news-on-demand), home/catalogue shopping, various interactive/distributed training programs, etc [1, 2, 3].

Most video servers reported in the literature or existing today use magnetic disks for both video storage and presentation. Magnetic disk technology is used due to its high throughput, low access latency, random data access, and adequate storage capacity [4, 5, 6]. However, magnetic disks are still not ideal to store large volume of video files (e.g., $>$ 1000s video files for some applications). This is because even though array of disks promises high storage capacity, there are still reliability and scalability issues to be resolved before it can accommodate comfortably terabytes of data. Furthermore, magnetic disks nowadays still suffers comparatively high cost — currently, storing a single 90-minutes movie ($\simeq$ 1 GB) in such medium can cost as much as 300 dollars. As video files can also differ markedly in their popularities, some of them can be infrequently accessed. Storing all files on-line regardless of their popularities is therefore not very cost-effective [7].

Video servers based on hierarchical storage system have therefore been proposed to provide a cost-effective solution. They consist of both tertiary and secondary levels. Tertiary storage, commonly referred as library or "jukebox," is used to store all video files. The files are transferred or "staged" on demand into the secondary level to be displayed. The secondary level in the hierarchical storage system therefore acts as a "caching" or "staging" platform for video display [8, 9].

Table 1 shows typical performance characteristics for secondary and tertiary levels. From the table, we see that, 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 a tertiary library can shelve hundreds or even thousands of removable

Table 1: Typical performance characteristics for the secondary and tertiary storage levels

	Secondary level	Tertiary level	
	Magnetic disks	Optical disks	Tapes
Media cost (/GB)	\$300	\$10 – \$100	\$3 – \$10
Storage capacity	$\leq$ 100GB	$\geq$ 1TB	$\geq$ 1TB
Raw BW (MB/s)	2 – 10	1 – 4	1 – 4
Total access time[a]	10 ms	$\sim$ 20 sec.	$\sim$ 1 min.

[a]In case of tertiary level, total access time is the sum of both exchange time and file seek time.

tapes or disks, with each tape or disk offering high storage capacity ($\sim$10 – 100 GB), a tertiary system holds large storage capacity in the range of terabytes. Therefore, such a hierarchical storage system offers low cost high capacity storage as compared with a system based on magnetic disks only.

VOD applications differ widely in their characteristics, such as file sizes and popularity. Some applications also need to offer high interactive capability to their users. In this paper, we consider the design of a hierarchical storage system for interactive video-on-demand (VOD). Here, "interactive" means once users see their videos, they are free to interact with the videos from the beginning to the end, using commands such as fast-forward (FF), rewind (RW), pause, etc. Such interactive VOD offers the greatest flexibility to the users. The design of such a video server is therefore of greatest appeal in a highly interactive environment. As a result, we will not consider algorithms or system operations which do not offer this capability here [10, 11].

The design of a server based on a hierarchical storage system actually involves many architectural parameters such as secondary level bandwidth, secondary level storage capacity, tertiary level bandwidth, number of drives, etc., and many operational procedures including admission control, request scheduling and replacement policies. Designing such a server therefore requires a good understanding of the inter-dependence of these parameters and algorithms.

We have studied through simulation the performance and design issues of an interactive on-demand video server based on hierarchical storage system by taking into account various application characteristics. Our objectives are to understand its performance characteristics, so as design a video server to meet specific application requirements.

This paper is organized as follows. In Section 2, we describe VOD applications characteristics and the performance goals as pertaining to video server design. In Section 3, we describe a hierarchical server architecture and its operation. We have conducted a comprehensive study on a video server for interactive VOD, and specify in Section 4 some of the possible operational procedures that can be used. In Section 5, we report on the performance of such a system, by first considering a baseline case and then extend it to other variations. In this section, we address the scalability, trade-off and sensitivity issues of the system. Finally, in Section 6, we provide methodologies on how a video server based on hierarchical storage system can be designed to meet a specific delay requirement, taking into account current storage technologies.

2 VOD Applications Characteristics and Performance Goals in Server Design

In VOD, a user first selects a video file (such as a movie, an advertisement, or a piece of news) in a relevant video server. There are three outcomes of the request depending on the availability of resources (such as streaming bandwidth or storage capacity) at that time: i) If there is enough resources, the request is accepted and streamed; otherwise, the request is either ii) rejected and hence lost, or iii) accepted and asked to wait till resources are available. The users will then have to decide whether to wait or try back later.

Once a file is displayed to a user, the user usually can to a certain extend interact with the videos through fast-forward, rewind, pause, etc. However, users generally are not allowed to modify or change the content of the video file, i.e., the video file is read-only. After serving a user, the system updates various information, such as its database, accounting information, and, if video contents are updated or created, writes back the relevant information to the server.

2.1 Applications characteristics

In order to design a practical and useful video server, we need to understand the characteristics of VOD applications. We discuss here four applications characteristics: demand characteristics, video files characteristics, user interactions, and performance requirements.

Demand characteristics: Demand characteristics include three attributes: the requests arrival process, size of each arrival, and the request's holding time.

- Arrival process — An arrival consists of one or more simultaneous video requests, each of which asks for the display of a certain video in the database. In other words, each request demands a video stream from the video server. In a VOD environment, the arrival process can be very complicated. For example, the arrivals may be clustered in a classroom environment. For MOD or interactive news, the process may vary according to the time of day, day of week, or month of year. It may also be affected by external news (such as Olympic games, good or bad news around the world, or academy awards).

  Finding a realistic arrival process including both long-term and short-term variations is generally difficult, and is often possible only by undergoing some field trials or experiments. The Poisson process is generally used to model the arrival process for large number of users. For small number of users (e.g., local services), the inter-arrival time may be modeled as exponentially distributed with variable rate depending on the number of users currently seeing or waiting to see their videos. More general arrival processes can be modeled as series-parallel combination of exponential processes, such as hyperexponential distribution, Erlangian distribution, Cox-phased networks, etc [12, 13].

- 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. The traffic of most applications are characterized by one-at-a-time arrival. Multiple requests per arrival is also known as "bulk arrival." Some applications can have bulk arrivals, such as distance learning or multiuser training programs, in which multiple streams may have to be open at the same time.
- Request's holding time — Request holding time is the total time a user occupies a video stream, for either seeing or interacting with the video. Depending on the particular application, the request holding time may be somewhat random (e.g., as in interactive news), or relatively deterministic (e.g., as in movie-on-demand).

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 streaming bandwidth of a video, b_0, depends on the video compression scheme used (e.g., MPEG-I, MPEG-II, motion-JPEG, etc.). It can range from lower than 1 Mbps to more than 10 Mbps. Streaming bandwidth of a displayed video may also vary depending on encoding methods (e.g., Constant-bit-rate, Variable-bit-rate, etc.).
- Size of the video files — The size of a video file is the actual storage space the file consumes in a storage medium. It may range from $\sim$ 10 MB (advertising clips) to more than $\sim$ 1 GB (movies). All files in a VOD application may not be of the same size. In MOD (movie-on-demand), for example, the file size is likely to be similar or "homogeneous," with each file of about, say, 90 minutes playback time. On the other hand, file size in the interactive news environment can be rather "heterogeneous," depending on the piece of news and whether it is a documentary news or not. Somewhere between the "extremes" may be home shopping, in which file size may range from $\sim$ 5 MB to $\sim$ 30 MB (20 seconds to 2 minutes).

Note that the size of a video file does not necessarily relate to the request's holding time. This is because users may finish their displays at any time. Furthermore, some parts of the video can be repeated many times, other parts may be skipped (e.g., by fast forwarding or jump command), and the user may pause at any time. Therefore, file size does not have to be directly proportional to the holding time.

- Number of video titles — Applications for specialized users tends to have fewer titles than applications for general public. Furthermore, applications targeted to a large number of users are likely to have more titles than applications for a smaller 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 defined as a number proportional to the video popularity.

  If all of the video titles are equally likely to be chosen by an incoming request, we say the video popularity is uniform, i.e., we have uniform video popularity. 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 [14, 15, 16]: In this model, the popularity of the ith video is proportional to $1/i^{\zeta}$, where ζ is a (real) parameter known as Zipf parameter. Most models in the literature take ζ to be 1.0, though according to some actual rental data, $\zeta = 0.271$ may be more likely [15]. Given ζ, we can then express p_i as:

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

where

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

– Geometric distribution [17, 18]: In this model, the ratio of p_i to p_{i-1} is equal to a constant known as "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

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

User interactions: In VOD, after a video is displayed, the user may be able to interact with the video sequence. We describe here four attributes of user interactions: i) modifiability of video files, ii) the types of interactions; iii) the frequency of interactions; and iv) the locality of interaction.

i) Modifiability of video files — An application may not want or allow the general users to modify its video files. These read-only applications actually encompass most of VOD applications, such as MOD, home-shopping, interactive news, etc. As the users are not allowed to edit the files, there is no write-back traffic due to file modification on the server. On the other hand, in a video editing environment, video files are constantly being updated or created. Video servers for such purpose may have to be designed differently, because of the issues in write-back traffic and placement of data-blocks.

ii) Types of interactions — Types of interactions refers to the user commands in controlling the video displayed. Different VOD applications have different types of interactions. In MOD or distance learning, for example, users would most likely interact with the videos using VCR commands such as FF (fast-forward) and RW (rewind). In home-shopping, point-and-select interactive commands will most likely be characteristic. In video-authoring environment, functions similar to "cut and paste" may have to provided.

iii) Frequency of interactions — Interaction frequency refers to how often a user interacts with the video displayed. Different applications may vary markedly in terms of the interaction frequency. For example, the frequency in MOD would be lower (possibly $\leq$ 0.1 interaction/min.) than that of home shopping ($\sim$ 1 interaction/min.).

iv) Locality of interaction — Locality of interaction refers to the extent of temporal displacement of each user interaction using VCR commands such as a rewind, fast-forwards, etc. Certain video, such as a movie or a lecture, is displayed in a certain sequence if the user does not interact with it. Once a user starts to interact with the video, the sequence is changed and the video is playbacked at a different point in length of the video. An interaction is said to be "localized" if such new display point is temporally close to the point prior to the interaction. Some applications, such as MOD, are most likely characterized by localized interactions, while some others (e.g., interactive news and home-shopping) are expected to exhibit low locality of interaction.

Performance requirements: Different applications have different performance requirements. We list here five such requirements: i) start-up delay, ii) user interactions, iii) streaming capacity, iv) rejection/blocking probability and v) fairness.

i) Start-up delay — We define "start-up delay," D_{st}, as the waiting time from the moment when a user initially submits a video request until the moment when the user begins to see the video. It is therefore the total waiting time for the requested video to be streamed. Obviously, D_{st} is a random variable whose value can depend on where the user is in the queue, what the user's class is, and even which video the user requests. We distinguish here start-up delay with the response time of user interactions. While start-up delay is the waiting time of a user *before* the requested video is displayed, the response time of user interactions is the latency from the issue of a control command to the actual scene change in an *on-going* video session. Therefore, start-up delay may be much longer than the response time of user interactions.

For performance evaluation, one sometimes would consider the average start-up delay over all requests in the steady state, $\overline{D}_{st}$, or for a certain request class i, $\overline{D}_{st}^{(i)}$. A bounded start-up delay or more deterministic delay (in which users are similarly delayed) may also be of interest.

Different VOD applications have different minimum requirement for start-up delay. The requirement may depend on how long a user sees the video, i.e., the request's holding time. For example, in MOD in which the holding time is relatively long ($\simeq$ 90 min.), start-up delay can be as high as several minutes. However, when a user is still not sure what file to be displayed and wants some kinds of "preview" before making up his/her mind, the start-up delay for these "preview" video clips should be much lower and more stringent, possibly in the range of seconds. For home-shopping or news-on-demand applications, on the other hand, the start-up delay should be bounded to within several seconds.

While long start-up delay is undesirable, generally users would be able to wait longer if the followings are provided:

- Delay guarantee: Users may be more willing to wait if they are sure that they can watch their videos at a particular time, even if the time is possibly minutes or hours later. This is the principle behind delay guarantee systems, such as deterministic delay (i.e., users experience similar delay) or reservation system, in which users reserve videos to be displayed at a certain later time.
- Variance reduction and delay estimation: Generally, users are more willing to wait if they know approximately how long they have to wait, i.e., when the uncertainty in waiting time is reduced. Providing a good estimate of how long a user should wait before the display of his/her video is therefore very valuable.

  - Delay discount service: In VOD, users generally pay by, for example, pay-per-view, pay-per-time, or flat subscription rate. Users are usually willing to pay for the quality of service they receive, and hence they are more willing to wait if they can enjoy some kinds of service discount.

ii) User interactions — We list two such requirements here: response time of the interactions and control granuality of the interactions.

  - Response time of the interactions: The elapsed time between a user's interactive command (e.g., FF, RW, etc.) and the actual change of the display scene is called the response time of the interaction. Different applications have different requirement in the response time. For interactive news, the response time would be rather low, in the range of a second or so, while for MOD, the response time requirement can be relaxed.
  - Control granularity of user's interactions: In some VOD applications, users are allowed to view (e.g., using fast-forward or rewind) a particular point in the video sequence or pause/start at a particular time. For some applications, especially for MOD or distance learning applications, the point at which users can visit or pause/start does not have to be exact, i.e., the interaction point can be discrete. A coarse interaction "granularity" would not be acceptable for some applications such as home-shopping, whose granularity may be required to be at most couples of seconds. In other words, a system with very fine interaction granularity allows users to go to or pause at virtually any time, while a system with coarse granularity gates the users to interact at some fixed, specified time.

iii) Streaming capacity — Streaming capacity is the maximum number of concurrent users or the maximum request rate that a server can handle, under certain application performance requirements. Different applications lead to different streaming capacity requirement. A large VOD application, such as movie-on-demand and distance learning, may have 100s of concurrent users, while a smaller local VOD applications (for company training purpose) would have fewer users, e.g., 10–100 concurrent users.

iv) Rejection/Blocking probability — If an incoming video request cannot be served immediately, the request is blocked. The blocking may be due to a lack of resources (such as bandwidth or storage space), or according to some priority schemes. A blocked request may either be put into a queue, and hence it has to wait for later service, or be rejected for service, and hence is lost. A low rejection/blocking rate is essential in providing high quality of service to the users.

v) Fairness — The scheduling policies used in a VOD server should be fair in the sense that they should not discriminate between two users of equal service class. For example, in MOD, a user happens to request an unpopular movie should not be discriminated unduly against a user requesting a more popular movie, if both of them pay the same.

2.2 Performance goals in video server design

We state here three performance goals in designing a video server:

Meeting the applications performance requirements: Needless to say, the primary goal of a video server design is to meet specific applications requirements. Such requirements include start-up delay, user interactions, streaming capacity, etc., as stated above.

Cost-effectiveness: The design of video server involves trade-offs between many system parameters, such as storage capacity and bandwidth. A video server should satisfy specific applications requirements with the lowest cost, i.e., it should be *cost-effective.* The cost of a VOD server consists of many components such as storage cost (e.g., the cost of storage elements such as disks or tapes used), bandwidth/communication cost (e.g., the cost of disks/tapes drives, bus bandwidth, etc.), software cost (e.g., the

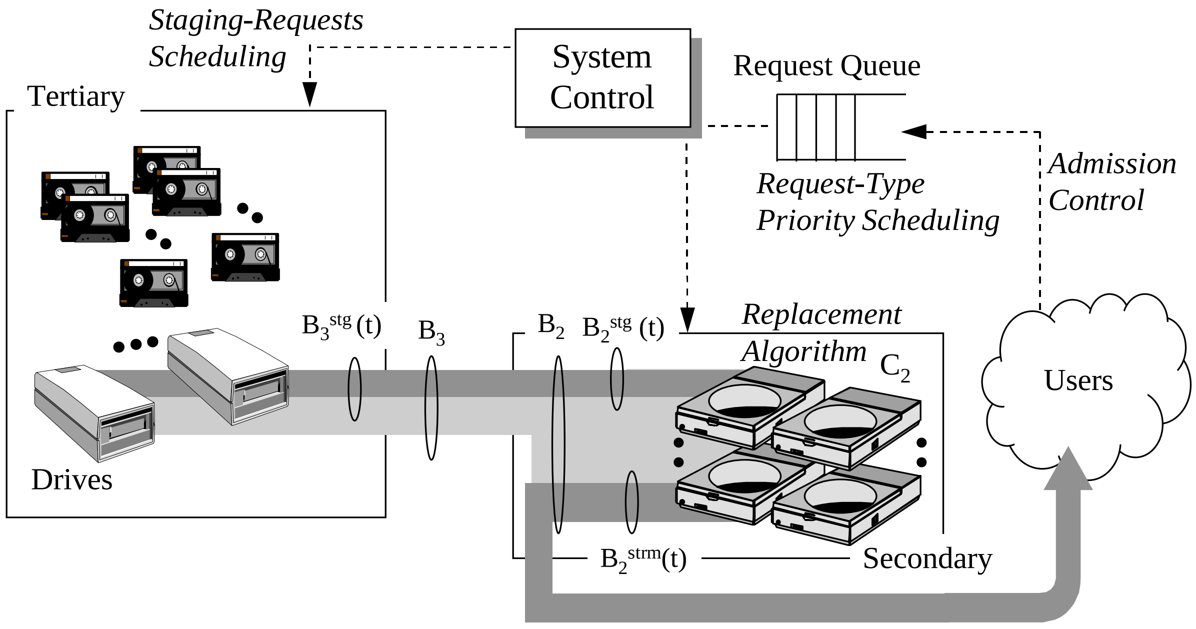

Figure 1: A hierarchical storage system for an on-demand video server

cost of development or software complexity), the cost of the associated peripherals (e.g, the cost of interface units or other hardware), etc. Optimal server design should minimize the cost while meeting performance requirements.

Robustness/Scalability: A VOD server should perform well under environmental changes, i.e., it should be designed to be "robust". The changes can be an increase in video titles, a skew in video popularity, or a rise in traffic intensity. A server should also be designed so as to be scalable, i.e., the server should be able to expand easily so as to accommodate an increase in application requirements (such as start-up delay, user interaction, streaming capacity, etc.).

3 The Architecture of a Hierarchical Storage System

Figure 1 shows the architecture of the hierarchical storage system for an on-demand video server. It consists of a tertiary storage level and a secondary storage level described as follows.

Secondary level: The secondary storage level consists of multiple magnetic disks, though rewritable optical disks may also be used. The level is characterized by high throughput and fast data access, and hence is particularly suitable for video streaming. A number of disks in this level can be configured to form a single logical unit known as a disk array, or RAID (Redundant array of inexpensive disks) [19, 20]. In this way, data can be accessed in the array as if it were a single disk volume. There are different RAID levels, each corresponding to how data are laid out and accessed in an application [21, 22, 23].

As a first-step performance study on the hierarchical storage system, we characterize the secondary level by its total storage capacity, C_2, and its total aggregate "effective" bandwidth, B_2. The bandwidth B_2 has taken into account the access overheads in the disks (which is related to assess latency and block size of the disks).

The secondary level is used for displaying or streaming videos to the users. At any particular time t, a portion of B_2 is used for video streaming ($B_2^{strm}(t)$), and another portion is used for video staging ($B_2^{stg}(t)$). Obviously, we must have,

$$B_2^{stg}(t) + B_2^{strm}(t) \quad \leq \quad B_2. \tag{4}$$

As the secondary level has limited storage capacity C_2, not all video titles can be stored at the level simultaneously. Using file replacement and staging mechanisms, video files are able to "share" the limited storage space in order to be displayed to the users.

Tertiary level: Tertiary storage, also commonly known as the "automated library" or "jukebox," is characterized by three components:

- The cabinet of removable media — A tertiary library shelves many ($\geq\sim$ 100) removable storage media, such as tapes (magnetic tape or optical tape) or optical disks (e.g., CD, WORM, MO disks, or Phase-change disks, etc.). As each tape or disk has a high storage capacity (in the range of 10 – 100 GB), the total storage capacity in this level can be greatly enhanced by shelving more tapes or disks. Hence, the tertiary level can offer highly-scalable storage in excess of terabytes.

- The number of drives — The large number of tapes or disks in the library share a limited number of drives. The number, $N_{dr}^{(3)}$, typically ranges from 2 – 16.

  Each drive comes with a certain "effective" bandwidth (taking into account file access overhead and block size), b_3; hence at any time the maximum bandwidth the tertiary storage level can deliver, denoted by tertiary bandwidth B_3, is given by $b_3 N_{dr}^{(3)}$. A number of tertiary drives can be configured as if they form a single drive (similar to a disk array) through fine-grained striping [24, 25]. In this way, the bandwidth of the drives can be used in parallel to deliver a file and hence delivery bandwidth of a file is increased. As the drives are no longer independent in this case, it is as if $N_{dr}^{(3)} = 1$. Such a configuration achieves maximal bandwidth parallelism (using all the bandwidths at the same time), but minimal request concurrency (serving multiple requests at the same time). On the other hands, multiple independent drives achieve higher request concurrency but less bandwidth parallelism.

- An automated mechanism — The tapes or disks in the library have to share the limited number of drives by being frequently swapped in and out of the drives. Such swapping or exchange is done through some robotic or automated mechanisms. There are N_{rbt} robots in the library. In this paper, we consider $N_{rbt} = 1$ (which is generally the case in reality). The total time it takes for an exchange is called exchange latency, T_{ex}. Note that T_{ex} does not include file seek time, which has been taken into account when defining the effective drive bandwidth, b_3.

All video files are stored in the tertiary level. Video files which need to be displayed but are absent from the secondary level have to be staged from this level by using a certain amount of tertiary bandwidth B_3 ($B_3^{stg}(t)$) and also a portion of the secondary bandwidth B_2 ($B_2^{stg}(t)$). We obviously must have

$$
\begin{aligned}
B_3^{stg}(t) &\le B_3, && (5)\\
B_3^{stg}(t) &= B_2^{stg}(t). && (6)
\end{aligned}
$$

4 An Interactive Video Server Based on Hierarchical Storage System

"Interactive" here means that, once a video is being displayed, the user has complete freedom in interacting with the video from the beginning to the end, i.e., users have complete control with the entire video. Such a system therefore offers the highest flexibility to the users, and hence is of great appeal in highly interactive environments.

Due to the random access and low latency nature in the secondary level, one of the ways achieving interactivity is to require the entire video to be in the secondary level before the beginning of its display. A disadvantage of an interactive system is therefore that a user suffers longer start-up delay because of having to wait for the requested video files to be completely transferred in the secondary level.

In this study, we consider file-by-file staging whereby a file in a tertiary drive is staged to its completion before it is swapped out of the drive and replaced by another file to be staged. In other words, we consider a non-preemptive service discipline. This staging mechanism minimizes swapping compared with block-by-block staging, in which only a portion of a file is staged before it is swapped out of the drive. Therefore, the exchange overheads, which may be significant in the tertiary level, is minimized. Furthermore, as a file is transferred to its completion before another file is staged in the same drive, file-by-file staging reduces start-up delay for a user in an interactive system.

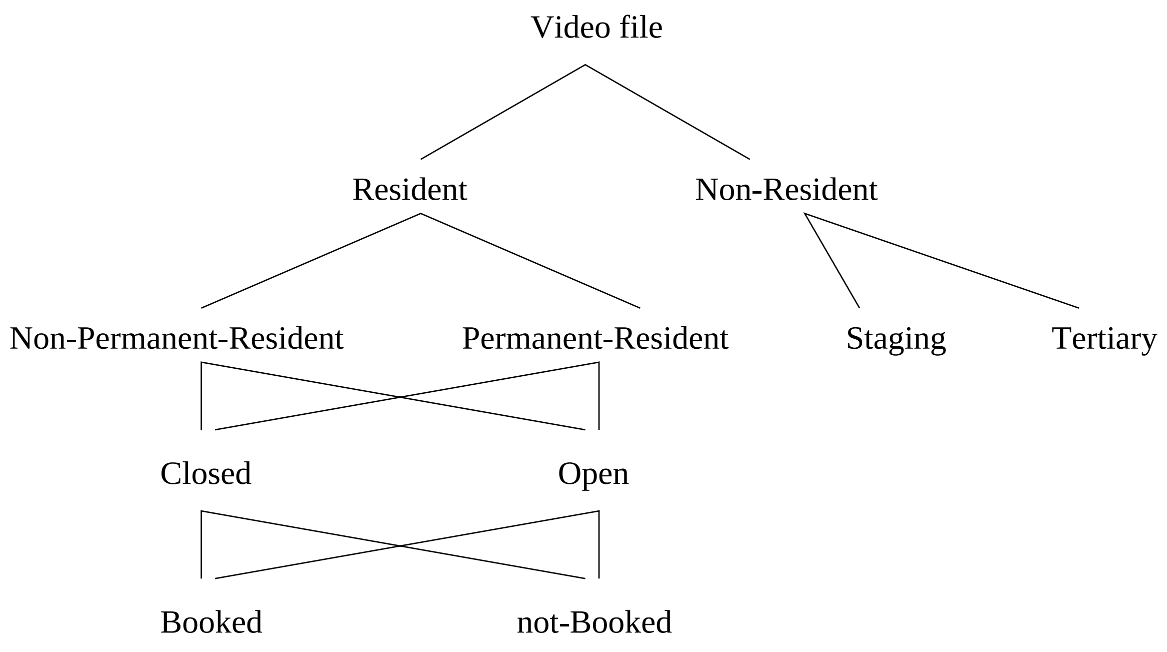

Figure 2: Classification of video files in the hierarchical storage system

4.1 Video files classification and characteristics

Video files in an interactive video server based on hierarchical storage system can be classified according to Fig. 2. There are two types of video files in the system: i) Non-resident files, and ii) Resident files.

Non-resident files are files that are not entirely present in the secondary level. There are two types of non-resident files:

- "Staging" files, files already committed to be staged, i.e., currently being exchanged or being staged. These files are hence only partially present in the secondary level; and

- "Tertiary" files, files entirely absent in the secondary level.

Resident files are entirely present in the secondary level. Such files can be divided into two categories:

- Permanent-resident files — These are files residing permanently in the secondary level and hence can never be deleted; and

- Non-permanent-resident files — These files may be deleted, and whose space can be re-used for staging purpose.

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

Besides "open" or "closed," a resident file can also be called "booked" or "not-booked." A "booked" video file is a file which is being requested by one or more items in the request queue, while a "not-booked" video file is a file which is not being 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 such that the file it is asking is a resident video file, while a miss request is a request such that the file it is asking is a non-resident video file. There are two types of misses:

- If the file requested is a staging file, we call the miss a "staging miss;"
- If the file requested is a tertiary file, we call the miss an "absence miss." Note that an absence miss is also a staging request.

Obviously, only video files corresponding to absence misses are needed to be staged.

A video request is defined to be "served" if:

- the requested video is being displayed to the user; or
- the requested video file is currently being exchanged or being staged from the tertiary level onto the secondary level.

A request is said to be "blocked" or "delayed" if it cannot be served immediately. Such a request may be lost due to:

- 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
- User reneging — The waiting user gives up waiting in the queue and leaves the system.

4.2 System operation

The operation of the hierarchical storage system is described as follows: A user requests a certain video to be displayed. According to an *Admission Control Scheme*, the request is either accepted or denied. Request admission may be regulated by the request queue, which may only hold a certain maximum number of requests waiting for streams allocation. All "overflown" requests are lost. At any point in time, since there are two request types (hits and misses) in the queue, these request types can be scheduled according to a *Request-Type Priority Scheduling* algorithm. Requests are taken out of the request queue once their videos begin to be displayed.

All staging requests are served according to a *Staging-Request Scheduling Algorithm*, which selects a particular files to be staged from the tertiary level. After the file is chosen, upon the availability of a free library robot and an idle drive, the corresponding tape/disk is loaded into a drive. After the tape is loaded, video staging proceeds when the the following two resources in the secondary level are available:

1. Staging bandwidth — Before video staging takes place, not only has the corresponding tape/disk been already loaded in a tertiary drive, the secondary level must first have some bandwidth available for staging the video. This bandwidth is called "staging bandwidth."

2. Staging room — There must be enough storage space to contain the in-coming file. Such space is called "staging room."

Each request is served by assigning a certain secondary bandwidth to it. The assignment can be either a streaming bandwidth if a video is displayed to its user, or a staging bandwidth in which case the requested file is being staged. After a video is completely staged, it can then be streamed.

The secondary level gets its staging room by deleting some of its data. The data blocks are deleted according to a *Replacement Algorithm.* Staging begins after such deletion. After a certain "staging time," the video blocks from the tertiary level will be completely staged and the video can be displayed to the user.

4.3 Operational procedures

We elaborate in this section on the operational procedures chosen in our study.

- Admission control schemes — Admission control algorithms are used to guarantee a certain level of server performance [26]. As a first step in the performance study of a hierarchical storage system, we will suppress its effect here so as to focus on other more important parameters. Therefore, we are not going to investigate admission control for the time being, i.e., we consider Blocked-Call-Held (BCH) algorithm in this paper. Therefore, we accept all the incoming requests and put them into the request queue.

- Request-type priority scheduling — Secondary bandwidth can be either "split" or "shared" for staging and streaming purposes. In the "split" policy, a certain amount of secondary bandwidth is permanently set aside for staging or streaming purpose. The bandwidth allocated is therefore left unused if it is not used for the intended purpose. On the other hand, in the "shared" policy, there is no such strict bandwidth partition — the unused secondary bandwidth can be allocated for either staging or streaming purpose. In this paper, we will consider the "shared" policy.

  Request scheduling is needed to be done under any one of the following three conditions:

  - There is a new incoming request; or
  - A user currently holding a stream leaves the system, hence freeing some streaming bandwidths; or
  - A video file has just staged onto the secondary level, freeing its own staging bandwidth.

At any point in time, the request queue may contain two type of requests: some hits followed by some misses, which in turn are followed by some hits, etc. In this paper, we consider the following prioritization schemes in scheduling these request types:

- First-Come-First-Served (non-prioritized scheme): This algorithm tries to assign secondary bandwidth according to the arrival order of the request types: if the request is a hit (i.e., a video is to be displayed), a streaming bandwidth for that request is assigned (followed immediately by taking the request out of the queue). If it is a staging miss, no bandwidth is allocated (as the file is currently being staged). If it is an absence miss, a file is first chosen according to the *staging-request scheduling algorithm.* Then, i) if the corresponding tape has already been loaded onto a tertiary drive, a bandwidth up to b_3 is allocated for staging at that time; otherwise ii) if an idle robot exists, an exchange then occurs while the next request in the queue is examined. The above algorithm continues until the entire secondary bandwidth is used up, or no more requests in the queue can be served (either because all requests are served, or there remains all hits and not enough streaming bandwidth for a hit is available at that time, or no misses can be served due to no idle robots, no idle drives, no staging bandwidth or staging room).

  The FCFS algorithm does not always serve the requests in strict arrival order as it puts throughput at a higher priority than order preservation when both conditions cannot be satisfied at the same time. Let's illustrate this with two cases:

  1. Due to the limited number of drives in the tertiary level, all the drives may be busy at one time. If this happens, staging requests cannot be served anymore;
  2. In preparing staging room for an in-coming file, a non-permanent-resident file has to be deleted. It may happen that there is not a single deletable file

in the secondary level. As a result, staging cannot proceed and hence no more staging request can be served.

In both cases above, our FCFS algorithm then "skips" all the misses in the queue and serves only the requests down in the queue, even though they arrived later. Similarly, in scheduling a hit request at a particular time, there may not be enough bandwidth to open a stream. In this case, the hit request cannot be served and hence skipped, and the next miss (absence miss or staging miss), if any, will then be served.

  - Hit-Requests-First (prioritized scheme): This algorithm puts the hits at a higher priority in assigning the secondary bandwidth than the misses. The algorithm schedules the hits first according to their arrival order, by assigning a streaming bandwidth to the hits and then taking them out of the queue (as their videos are displayed). After such assignment, if there is still unused bandwidth left, i.e., $B_2 - B_2^{strm}(t) > 0$ at that time t, the misses are then served (note that in this case, only misses remain): first by assigning bandwidth to the staging misses, followed by serving the absence misses scheduled according to the *staging-request scheduling.*

    Note that in the process of assigning bandwidth to the hits, the staging bandwidth of an on-going staging session may be affected: while in FCFS, the staging bandwidth of a staging session is never decreasing, in HRF, the assignment of a streaming bandwidth to a hit may take away some bandwidths from an on-going staging session. Therefore, a staging process can even be discontinued if there is insufficient secondary bandwidth B_2 to serve all the hits in the queue.

- Staging-request scheduling — We consider the following three staging-request algorithms:

  - First-Come-First-Served (FCFS): The tertiary level schedules the staging of a tertiary file according to the arrival order of the corresponding misses. This service policy, along with FCFS request queue scheduling, preserves the order of requests service according to their arrival sequence much better than the other combinations of request queue scheduling and staging-request scheduling.

  – Most-Requests-First (MRF): The tertiary file which has the most outstanding absence misses is staged first. In case of a tie, the absence miss with the earliest arrival time is served first. This algorithm therefore stages those files which have the most outstanding staging requests.

  – Most-Popular-First (MPF): The most popular tertiary file among the absence misses is staged first. In case of a tie in popularity, the absence miss with the earliest arrival time is served first. This algorithm is therefore relevant in those systems where video popularity is more or less stationary over time (or at least stationary over a period of time), and hence the popularity can be predicted or estimated pretty well (e.g., in a system which runs for a reasonably long time). Note that in reality, the popularity of a video file may be hard to assess or may not even be stationary over time.

- Replacement algorithm — File replacement actually involves two steps:

  1. Selection of a set of replaceable files: From all the non-permanent-resident files, this step selects a set of replaceable candidates according to a *replaceable-files-selection* algorithm.

  2. Replacement of a file: Within the set of the replaceable files, this step selects the one to be deleted according to a *deletion* algorithm. In case of a tie, it deletes the first file found.

  In this paper, we consider two replaceable-files-selection algorithms:

  – Closed-Deletable (CD): According to this algorithm, all the non-permanent-resident files which are not being displayed (i.e., the files which are closed) are deletable. As this algorithm would delete some "booked" files[1] which are not currently being displayed, such deletion would necessitate the staging of these files again. As staging is a system overhead (in terms of staging time, staging bandwidth and exchange latency), this will adversely affect system performance (i.e., lower streaming capacity and higher delay), especially under heavy traffic.

- Closed-not-Booked-Deletable (CnBD): We do not allow deletion of "booked" video files in this algorithm. Therefore, a video file is deletable only if it is not being displayed (i.e., closed) *and* is not booked. This algorithm therefore tries to decrease the number of staging, and hence staging overheads.

No matter which replaceable-files-selection algorithm we use, it may happen that all the files in the secondary level are not deletable at a time. In this case, a staging request cannot be served even though we have an idle tertiary drive and available secondary bandwidth. This "all-files-undeletable" phenomenon may persist for a while until some users leave the system, and hence "close" some files. The waiting time for the availability of a deletable file, W_d, can be rather long, especially under heavy traffic load, long user holding time and/or small secondary storage.

We note here that if we pick Hit-Requests-First (HRF) as our request-queue-scheduling policy, the replaceable set obtained using CD and CnBD are the same. This is because HRF schedules the hits at a higher priority than misses. As, by definition, "booked" files are also hit files, the files left behind after serving the hits are already not booked — if a file is closed, it is also not booked. Therefore, using CD or CnBD comes up with the same replaceable set.

Regarding the particular file to delete, we consider the following three deletion algorithms:

- Least-Recently-Used (LRU): A file is being "used" if it is being displayed (i.e., open). Hence, the algorithm deletes the file which has not been used or accessed

[1]Recall that a booked file is a file requested by some users in the queue.

(i.e., has been closed) for the longest time. This is based on the assumption that least-recently-used file is also most likely not to be accessed in the near future.

  - Least-Popular-First (LPF): Within the set of replaceable files, this algorithm deletes the least popular ones. This algorithm is therefore useful if we know the video popularity.
  - Random (RND): We delete randomly one of the files in the replaceable set.

4.4 Baseline operational procedures

We see that the design of a hierarchical storage system actually involves a lot of operational parameters. In order to study the performance of such a system, we have looked into a baseline operation. Using the baseline, we can then obtain its performance characteristics, against which other variations of policies can be compared. The baseline operations we consider are:

- Admission control: Blocked-Call-Held (BCH);
- Request-queue scheduling:
  - "Shared" bandwidth allocation;
  - First-Come-First-Served (FCFS);
- Staging-request scheduling: First-Come-First-Served (FCFS);
- Replacement algorithm:
  - Replaceable-files-selection: Closed-not-Booked-Deletable (CnBD);
  - Deletion algorithm: Least-Recently-Used (LRU).
  - In case of a tie, the file found first is deleted.

5 Performance Characteristics

5.1 Performance measure

The performance measure we mainly consider here is the "start-up delay" D_{st}. In this study, we mainly consider average start-up delay for all requests, for all misses and for all hits in the steady state. We will also address their delay distribution and variance.

This section will be divided into three parts:

- The study of the influence of architectural parameters, i.e., C_2, B_2, and B_3, on the performance of the hierarchical storage system. We divide the study into two parts:
  - Sensitivity study: the study of the influence of one or more design choices on the performance measure of interest. The study allows one to understand the marginal benefit in altering the value of a design choice; and
  - Trade-off study: the study of the inter-dependence of two or more design choices to see how one parameter can be "traded" off with the other(s) in order to achieve the same performance.
- The study of various operational procedures on the performance of the system. In this study, we vary the operational policy from our baseline operation and examine its effect.
- The influence of applications characteristics on the performance of the hierarchical storage system. In this section, we consider how different application environments or characteristics, e.g., file size, user's holding time distribution, etc., would affect system performance.

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

- Operational procedures: According to the baseline operation given in Section 4.4

- Demand 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 making their requests, 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.
- The tertiary exchange latency is negligible (compared with staging time) and can be ignored. We will address non-negligible tertiary exchange time later. We further consider that at any one time only one file can be staged, and hence the entire bandwidth of the tertiary level, B_3, can be used in parallel to deliver the file. In other words, we consider $N_{dr}^{(3)} = 1$ (maximum parallelism), and hence with bandwidth $b_3 = B_3$. We will address the issue $N_{dr}^{(3)} > 1$ later.

We have implemented an event-driven simulation to study the performance of the hierarchical storage system, and collected statistics of the users' start-up delay after the transient response has passed. Using "batch mean procedures,"[27, 28] most of our 95% confidence intervals fall within 5% of the simulation values, with the majority of them falling within 3%. For plot clarity, we will not show the confidence intervals here.

5.2 Architectural parameters

5.2.1 Sensitivity study

We consider N_v video files in the system and let p_i be the popularity for video i, where $1 \leq i \leq N_v$. Therefore, for uniform video popularity, we have $p_i = 1/N_v$. In this section, we first address the hit, miss and overall delay as a function to the arrival rate. Then we examine the effect on delay of secondary level characteristics (C_2 and B_2) and tertiary level characteristics (B_3, $N_{dr}^{(3)}$, and T_{ex}).

Before we examine the hit, miss, and overall delay, we have to note here that a request type (i.e., hit or miss) may change while it is waiting to be served.[2] In order to investigate whether a scheduling policy shows favoritism to a particular request type, we need to define at what instant the request type is measured. In studying the delay performance, we consider the request type to be specified at arrival — the hits are those requests having their videos resident upon their arrivals, while the misses are those requests whose videos are not resident upon their arrival. Note that with CnBD (the baseline replaceable-files-selection algorithm), a hit will never change its type while being queued.

Figure 3 shows the average start-up delay for misses (staging and absence misses), hits and all requests as a function of the arrival rate λ, for $B_2 = 10$ MB/s and 20 MB/s. We have used $C_2 = 100$ GB, $B_3 = 10$ MB/s, and $N_v = 500$. We note from the figure that there is a sharp "knee" at around $\lambda \simeq 16$ and 35 req./hr for $B_2 = 10$ MB/s and 20 MB/s, respectively. We notice that at the kink, long staging requests is developed, i.e., the tertiary level becomes "congested." In designing a server with low delay, we should therefore limit the arrival rate below the knee. In this sense, the "knee" is the capacity of the server. We note here that the true capacity of the server, indicated by the infinite delay, is about 70 req./hr for $B_2 = 20$ MB/s (given by $B_2/(T_h b_0)$), occurring when all the secondary bandwidth is used for streaming.

[2]Note that a miss has to change to a "hit" before its video can be displayed, and a hit while being queued will change to a miss if its requested video gets deleted. Such changes may occur many times before a requested video is finally displayed.

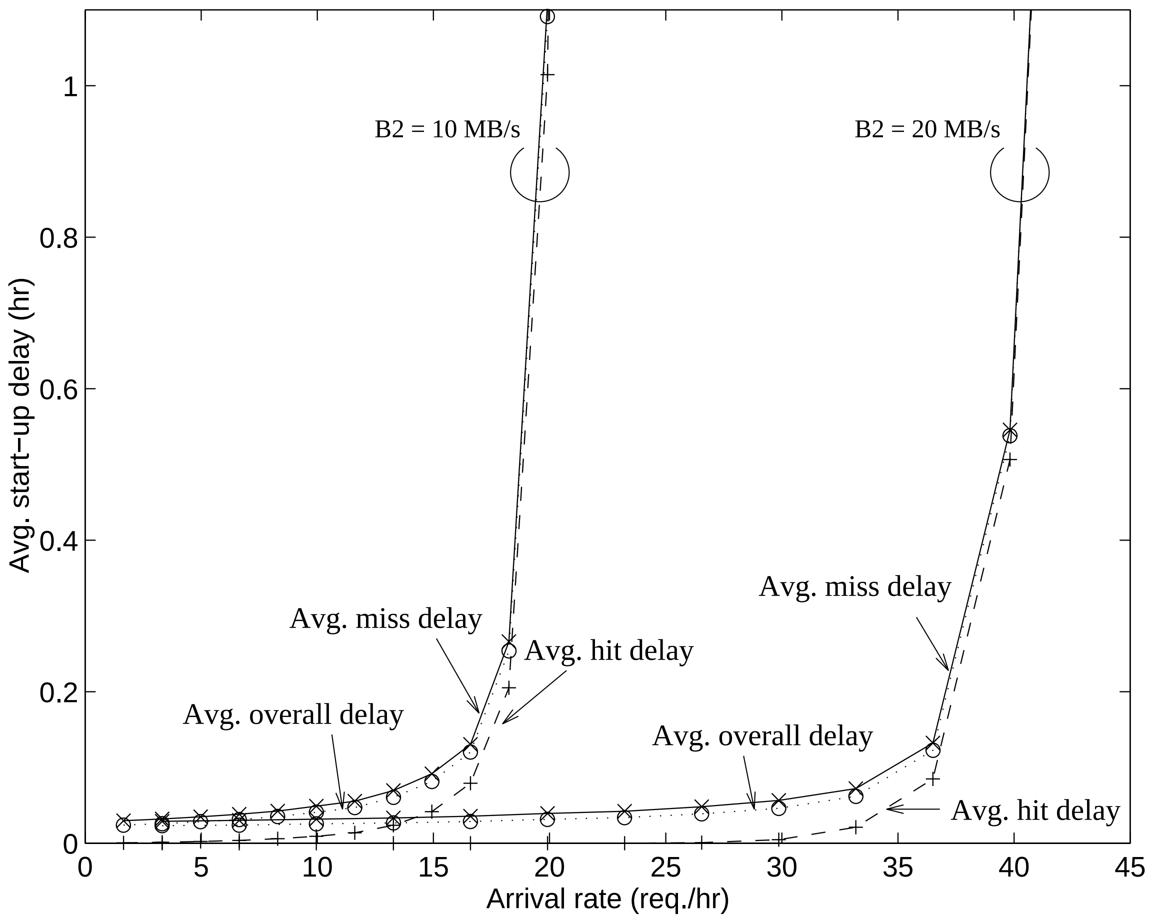

Figure 3: $\overline{D}_{st}$, $\overline{D}_{hit}$ and $\overline{D}_{miss}$ as λ is increased, with $B_2 = 10$ MB/s and 20 MB/s ($C_2 = 100$ GB, $B_3 = 10$ MB/s, $N_v = 500$, and uniform video popularity)

We therefore have,

$$\overline{D}_{st} = \alpha_2 \overline{D}_{hit} + (1-\alpha_2)\overline{D}_{miss}, \tag{7}$$

where $\overline{D}_{hit}$ and $\overline{D}_{miss}$ are average hit and miss delay respectively, and α_2 is the hit probability. In our case, since our secondary level is able to store 20% of the video files ($C_2 = 100$ GB), the fraction of hits on arrival (probability of hit on arrival) is $\alpha_2 \simeq 20\%$.[3] This is confirmed by simulation. We therefore have $\overline{D}_{st} \approx 0.2\overline{D}_{hit} + 0.8\overline{D}_{miss}$.

From the figure, we see that the hits enjoy minimal delay up to the knee (i.e., $\lambda \simeq 35$ req./hr for $B_2 = 20$ MB/s and $\lambda \simeq 16$ req./hr for $B_2 = 10$ MB/s), beyond which both hits and misses are delayed similarly (due to our FCFS policy). For a system operating below the knee region, we can have the following approximation:

$$\overline{D}_{st} \simeq (1-\alpha_2)\overline{D}_{miss}. \tag{8}$$

We are now ready to examine more closely the delay distribution of various request types. Figure 4 shows the delay distribution and the percentage of misses for the delay= 20.[4] We show here a system under low traffic load ($\lambda = 12.8$ req./hr). There are two "peaks" in the figure, one at the delay range 0 – 0.45 mins. and the other at 1.35 min. – 1.75 min. The bin with lower delay (0 – 0.45 mins.) actually consists of all hits with zero start-up delay ($D_{st} = 0$). From the examination of the delay profile, a large portion of the misses are delayed by 1.67 min., the staging time for a video file with no queuing delay (given by $C_f/B_3 = 1$ GB/(10 MB/s) $= 1.67$ min.) The tail of the distribution is due to queueing delay for the misses. Virtually all requested videos are displayed within 4 minutes.

[3]In fact, the fraction of hits is expected to be slightly less than 20%, since we define all arrivals finding their files being staged as misses. However, if the staging time is much less than the holding time (which is in fact our case of interest), the above is indeed a good approximation.

[4]The delay distribution can be obtained by putting the start-up delay for each user in the system into respective "delay bins." Note that each bin is composed of both misses and hits. Besides delay distribution, we can therefore also obtain the percentage of misses in each bin.

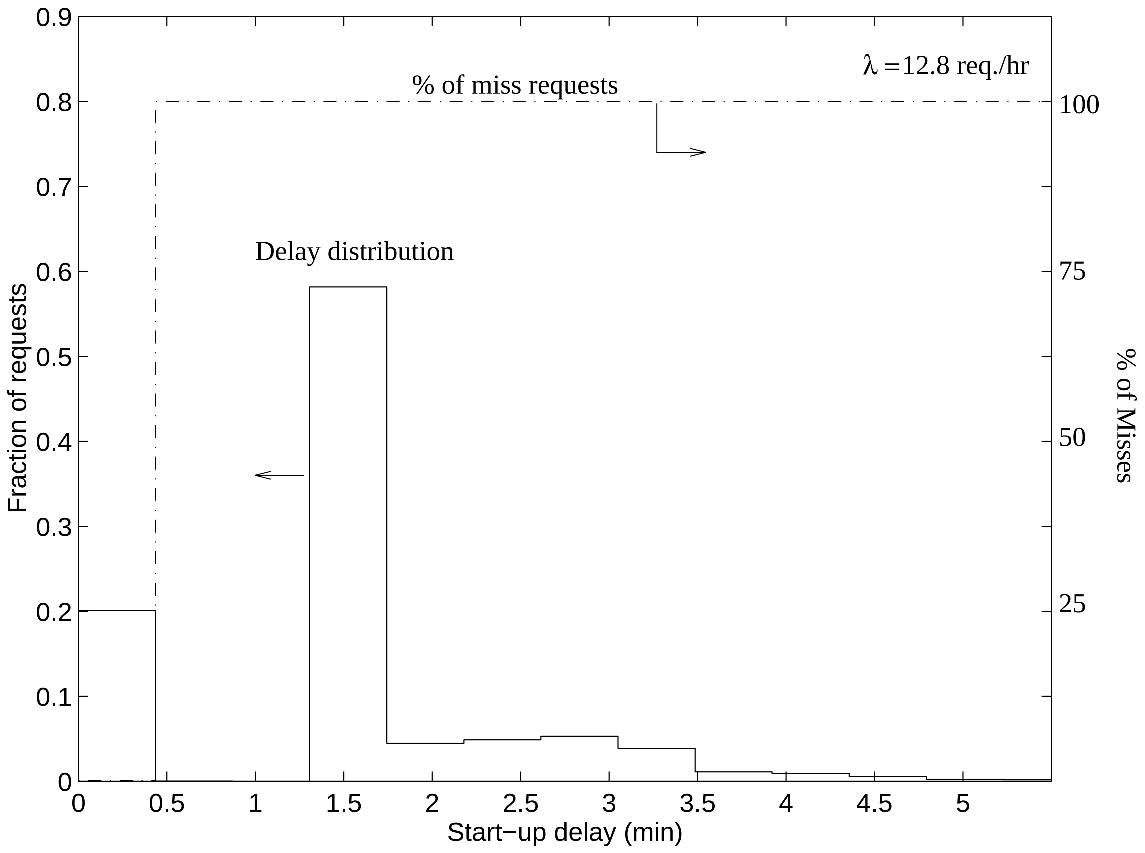

Figure 4: Delay distribution and its miss (upon arrival) percentage in the respective delay range, with low arrival rate ($C_2 = 100$ GB, $B_2 = 20$ MB/s, $B_3 = 10$ MB/s, $N_v = 500$, and uniform video popularity)

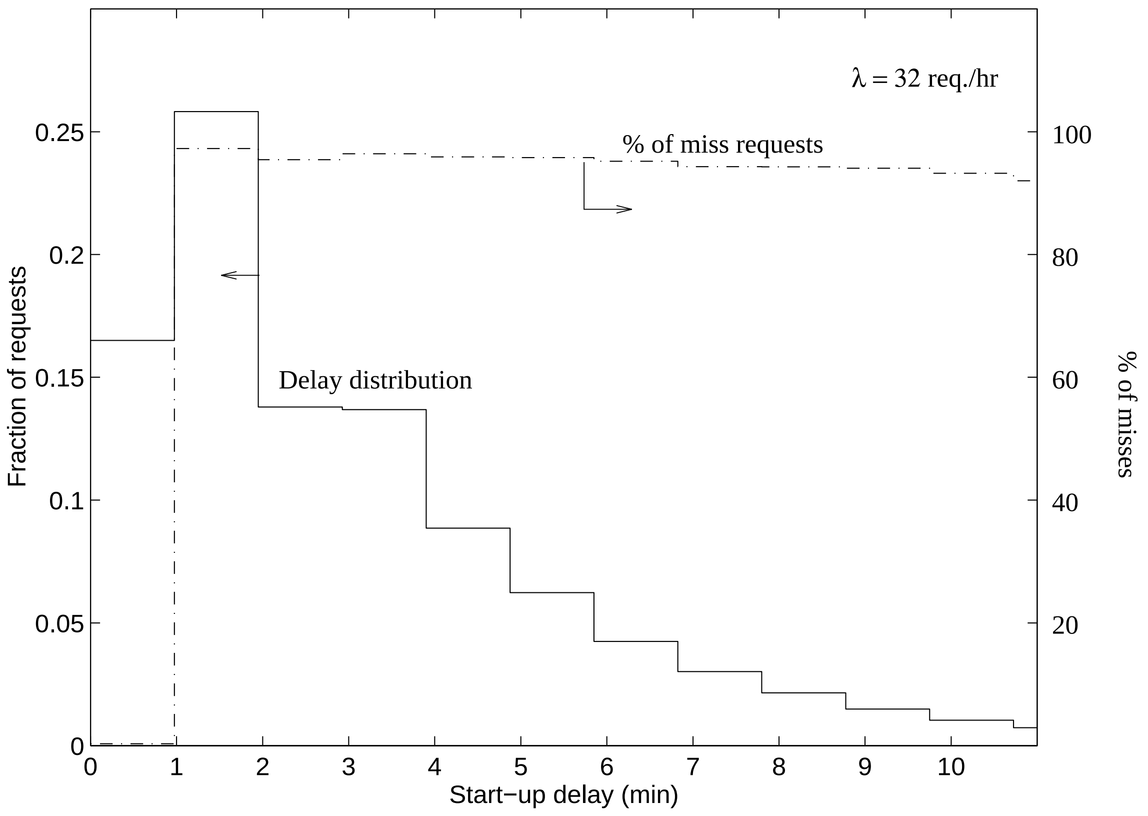

Figure 5: Delay distribution and its miss (on arrival) percentage in the respective delay range, with moderate arrival rate ($C_2 = 100$ GB, $B_2 = 20$ MB/s, $B_3 = 10$ MB/s, $N_v = 500$, and uniform video popularity)

Figure 5 shows the delay distribution under moderate traffic (at the "knee" with $\lambda = 32$ req./hr). Compared with Fig. 4, we note that the delay distribution has a higher variance (i.e., with a reduced, broader peak and longer tail). We see that some hits (20% of the hits) are delayed. The tail of the distribution is almost entirely composed of misses. We see a relatively high delay peak at $1-2$ min., the minimum staging time for a miss request. However, only about 24%, as compared to 60% in low traffic, of the requests enjoy such low delay. We see that at moderate traffic, virtually all the requested videos are displayed within 10 minutes, with hits generally enjoying low delay and misses suffering longer and more undeterministic delay.

Figure 6 shows the distributions as we increase the traffic further (heavy traffic condi-

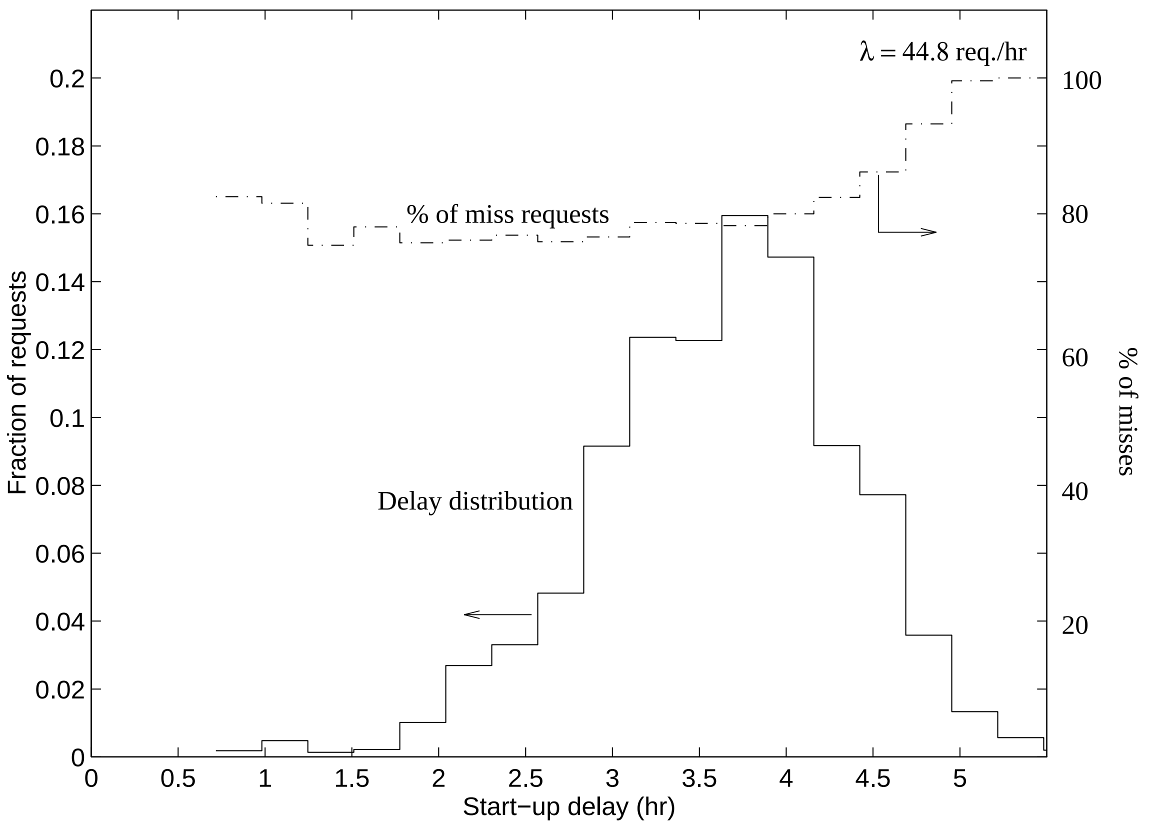

Figure 6: Delay distribution and its miss (on arrival) composition under heavy traffic (C_2 = 100 GB, B_2 = 20 MB/s, B_3 = 10 MB/s, N_v = 500, and uniform video popularity.)

tion: $\lambda = 44.8$ req./hr). All requests are now delayed – none enjoy delay less than 0.7 hour. The hits no longer have low delay as they do under low and moderate traffic — both hits and misses are delayed. Compared with moderate traffic, the delay distribution is further spread out (with a higher delay variance). We observe from our simulation traces that such long start-up delay is due to congestion in tertiary level, where the staging of video files is not fast enough compared with arrival rate.

Having investigated the delay with respect to arrival rate, we now vary the secondary characteristics (C_2 and B_2) to see how they influence the server performance. From this study, we are able to identify bottlenecks or extra resources in the system.

We show in Fig. 7 how $\overline{D}_{st}$ is affected when C_2 increases for three arrival rates: $\lambda = 20$ requests/hr (low arrival rate), 35 requests/hr (moderate arrival rate), and 45 requests/hr (heavy arrival rate). We see that, there is a threshold C_2 for each arrival rate in order to achieve low latency: all curves show an initial sharp decrease, followed by a slower decrease to zero. For an arrival rate as low as 20 req./hr, $\overline{D}_{st}$ achieves low value for $C_2 \approx 40$ GB. When $\lambda = 35$ requests/hr, a higher storage capacity is required ($C_2 > 70$ GB) to achieve low delay. With this arrival rate, we observe from our simulation trace that if $C_2 < \sim 60$ GB, there is frequent occurrence of the "all-files-undeletable" phenomenon. As deletable files are not always available, staging time is increased due to waiting for a file to be closed. For a slightly higher traffic rate such as $\lambda = 45$ requests/hr, C_2 has to be increased to over 250 GB to achieve low average delay, more than twice of that required for $\lambda = 35$ requests/hr!

Figure 8 plots the average start-up delay, $\overline{D}_{st}$, as B_2 increases, for λ's = 20 req./hr, 35 req./hr and 40 req./hr. Here, we have $C_2 = 100$ GB, $B_3 = 10$ MB/s, $N_v = 500$ and uniform video popularity. All curves eventually settle to their limiting values. The limiting value can be obtained by observing that as B_2 increases to infinity, all the hits will be served immediately and hence there are only misses (a fraction $1 - \alpha_2$ of the arrivals) needed to be staged from the tertiary level. Since the files are of constant size and, as $B_2 \to \infty$ the staging bandwidth is becomes constant and equals to B_3, the tertiary level can be modeled as an $M/D/1$ system. Therefore,[5]

$$\begin{aligned} \lim_{B_2 \to \infty} \overline{D}_{st} &= (1 - \alpha_2)\overline{D}_{miss} && (9) \\ &\approx (1 - \alpha_2)\left[\frac{1}{\mu_3} + \frac{(1 - \alpha_3)\lambda/\mu_3}{2(\mu_3 - (1 - \alpha_3)\lambda)}\right], && (10) \end{aligned}$$

where μ_3 is the tertiary service rate and is given by $B_3/((1-\alpha_2)\lambda C_f)$. For $\lambda_2 = 20$ req./hr, 35 req./hr, and 40 req./hr, we therefore get (with $\alpha_2 = 0.8$) $\overline{D}_{st} = 0.02$ hr., 0.05 hr., and 0.1 hr., respectively. The delay settles to its limiting value when [29]:

$$B_2^* \simeq B_3 + \lambda T_h b_0 + z\sqrt{\lambda T_h} b_0, \qquad (11)$$

[5]Equation (10) is only approximately true because i) the number of video files is not infinite, ii) we have ignored the deletable-file-wait, W_d (i.e., a deletable file can always be found), and iii) we have ignored hit delay. The complete treatment taken into account of the aforementioned conditions can be found in [29].

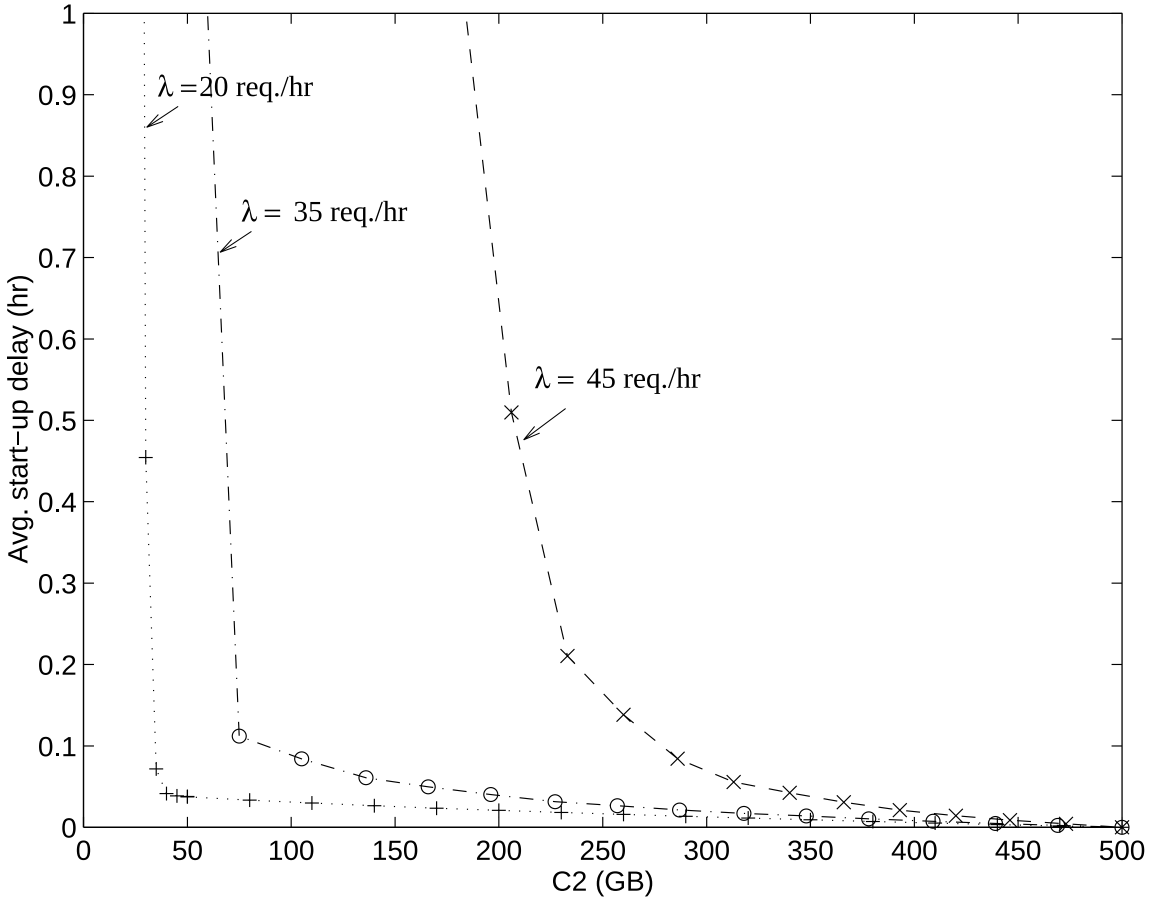

Figure 7: $\overline{D}_{st}$ versus C_2, for three different λ's. (C_2 = 100 GB, B_2 = 20 MB/s, B_3 = 10 MB/s, N_v = 500, and uniform video popularity.)

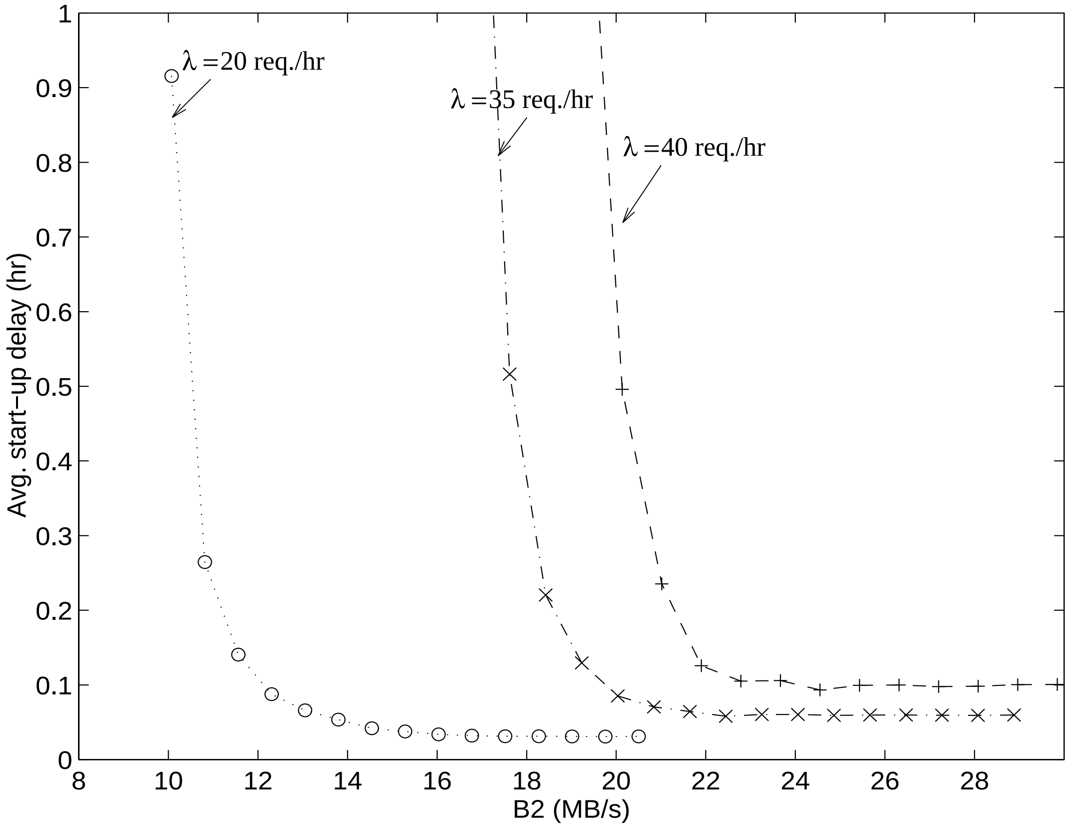

Figure 8: Average start-up delay versus B_2, for three values of λ's. ($B_2 = 20$ MB/s, $B_3 = 10$ MB/s, $N_v = 500$, and uniform video popularity.)

where the third summand is due to stochasticity in bandwidth demand. Under circumstances of practical interest, z is given by $(1/\sqrt{2\pi})\int_z^\infty \exp(-x^2/2)dx = \beta/\rho_3$, where ρ_3 is the utilization in the tertiary level and β is small (≤ 0.1). For $B_3 = 10$ MB/s and $\lambda = 20$ req./hr, 35 req./hr and 40 req./hr, $B_2^* \simeq 15$ MB/s, 20 MB/s, and 22 MB/s respectively. In fact, since usually $|z| \leq 1.5$ [29],

$$B_2^* \approx B_3 + \lambda T_h b_0. \tag{12}$$

To understand better how B_2 is used between staging and streaming, we need to plot the respective utilization versus the arrival rate λ. We define total secondary bandwidth utilization, ρ_2, as the time average of the fraction of B_2 used to serve both hit and miss requests. In other words, ρ_2 indicates how well B_2 is used for streaming and staging, i.e., if $\rho_2 = 100\%$, B_2 is fully utilized and never idle. We further define the utilization for streaming, ρ_2^{strm}, and the utilization for staging, ρ_2^{stg}, as the time-averaged fraction of B_2 used for streaming and staging, respectively. We see immediately that ρ_2^{strm} indicates the throughput of the server while ρ_2^{stg} indicates the staging overhead of the system. Denote ρ_3 the utilization of B_3, defined as the time average of the fraction of B_3 used for staging purpose, we obviously must have:

$$\rho_2 = \rho_2^{strm} + \rho_2^{stg}, \tag{13}$$

$$\rho_3 = \frac{\rho_2^{stg} B_2}{B_3}. \tag{14}$$

We plot in Fig. 9 ρ_2, ρ_2^{stg} and ρ_2^{strm} versus λ, for $B_2 = 20$ MB/s, $B_3 = 10$ MB/s, $N_v = 500$ (uniform popularity), and $C_2 = 50$ GB, 100 GB, and 200 GB. We note from the figure that, even with such a wide range of storage capacity (50 GB – 200 GB), the bandwidth utilizations ρ_2, ρ_2^{stg}, and ρ_2^{strm} are remarkably similar. For $C_2 = 100$ GB and when $\lambda \simeq 40$ req./hr, B_2 gets basically fully utilized, with 40% used for staging (i.e., 8 MB/s) and the remaining used for streaming.

In fact, ρ_2^{strm} and ρ_2^{stg} (and hence ρ_2 and ρ_3) can be derived as follows. Since in stability ("what comes in must come out"), the average streaming bandwidth is given by $\lambda T_h b_0$,

$$\rho_2^{strm} = \frac{\lambda T_h b_0}{B_2}. \tag{15}$$

This confirms with simulation that ρ_2^{strm} increases linearly with traffic λ.

We also see that ρ_2^{stg} initially increases rather linearly until a maximum is reached. Under light traffic, since each staging request require the staging of a file of size C_f,

$$\rho_2^{stg} = \frac{(1-\alpha_2)\lambda C_f}{B_2}, \tag{16}$$

and hence the initial linear slope can be easily shown to be $(1-\alpha_2)C_f/B_2$.

Using Eqs.~(15) and (16), the arrival rate at which ρ_2 is 100% utilized, or at which ρ_2^{stg}

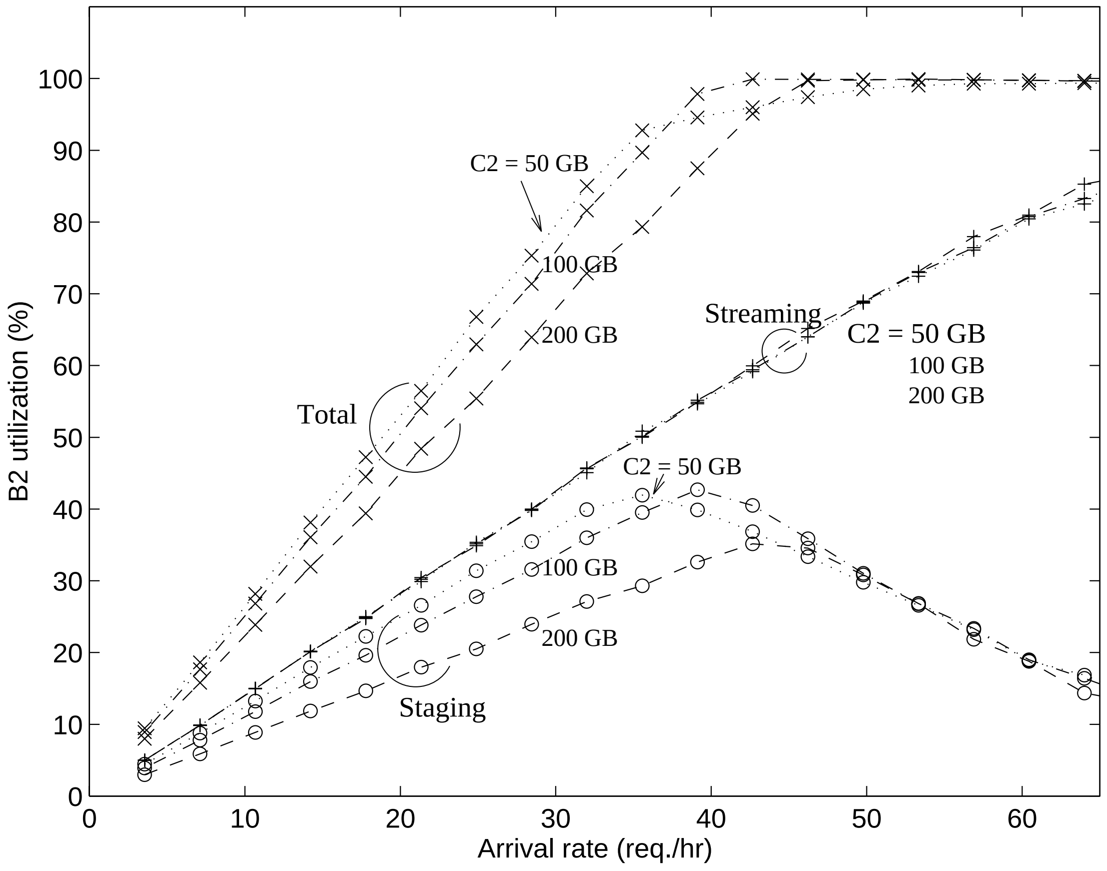

Figure 9: Total secondary bandwidth utilization, ρ_2, the utilizations for streaming (ρ_2^{strm}) and staging (ρ_2^{stg}) versus arrival rate λ, for three values of C_2 ($B_2 = 20$ MB/s, $B_3 = 10$ MB/s, $N_v = 500$, and uniform video popularity.)

achieves its maximum, is therefore given by:

$$\lambda^* \left(\frac{T_h b_0}{B_2}\right) + \lambda^* \left(\frac{(1-\alpha_2)C_f}{B_2}\right) = 1,^6 \tag{17}$$

from which we obtain,

$$\lambda^* = \frac{B_2}{T_h b_0 + (1-\alpha_2)C_f}. \tag{18}$$

Using the given parameters, we get $\rho_2 = 100\%$ when $\lambda^* = 37.6$ req./hr, 39.7 req./hr and 44.7 req./hr, for $C_2 = 50$ GB, 100 GB and 200 GB, respectively, agreeing well with our simulation results.

As λ is further increased beyond λ^*, the queue for staging requests quickly develops. If the system is stable (which is our case), ρ_2^{strm} will keep increasing according to Eq. (15) and ρ_2^{stg} will decrease so that $\rho_2^{strm} + \rho_2^{stg} \triangleq \rho_2 = 1$. This is the reason why ρ_2 increases until it reaches 100%, after which ρ_2^{stg} decreases so as to maintain ρ_2 at that level.

The point at which $\rho_2 = 1$ does not necessarily imply instability, i.e., infinite delay. In fact, simulation and analysis confirm that λ can be increased much beyond the point before the server becomes unstable. In fact, instability occurs where ρ_2^{strm} reaches 100%. For the numerical case under consideration, $\lambda = B_2/(T_h b_0) \simeq 70$ requests/hr. This is because staging bandwidth can be decreased (and hence staging time and delay can be correspondingly increased) in order to accommodate more streaming traffic. Note that since the number of tertiary files is finite, the delay of the staging requests in our baseline policy will never be infinite, albeit it can be very long. Hence, the only instability is due to queueing for available streams.

Having studied the influence of secondary level characteristics, we now look into the influence of tertiary level characteristics, namely, B_3, T_{ex} and $N_{dr}^{(3)}$. Figure 10 shows the influence of tertiary bandwidth on $\overline{D}_{st}$, for $\lambda = 20$ req./hr, 35 req./hr, and 40 req./hr. We have considered $B_2 = 20$ MB/s, $C_2 = 100$ GB, and $N_v = 500$ (uniform popularity). Obviously, we do not have to increase B_3 beyond B_2. The average delay decreases sharply as B_3 is increased and settles to a limiting value. For each arrival rate, there is also a

[6]We have considered the case where at its maximun , $\rho_3 \leq 1$. See [29] for a more complete treatment.

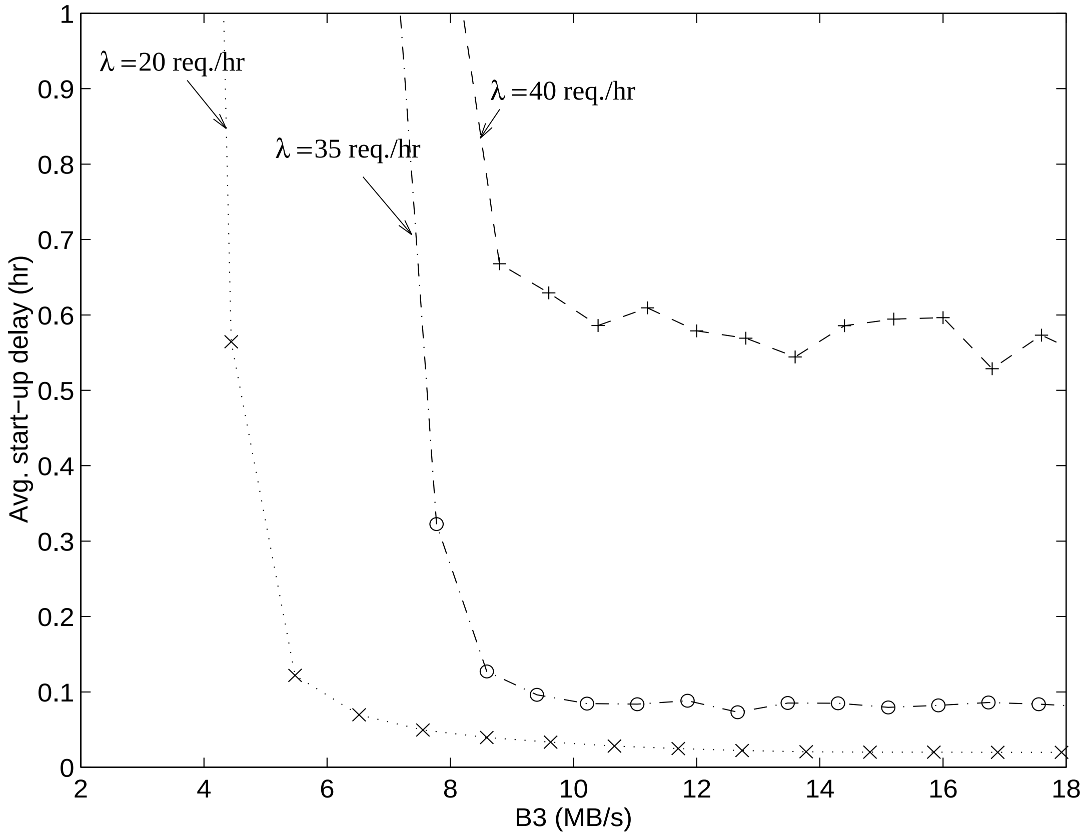

Figure 10: $\overline{D}_{st}$ versus B_3, for three values of λ's. ($B_2 = 20$ MB/s, $C_2 = 100$ GB, $N_v = 500$, and uniform video popularity.)

"knee" at which delay settles to its limiting value. For low to moderate arrival rate (i.e., $\lambda \leq 35$ req./hr), this knee is more or less at,[7]

$$B_3 \approx (1 - \alpha_2)\lambda C_f/\beta, \tag{19}$$

where $\beta \approx 0.8$. We once again see that there is no incremental value in increasing B_3 beyond $\approx B_2 - \lambda T_h b_0$.

Figure 11 shows the performance of the hierarchical storage system with non-negligible tertiary exchange time, T_{ex}. Here, we consider constant exchange times: negligible exchange time ($T_{ex} = 0$, the baseline case), moderate exchange time ($T_{ex} = 15$ sec.) and long

[7]This expression is approximately true as we have assumed i) no deletable-file-wait, and ii) infinite number of tertiary files. More complete treatment can be found in [29].

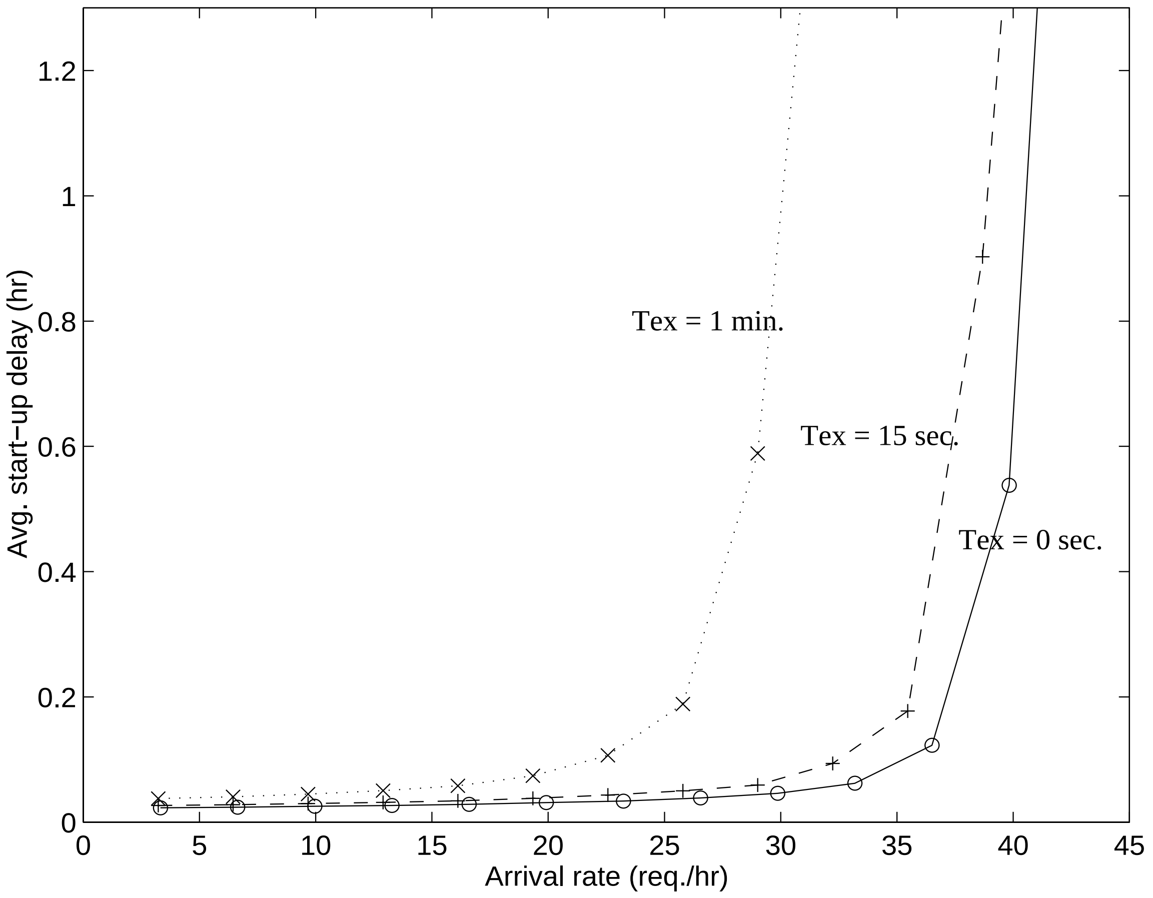

Figure 11: Average delay performance with different exchange latency, T_{ex} ($C_2 = 100$ GB, $B_2 = 20$ MB/s, $B_3 = 10$ MB/s, $N_{dr}^{(3)} = 1$, and uniform popularity)

exchange time ($T_{ex} = 1$ min.). We assume our usual baseline operations, with $C_2 = 100$ GB, $B_2 = 20$ MB/s, $B_3 = 10$ MB/s, $N_v = 500$ (uniform popularity). We see from the figure that there is a marked decrease in performance (in terms of delay and streaming capacity) when the exchange time increases from 15 sec. to 1 min. For example, to satisfy an average delay requirement of 6 min., the system with $T_{ex} = 1$ min. can only handle $\lambda \simeq 22$ req./hr, while the system with $T_{ex} = 15$ sec. and negligible exchange time can have much higher streaming capacity of $\simeq 32$ req./hr and $\simeq 35$ req./hr, respectively. This is over 40% increase in capacity!

Figure 12 shows the delay performance as the number of drives in the tertiary level, $N_{dr}^{(3)}$, increases from 1 (the baseline case) to 12 (multiple independent drives). Here we consider our baseline operation with negligible T_{ex}, $C_2 = 100$ GB, $B_2 = 20$ MB/s, $B_3 = 10$ MB/s, and $N_v = 500$ (uniform video popularity). We keep the maximum tertiary bandwidth, B_3, the same in all cases (i.e., $B_3 = 10$ MB/s). In other words, as we vary $N_{dr}^{(3)}$ from 1 to 12, the bandwidth of each drive varies from $b_3 = 10$ MB/s to 0.833 MB/s. As $N_{dr}^{(3)}$ increases, bandwidth parallelism decreases while service concurrency increases (maximum concurrency increases from 1 to 12).

At low to moderate traffic (i.e., $\lambda \leq 30$ req./hr), the start-up delay with $N_{dr}^{(3)} = 12$ is much higher than that of the system with $N_{dr}^{(3)} = 1$, because of its lower drive bandwidth. In fact, the delay ratio is approximately equal to the ratio between the number of drives (i.e., $\overline{D}_{st} \propto N_{dr}^{(3)}$). Therefore, in designing a hierarchical storage system with file-by-file staging, $N_{dr}^{(3)} = 1$ (i.e., maximum parallelism) achieves the lowest delay. At high traffic (i.e., $\lambda \geq\sim 40$ req./hr), all systems perform similarly (and poorly). This is expected as all the tertiary drives are used, rendering the service rate to be the same in all cases.

5.2.2 Trade-off study

In this section, we consider the trade-off between various architectural parameters to achieve a certain server performance.

Figure 13 shows the trade-off between C_2 and B_2, given a certain arrival rate $\lambda = 30$

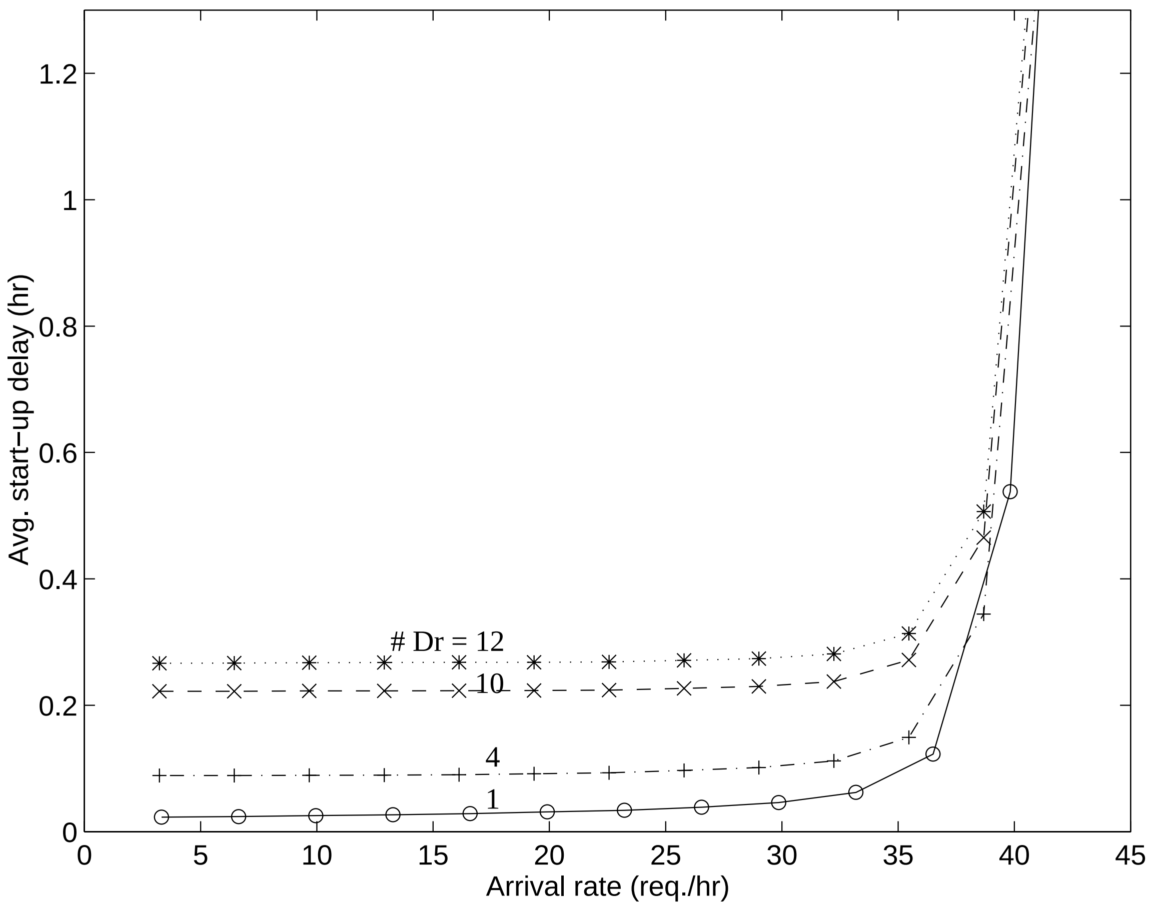

Figure 12: Influence of $N_{dr}^{(3)}$ on the performance of a hierarchical storage system ($C_2 = 100$ GB, $B_2 = 20$ MB/s, $B_3 = 10$ MB/s, and $N_v = 500$ with uniform video popularity, and negligible T_{ex})

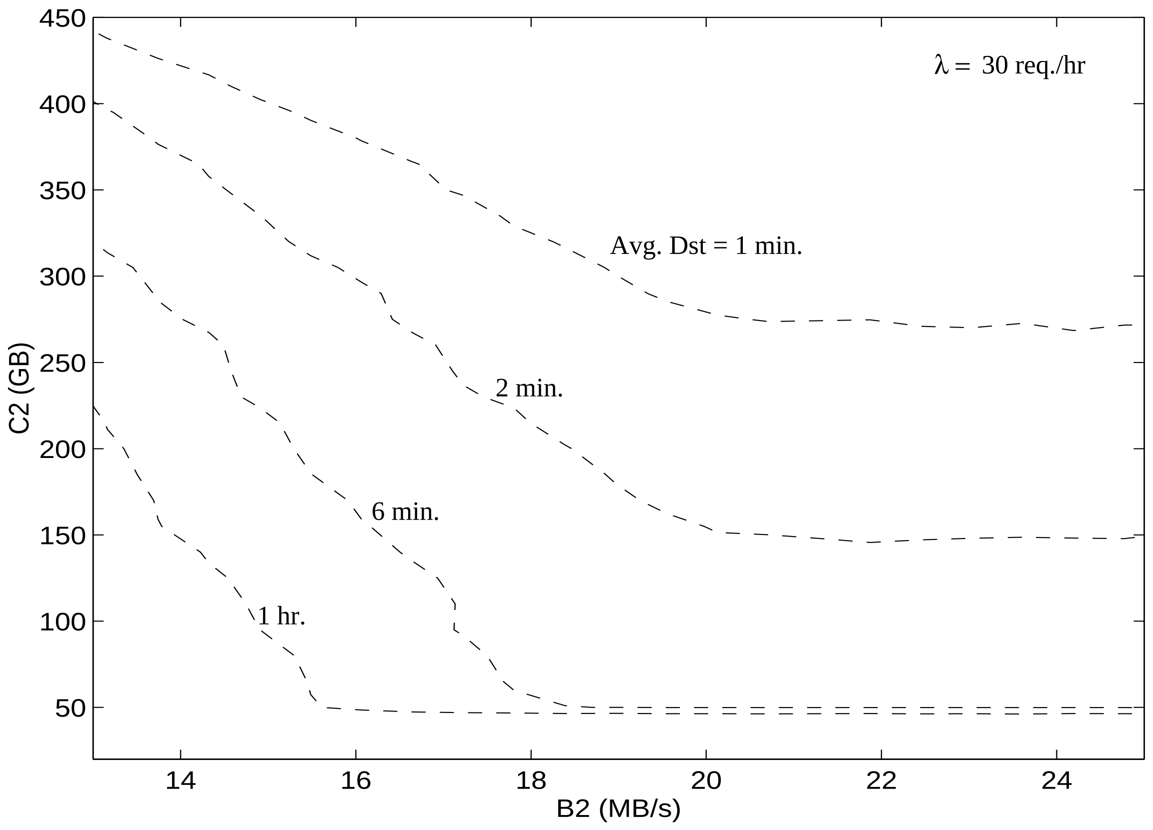

Figure 13: Trade-off between C_2 and B_2, for various of $\overline{D}_{st}$ ($\lambda = 30$ req./hr, $B_3 = 10$ MB/s, negligible T_{ex}, $N_v = 500$, and uniform video popularity.)

req./hr. We show the trade-off for four average start-up delays $\overline{D}_{st} = 1$ minute, 2 minutes, 6 minutes, and 1 hour. The system parameters we consider are $B_3 = 10$ MB/s, $N_v = 500$, negligible T_{ex}, 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 beyond which no "trade-off" is possible. Therefore, to achieve a certain average delay, a higher secondary bandwidth can generally be traded off with a smaller storage capacity up to a certain point.

We remark from Fig. 13 that a system would most likely be operated at the knee of the curve, (B_2^*, C_2^*). This is because any B_2 higher than B_2^* cannot effectively "trade with" a lower storage capacity, and hence is of no marginal benefit (i.e., B_2 is in excess if it is larger than B_2^*). On the other hand, due to the rather steep linear region, the marginal value of the decrease in bandwidth can hardly offset the cost in the increase in storage. This point can be illustrated as follows: Suppose we require 2 minutes delay (and hence, the knee is at $(B_2^*, C_2^*) \approx (20$ MB/s, 155 GB)). By decreasing B_2 slightly from 20 MB/s to 18 MB/s (10% decrease), we need $C_2 = 210$ GB, a significant 40% increase in storage capacity! If we consider $\overline{D}_{st} = 6$ minutes, when B_2 is decreased slightly from 18.5 MB/s to 17.5 MB/s (5.4% decrease), the storage requirement has to be increased to 80 GB (a 60% increase)!

For the 1-hr delay contour, we observe from our simulation traces that when $C_2 < \sim 50$ GB (50 videos), there is a long queue of staging requests. This long "staging queue" is caused by frequent "all-files-undeletable" phenomenon and hence long waiting time for the availability of a deletable file, W_d (in the order of minutes). As staging cannot proceed without finding a replaceable file, a long W_d adversely slows down the staging process of a file. This inefficiency leads to long staging queue. Therefore, B_2 is no longer the scarce resource (but C_2) and so increasing it does not help in the delay performance. For W_d to be negligible, we need $C_2 \geq (\lambda T_h + n_2)C_f$, where $n_2 \approx 6 - 10$ [29].

Figure 14 shows the influence of video popularity on the trade-off between C_2 and B_2, with $\lambda = 30$ req./hr, $B_3 = 10$ MB/s, and $N_v = 500$. We have used the geometric video popularity model given in Section 2 and assume 80/20 popularity model, i.e., 80% of the requests ask for 20% of the videos. This gives $\sigma = 0.98405$ in Equation (3). We see immediately that non-uniform video popularity is able to achieve the same average delay with markedly less storage and/or bandwidth.

Figure 15 shows the trade-off between secondary storage capacity C_2 and tertiary bandwidth B_3, with $\overline{D}_{st} = 1$ minute, 2 minutes, 6 minutes, and 1 hour. Here, we use $\lambda = 30$ req./hr, $B_2 = 20$ MB/s, $N_v = 500$, and uniform video popularity. We again observe a linear region followed by a horizontal asymptote — the linear region indicates that C_2 can flexibly trade off with B_3, while the horizontal asymptote indicates the limit of such trade-off (excess B_3). Therefore, for a given delay of, say, 2 min., it may not be worthwhile to increase tertiary bandwidth beyond 11 MB/s. We observe, as in Figure 13, low performance with $C_2 < \sim 50$ GB. This is again due to the frequent "all-files-undeletable" phenomenon and

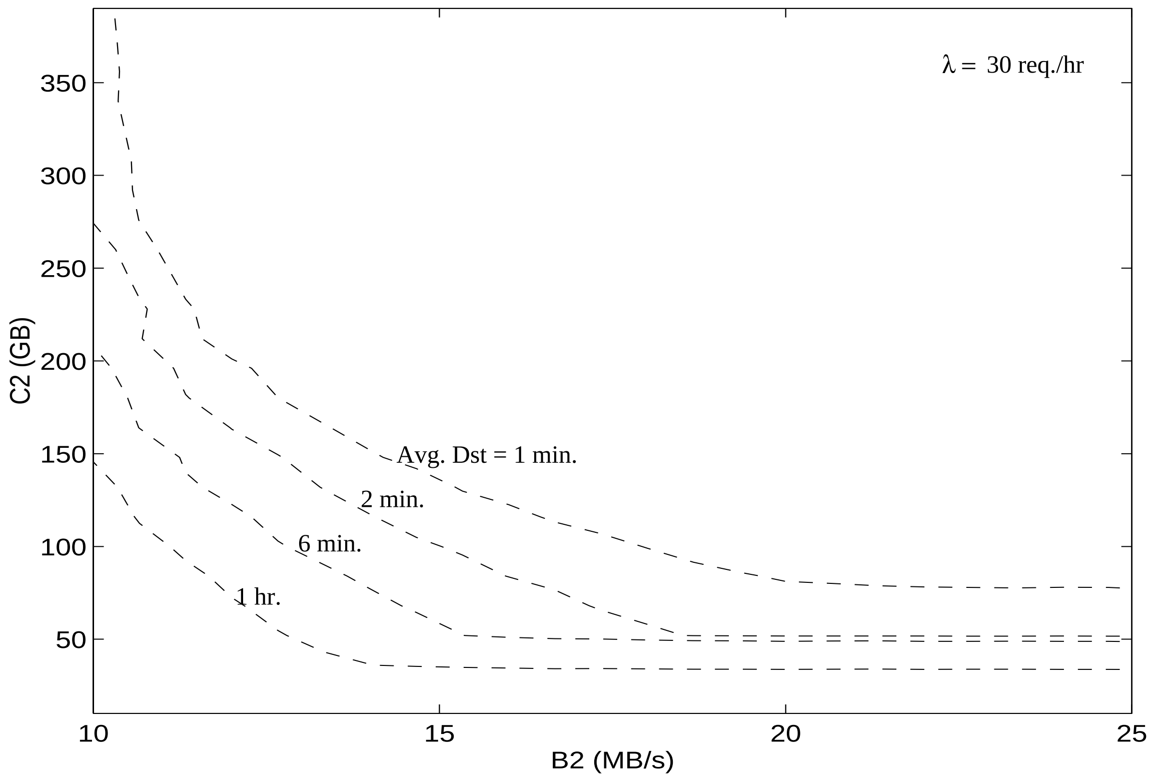

Figure 14: Trade-off between C_2 and B_2, with 80/20 video popularity ($\lambda = 30$ req./hr, $B_3 = 10$ MB/s, $N_v = 500$)

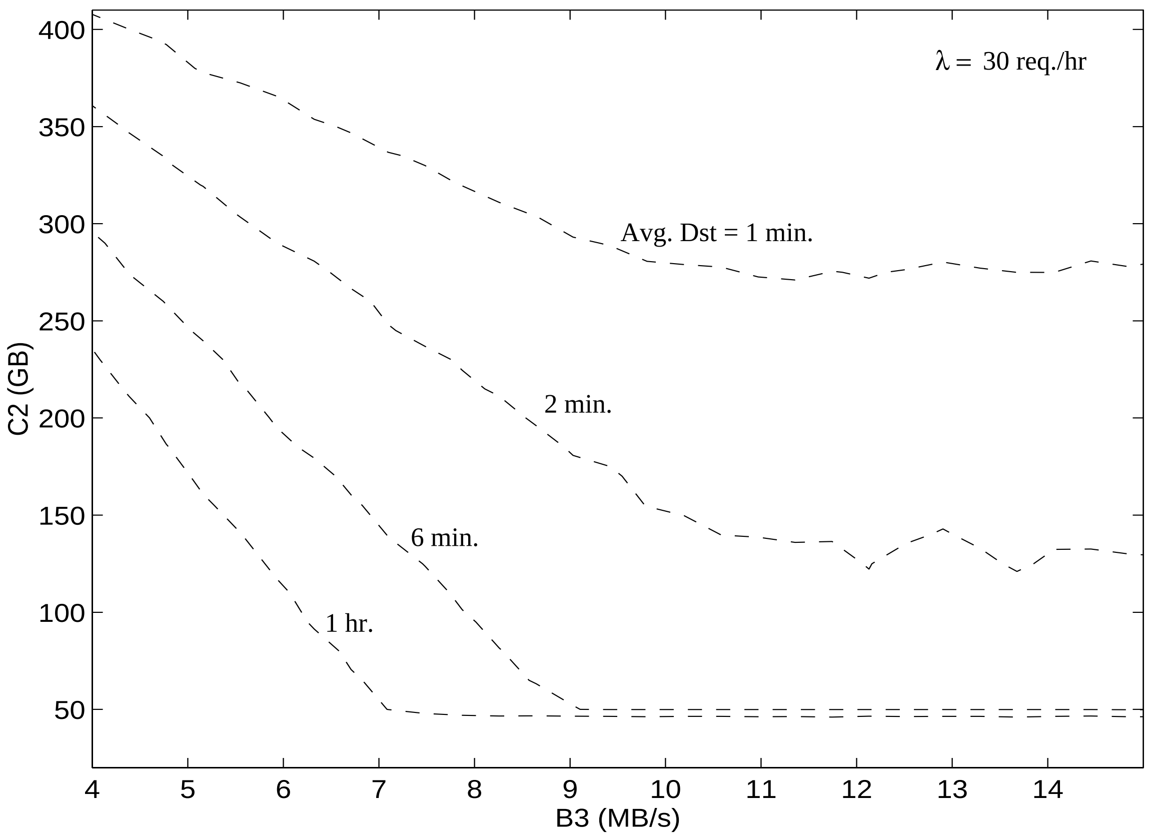

Figure 15: Trade-off between C_2 and B_3 ($\lambda = 30$ req./hr, $B_2 = 20$ MB/s, $N_v = 500$, and uniform video popularity)

long W_d.

Besides trading off capacity with bandwidth, we think it may be possible to trade B_2 off 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, the staging will take longer 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 with $\lambda = 30$ req./hr, with delay ranging from 1 minute to 1 hour. We have used $C_2 = 100$ GB, $N_v = 500$, and uniform video popularity. We remark that there is not much trade-off between B_2 and B_3. Also indicated in the diagram is the linear line $B_2 = B_3 + \lambda T_h b_0$, in accordance to Eq. (12). The line shows the approximate relationship between B_2 and B_3 so that no resources are in excess. Figures 13, 15, and 16 show that B_2 and B_3 can

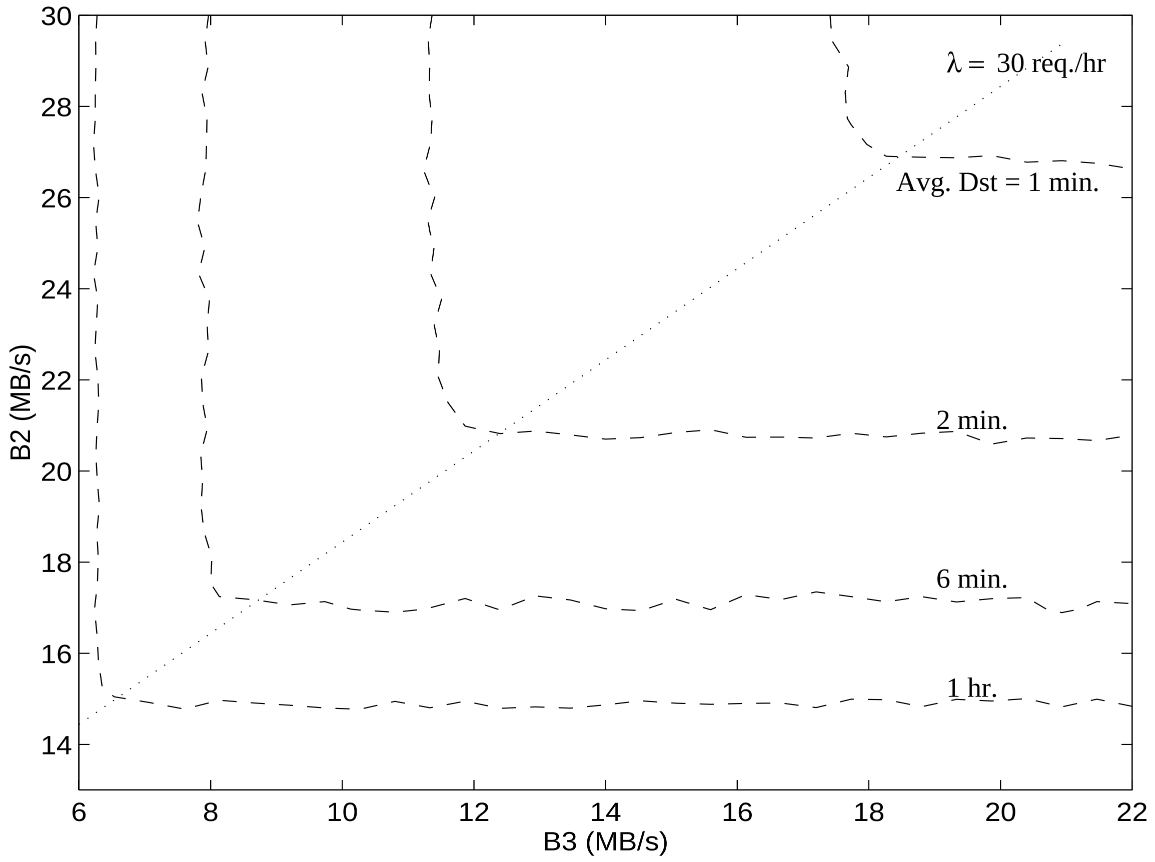

Figure 16: Trade-off between B_2 and B_3, for various values of average start-up delays. The linear line is $B_2 = B_3 + \lambda T_h b_0$. ($\lambda = 30$ req./hr, $C_2 = 100$ GB, $N_v = 500$, and uniform video popularity)

generally be flexibly traded with C_2 but not with each other. Therefore the design space of a hierarchical storage system in terms of architectural parameters is reduced from (C_2, B_2, B_3) to (C_2, B_2) or (C_2, B_3).

5.3 Operational procedures

5.3.1 Request-type priority scheduling

In this section, we show the delay performance using hit-requests-first (HRF) request type priority scheduling. Recall that HRF schedules the hits before it schedules the misses. Figure 17 shows its delay performance along with that of FCFS, where the baseline oper-

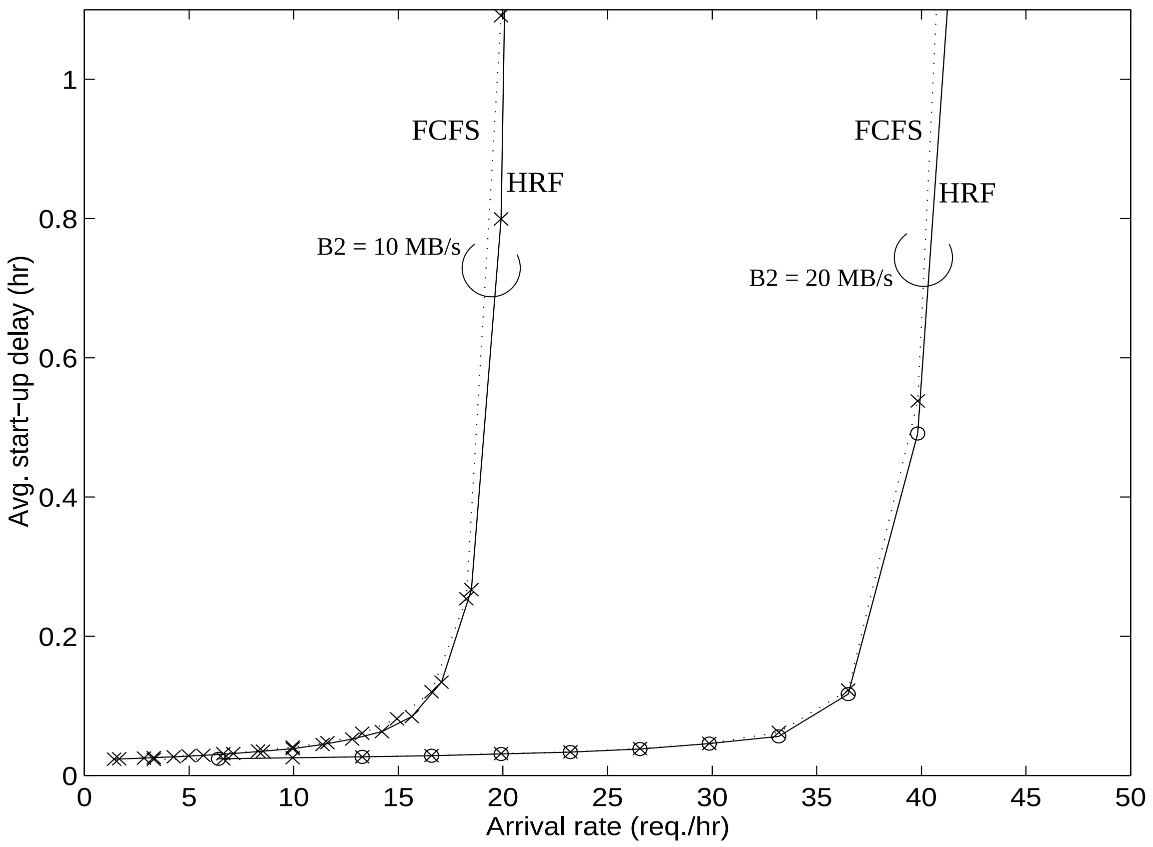

Figure 17: Delay performance using First-Come-First-Served (FCFS) and Hit-Requests-First (HRF) scheduling policies ($C_2 = 100$ GB, $B_2 = 20$ MB/s, $B_3 = 10$ MB/s, $N_v = 500$, and uniform video popularity).

ational procedures are used (except, of course, the request type priority scheduling), with $C_2 = 100$ GB, $B_2 = 20$ MB/s, $B_3 = 10$ MB/s, $N_v = 500$, and uniform video popularity. In terms of average delay, we see that FCFS policy performs similarly as HRF for all traffic.

Even though FCFS and HRF show similar overall average delay, they have markedly different miss and hit delay. Figure 18 shows the average overall delay, miss delay and hit delay with HRF scheduling policy. The figure is best compared with Fig. 3. We see that with HRF, hits enjoy minimal delay even when the load is high ($\lambda \geq 15$ req./hr and 35 req./hr for $B_2 = 10$ MB/s and 20 MB/s, respectively). This is because HRF favors the hits. This point can be made clearer if we compare the delay distribution of both FCFS

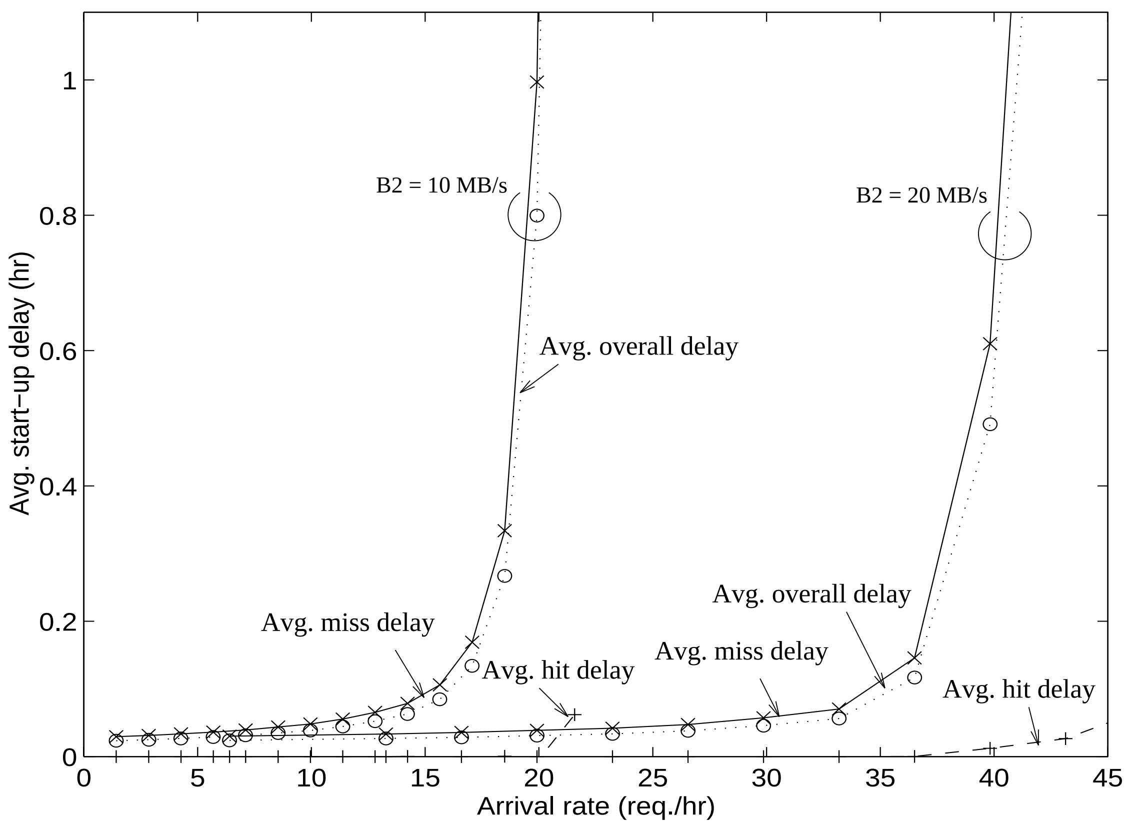

Figure 18: Average overall delay, miss delay and hit delay using Hit-Requests-First (HRF) request-queue scheduling (C_2 = 100 GB, B_2 = 20 MB/s, B_3 = 10 MB/s, N_v = 500, and uniform video popularity)

and HRF scheduling (for B_2 = 20 MB/s) under low (λ = 12.8 req./hr), moderate (λ = 32 req./hr) and high traffic (λ = 44.8 req./hr), which will be shown in Figs. 19, 20 and 21 respectively.

Figure 19 shows the delay distribution and the percentage composition of misses using HRF. We show here a system under low traffic load. Note the marked similarity between this figure and Fig. 4, which shows that FCFS and HRF polices perform similarly in terms of delay under low traffic.

Figure 20 shows the delay distribution under moderate traffic. Compared with Fig. 5, there are more requests with delay less than 1 minute compared with FCFS (20% as

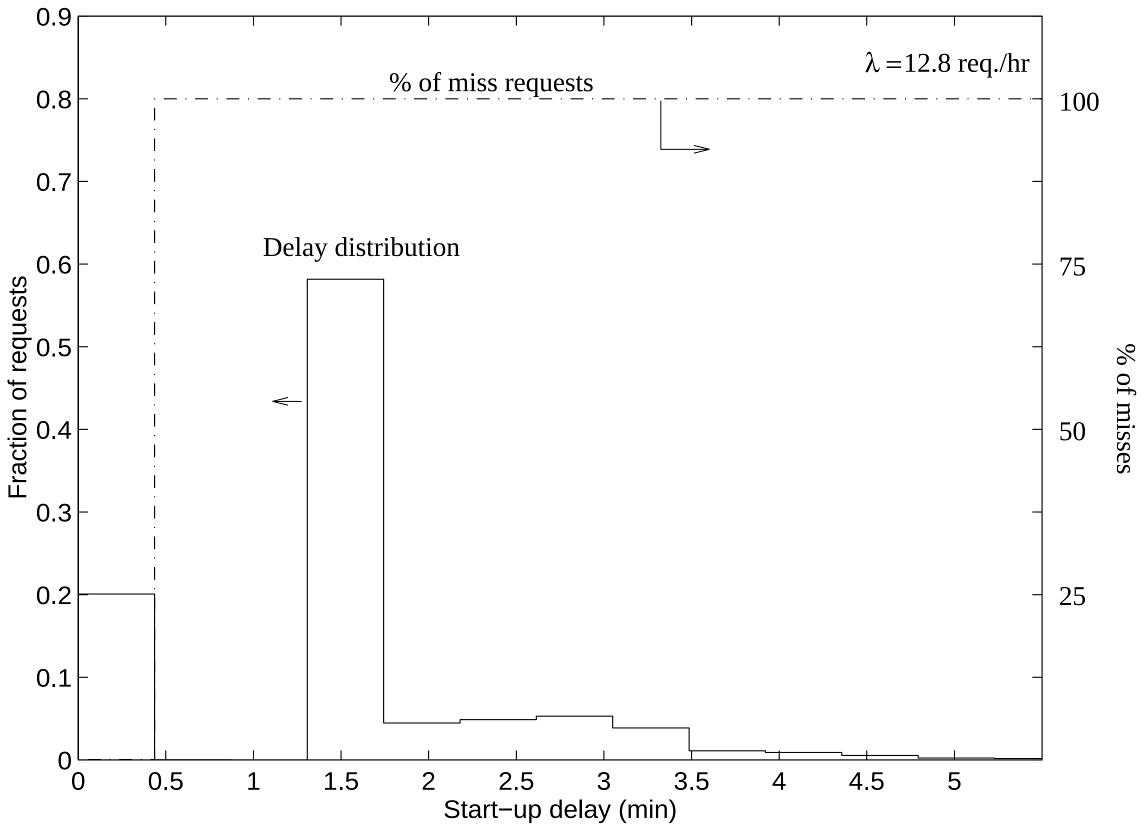

Figure 19: Average delay distribution and its miss composition for HRF scheduling with low arrival rate (C_2 = 100 GB, B_2 = 20 MB/s, B_3 = 10 MB/s, N_v = 500, and uniform video popularity)

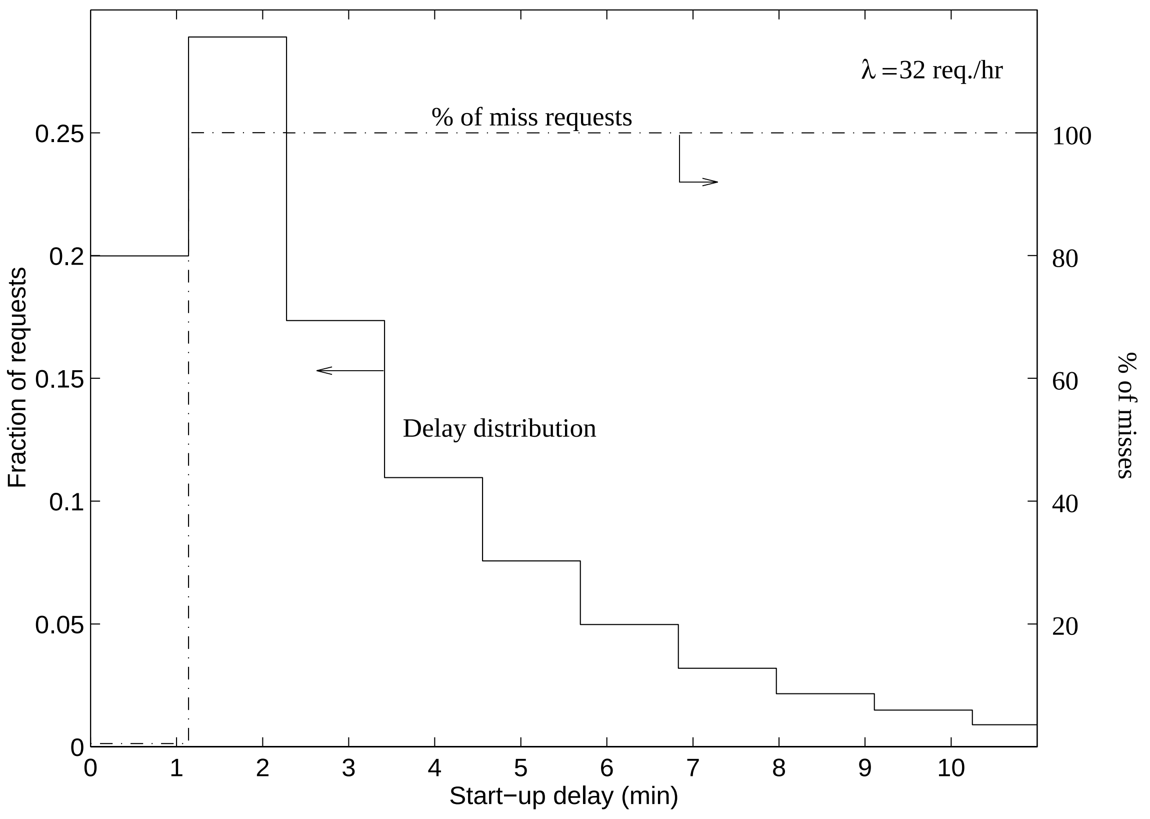

Figure 20: Delay distribution and its miss percentage for HRF scheduling with moderate arrival rate ($C_2 = 100$ GB, $B_2 = 20$ MB/s, $B_3 = 10$ MB/s, $N_v = 500$, and uniform video popularity)

compared to 16%), and slightly more requests with delay between 1 minute and 2 minutes (29% compared with 26%). All hits in the HRF case enjoy minimal delay (i.e., in the range of $0 - 1$ min.), while misses occupy the tail of the distribution. Virtually all the requested videos are displayed within 10 minutes

As we increase the traffic further, the similarity between FCFS and HRF in terms of delay distribution becomes less and less. Figure 21 shows the delay distributions for HRF under heavy traffic. Under heavy traffic, all requests using FCFS scheduling are delayed (see Fig. 6); however, the hits for HRF still enjoy very low delay, with most of them enjoying delay less than 0.3 hr. From simulation trace, we see that in fact a majority of the hits have zero delay. The tail of the distribution is composed entirely of misses.

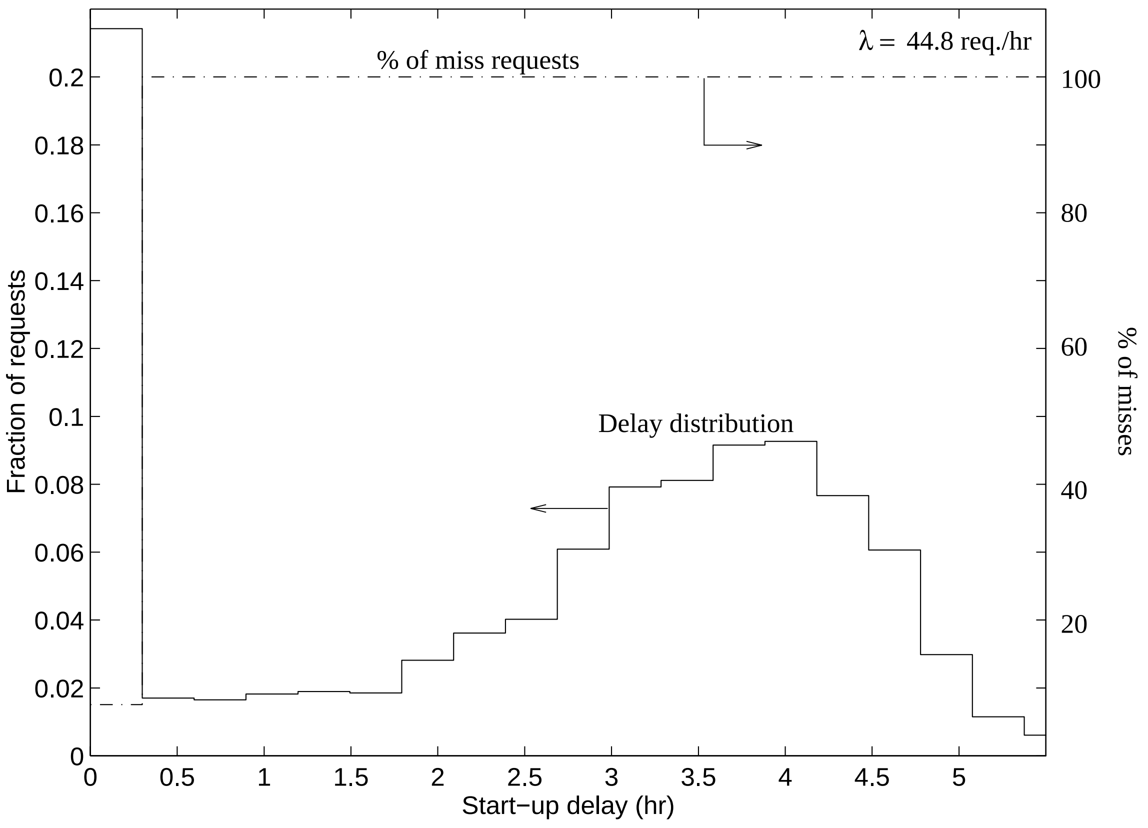

Figure 21: Delay distribution and its miss percentage for HRF scheduling under heavy traffic (C_2 = 100 GB, B_2 = 20 MB/s, B_3 = 10 MB/s, N_v = 500, and uniform video popularity)

Figure 21

We have shown that FCFS and HRF share similar average delay characteristics. However, they have different delay distribution under heavy traffic, in which the hits in HRF scheduling enjoys very little delay while they suffer much higher delay in FCFS scheduling.

5.3.2 Replacement algorithm

In this section, we investigate the performance using the different replaceable-files-selection policy and deletion algorithm mentioned in Section 4.3.

Figure 22 shows the delay performance of the system using two different replaceable-files-selection policies: Closed-Deletable (CD) and Closed-not-Booked-Deletable (CnBD),

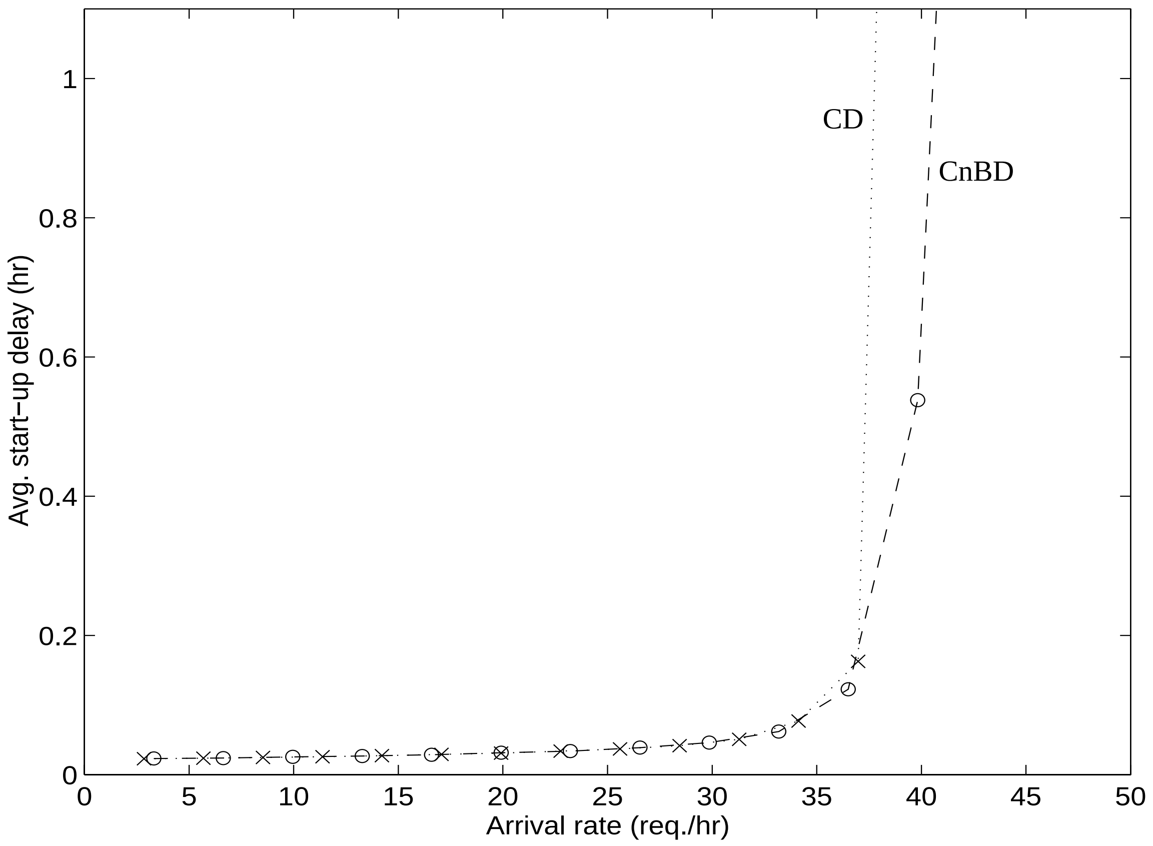

Figure 22: Effect of replaceable-files-selection algorithms (Closed-Deletable (CD) and Closed-not-booked-Deletable (CnBD)) on the performance of hierarchical storage system ($C_2 = 100$ GB, $B_2 = 20$ MB/s, $B_3 = 10$ MB/s, $N_v = 500$, and uniform video popularity.)

with $C_2 = 100$ GB, $B_2 = 20$ MB/s, $B_3 = 10$ MB/s, $N_v = 500$ (uniform video popularity), and our usual baseline operations (except, of course, the replaceable-files-selection algorithm). We see from the figure that for low to moderate traffic rate ($\lambda \leq 35$ req./hr), CD and CnBD give similar performance. However, at higher arrival rates, the system using CD shows much higher delay. This is due to the phenomenon much similar to "thrashing" in operating system: According to the CD algorithm, a resident file not currently being displayed can be deleted. As a result, some booked video files may be unwisely deleted. These files have to be brought back again in the short future, leading to very high overheads (in terms of staging time, bandwidth and exchanges). This leads to a decrease in system performance. However, the difference seems to be in the (less interesting) region above the knee when the delay increases very sharply.

Figures 23 and 24 show the performance of the hierarchical storage system using different deletion algorithms: Least-Recently-Used (LRU), Least-Popular-First (LPF) and Random replacement (RND), using uniform video popularity and 80/20 non-uniform video popularity respectively. With uniform video popularity, as expected, there is no difference in performance using any of the algorithms. By comparing with the uniform case, we see that with 80/20 video popularity, there is an extension in streaming capacity and a great decrease in delay for all three deletion algorithms. LPF obviously has the lowest average delay performance compared with the other algorithms, because it makes the wisest decision in deleting non-permanent-resident files, and hence minimizes staging overheads. If video popularity is known, LPF is therefore the best deletion algorithm to use. RND performs surprisingly well compared with LRU algorithm. In choosing a deletion algorithm, therefore, the simple RND may be used instead of the more sophisticated LRU algorithm.

5.3.3 Staging-request scheduling

Figures 25 and 26 show the performance of the hierarchical storage system using different staging-request scheduling algorithms, with uniform and 80/20 non-uniform video popularity respectively. The algorithms we consider here are: First-Come-First-Served (FCFS),

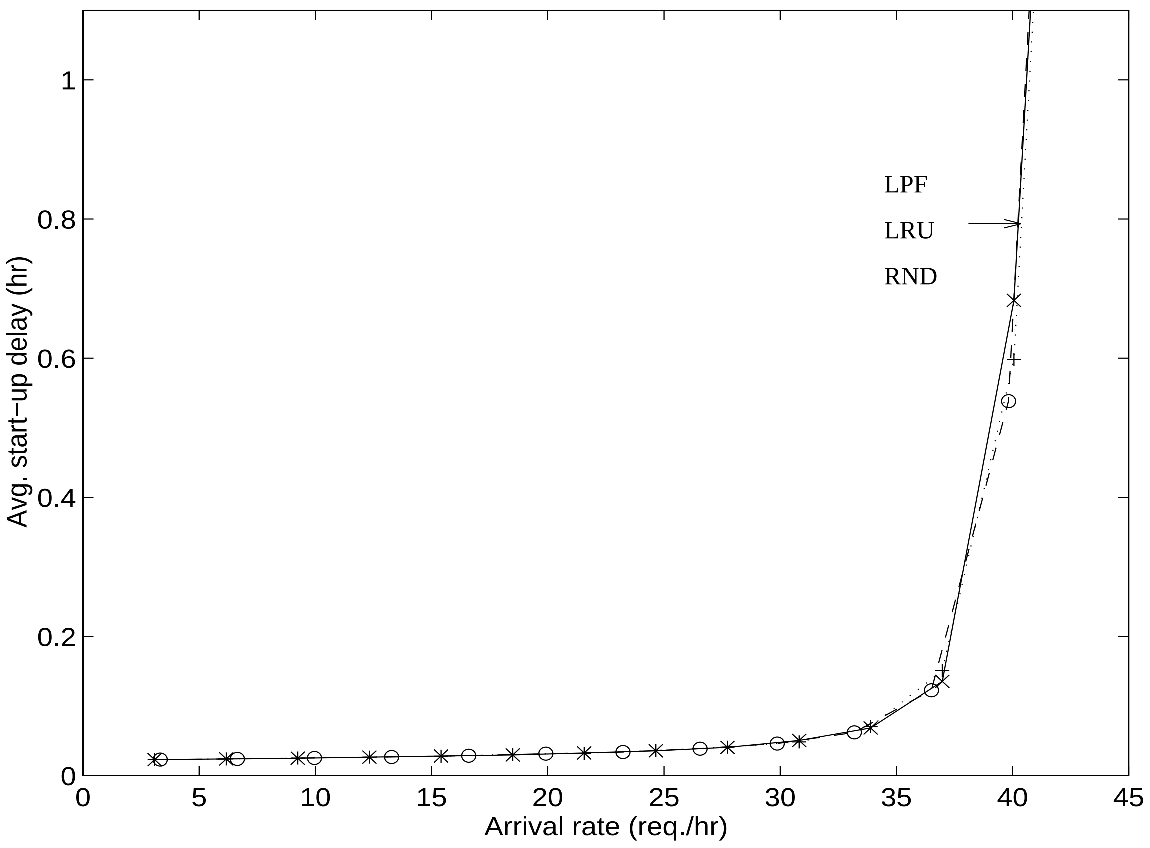

Figure 23: Under uniform video popularity, different deletion algorithms (Least-Recently-Used, Least-Popular-First (LPF) and Random replacement (RND)) give the same average delay (C_2 = 100 GB, B_2 = 20 MB/s, B_3 = 10 MB/s, N_v = 500, and uniform video popularity.)

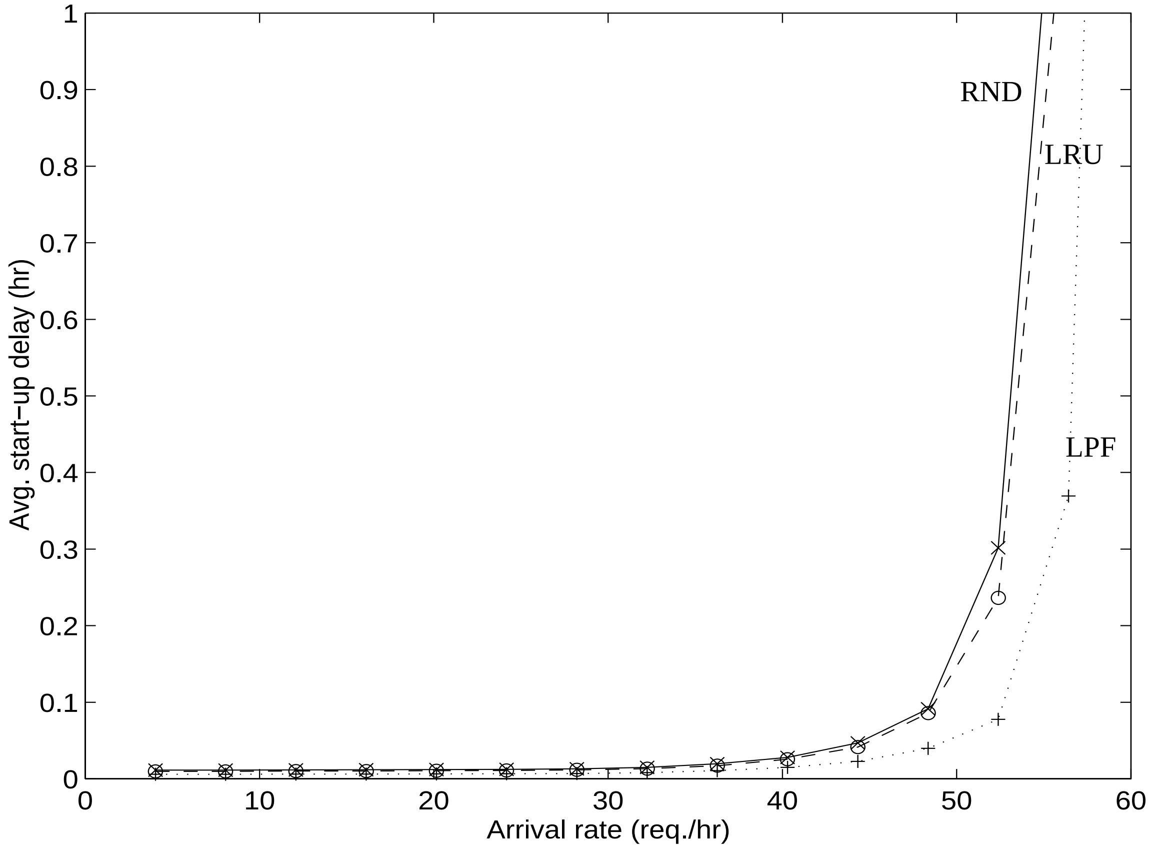

Figure 24: Performance of the hierarchical storage system using different deletion algorithms (Least-Popular First (LPF), Least-Recently-Used (LRU) and Random replacement (RND)), under 80/20 non-uniform video popularity ($C_2 = 100$ GB, $B_2 = 20$ MB/s, $B_3 = 10$ MB/s, $N_v = 500$, and 80/20 non-uniform video popularity.)

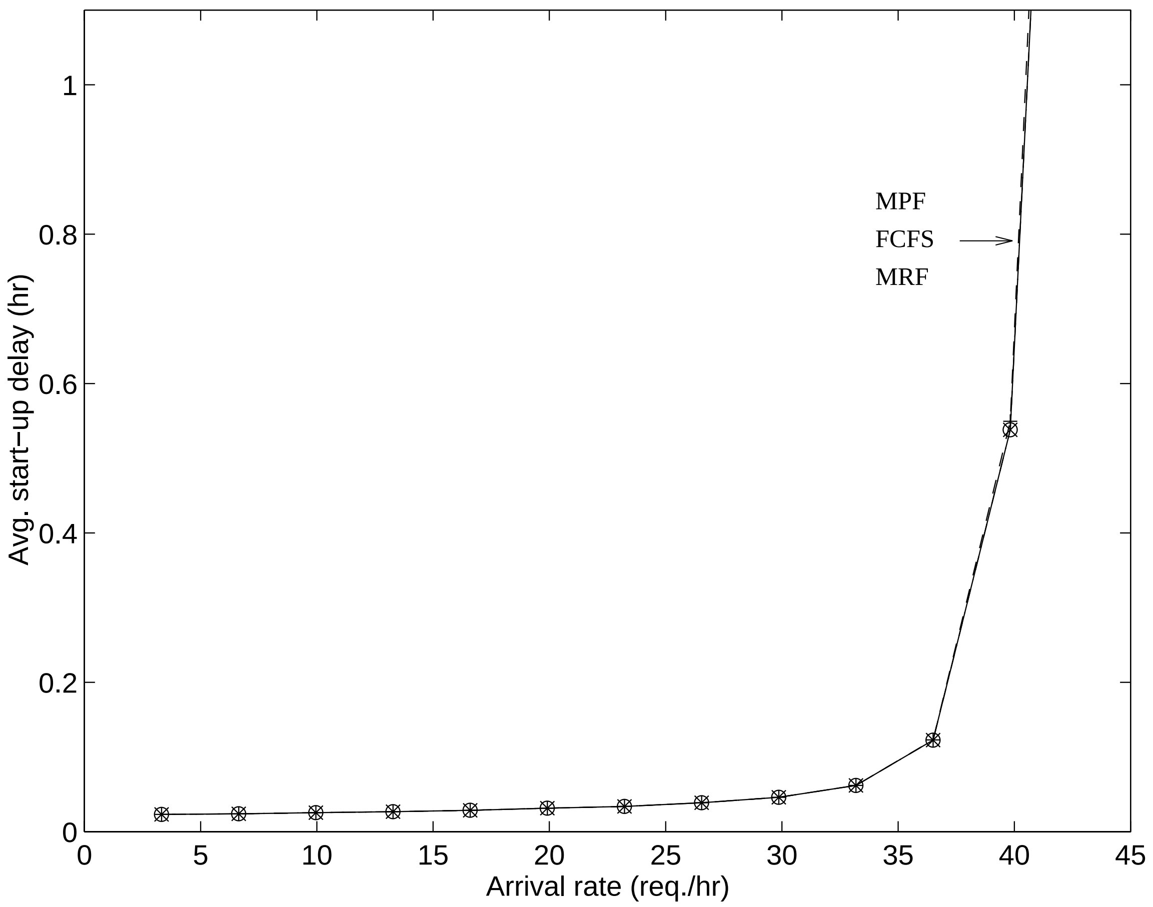

Figure 25: Performance of the hierarchical storage system using different staging-request scheduling algorithms, with uniform video popularity. (C_2 = 100 GB, B_2 = 20 MB/s, $B_3 = 10$ MB/s, and $N_v = 500$)

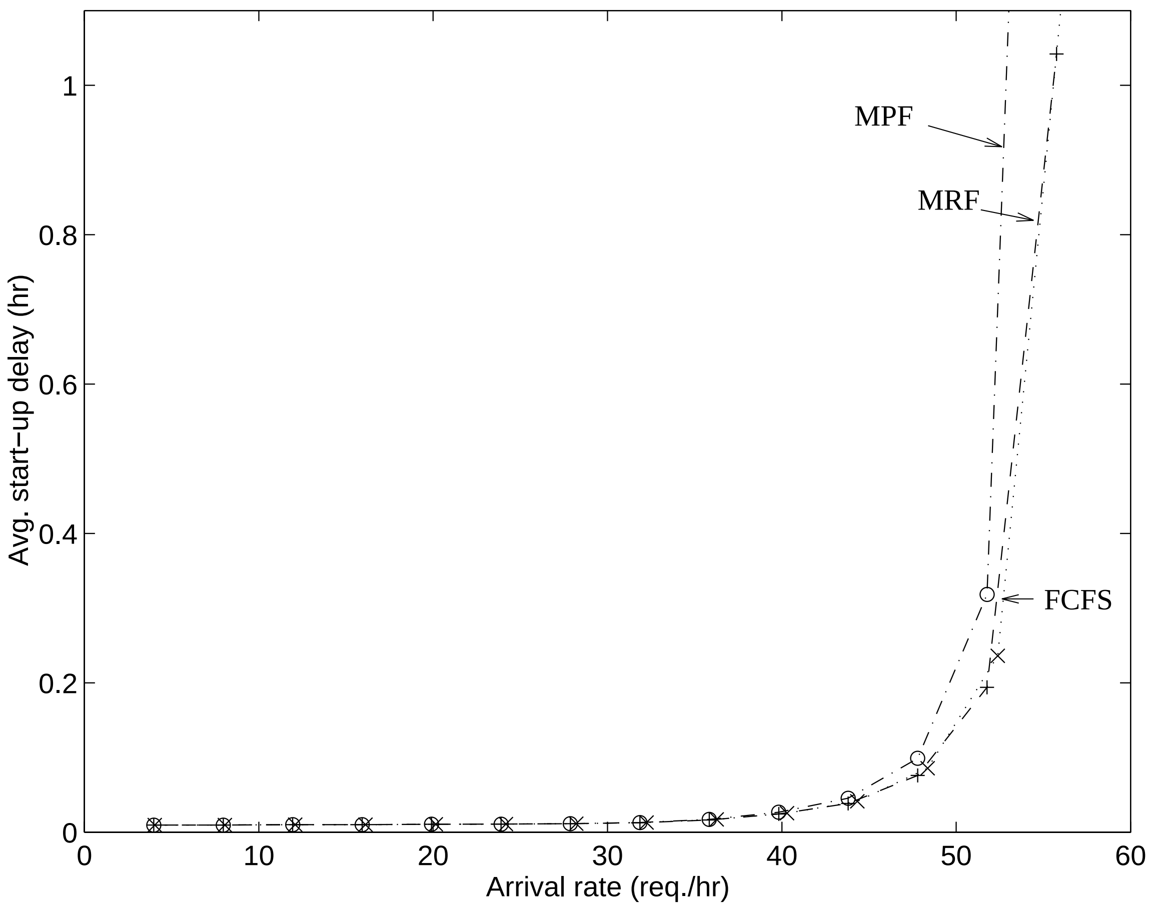

Figure 26: Staging-request scheduling in the performance of the hierarchical storage system, with 80/20 non-uniform video popularity ($C_2 = 100$ GB, $B_2 = 20$ MB/s, $B_3 = 10$ MB/s, and $N_v = 500$)

Most-Popular-First (MPF), and Most-Requests-First (MRF). With uniform popularity, as expected, all algorithms performs similarly.

With non-uniform video popularity, besides a marked decrease in delay and a great extension in streaming capacity, we see that the three algorithms show similar performance up to moderate traffic ($\lambda \leq 45$ req./hr); under high traffic ($\lambda \geq\sim 55$ req./hr), MPF performs the worst. This is because MPF stages the most popular file first. Therefore, misses for unpopular videos may suffer very long delay. This is especially apparent under high traffic, where such misses may never get served as their staging schedule is very probably delayed by an arrival requesting a more popular file. Therefore, MPF is a poor policy to use.

Though the systems using MRF and FCFS exhibit similar average delay, the users are treated differently. In FCFS, the staging requests for unpopular files share similar delays with the requests for more popular files. However, in MRF, requests for unpopular files have to wait much longer before their cumulative outstanding requests is high enough for their videos to be staged. In other words, the waiting time using MRF is dependent on the popularity of the video selected.

From the figures we presented in this section, we see that the difference in performance between various operational procedures is little. It is also apparent that the architectural parameters is more important in determining the performance of the hierarchical system.

5.4 Applications characteristics

In this section, we examine how changes in applications characteristics affect the performance of a hierarchical storage system. We will investigate the effects of video popularity, the number of video titles, streaming bandwidth (b_0), video length (T_h), and the distribution of the holding time.

Figure 27 shows the influence of video popularity in the performance of the server, in which we use uniform and non-uniform 80/20 video popularity. We consider as usual our baseline operational procedures with $B_2 = 20$ MB/s, $B_3 = 10$ MB/s, and $N_v = 500$. For uniform video popularity, we show $C_2 = 100$ GB while for non-uniform video popularity, we show $C_2 = 100$ GB and 50 GB. We see immediately that non-uniform video popularity is able to cut the storage requirement by more than half. Therefore, skewed video popularity leads to tremendous decrease in average delay, which ultimately is able to translate to a decrease in storage and bandwidth requirements.

Figure 28 shows the delay performance of an on-demand video server as the number of video titles increases from 100 – 1000 with four arrival rates, $\lambda = 20$ requests/hr, 35 requests/hr, 40 requests/hr and 42 requests/hr. We again have $B_2 = 20$ MB/s, $C_2 = 100$ GB, $B_3 = 10$ MB/s, and uniform video popularity. We see from the figure that, as

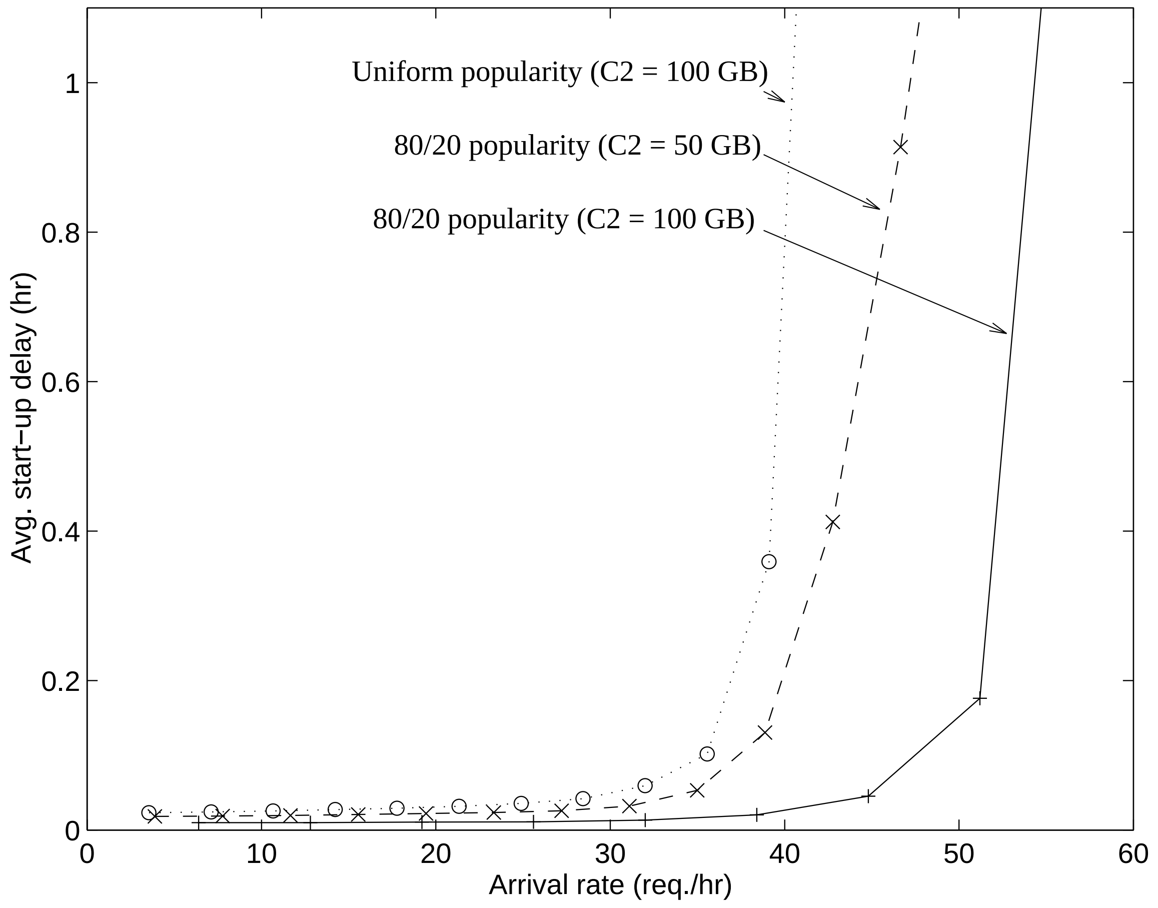

Figure 27: Influence of video popularity on the performance of a hierarchical storage video server ($B_2 = 20$ MB/s, $B_3 = 10$ MB/s, $N_v = 500$, and uniform popularity)

Table 2: Systems of investigation by changing b_0 from 1.5 Mb/s to 3 Mb/s.

System	b_0 (Mb/s)	C_f (GB)	C_2 (GB)	B_2 (MB/s)	B_3 (MB/s)
SYS0	1.5	1	100	20	10
SYS1	3	2	100	20	10
SYS2	3	2	200	20	10
SYS3	3	2	200	20	20
SYS4	3	2	200	40	10
SYS5	3	2	200	40	20

the number of video files is increased, the performance of the hierarchical storage system decreases. In order to guarantee stability as $N_v \to \infty$, it is not hard to see that,

$$\lambda < \mu_3, \tag{20}$$

where μ_3 is the service rate of the tertiary level given by B_3/C_f. From the figure, we also see that the server is scalable in terms of video titles under only low utilization (i.e., when $\lambda \leq\sim 25$ req./hr). In fact, it is not difficult to see that, for low utilization and in the limit as $N_v \to \infty$,

$$\lim_{N_v \to \infty} \overline{D}_{st} = \overline{D}_{miss} \tag{21}$$

$$\approx \frac{1}{\mu_3} + \frac{\lambda/\mu_3}{2(\mu_3 - \lambda)}. \tag{22}$$

Figure 29 shows the effect of changing the streaming bandwidth b_0 from 1.5 Mb/s to 3 Mb/s, where the parameters of the six systems we consider are shown in Table 2. We still consider constant holding time with $T_h = 90$ minutes. SYS0 is our baseline system.

Note that SYS0 and SYS5 have the same performance, showing that if the streaming bandwidth b_0 (and hence the file size) is changed, C_2, B_2, and B_3 all have to be changed by the same factor in order to preserve the performance. This "invariance" is expected when we see that as b_0 is changed, the total streaming bandwidth $B_2^{strm}(t)$ is also changed

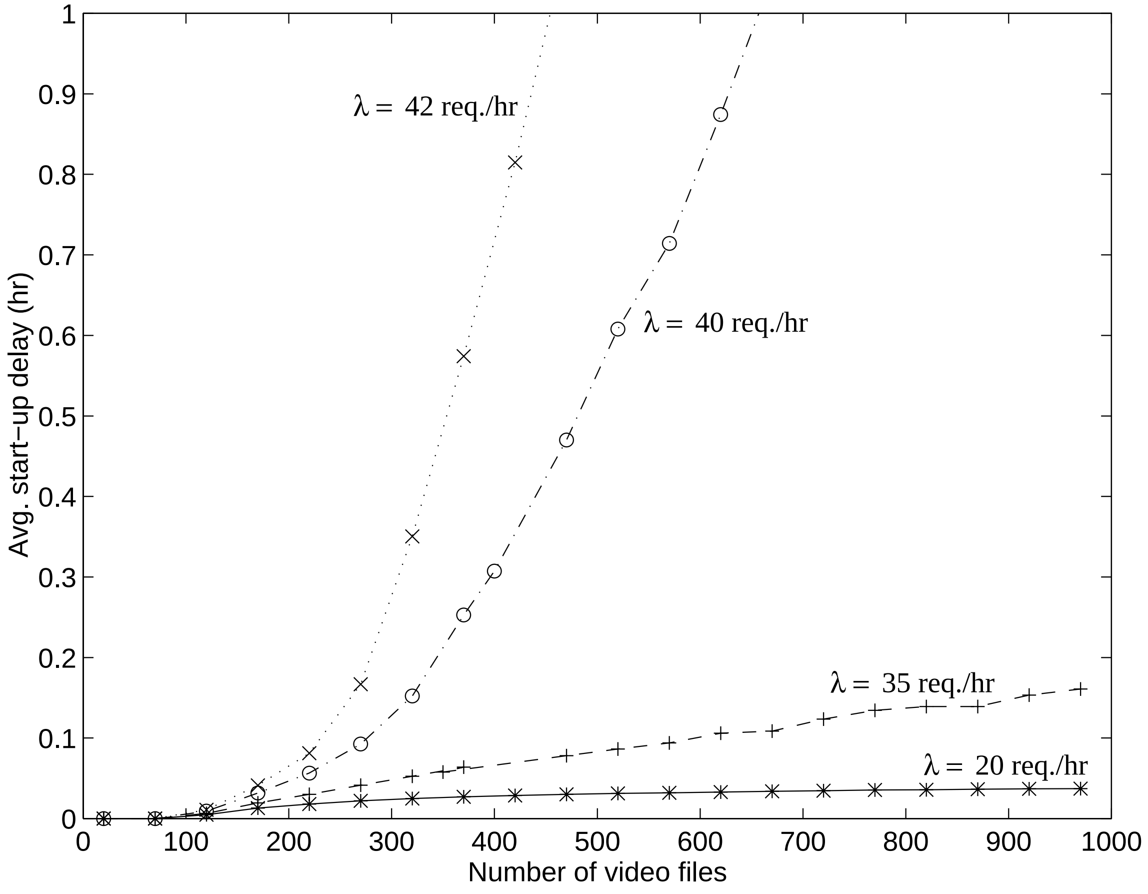

Figure 28: Delay performance of a hierarchical server versus N_v ($B_2 = 20$ MB/s, $C_2 = 100$ GB, $B_3 = 10$ MB/s, and uniform video popularity)

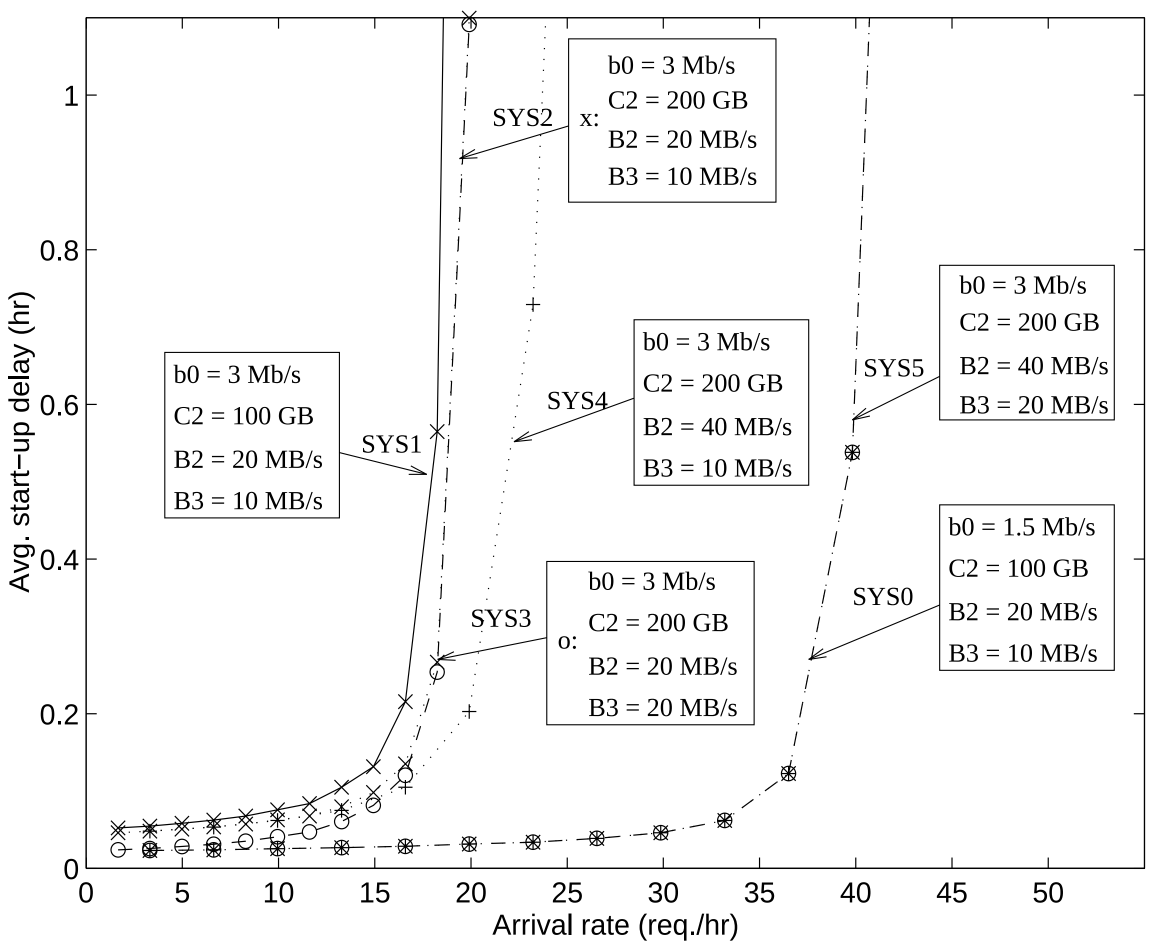

Figure 29: Change in b_0 (from 1.5 Mb/s to 3 Mb/s) in the performance of a hierarchical storage system ($N_v = 500$ and uniform video popularity.)

Table 3: Systems of investigation when the video holding time, T_h, changes from 90 min. to 45 min. We have kept streaming rate to be constant (1.5 Mb/s).

System	C_f (GB)	T_h (min.)	C_2 (GB)	B_2 (MB/s)	B_3 (MB/s)
SYS0	1	90	100	20	10
SYS1	0.5	45	100	20	10
SYS2	0.5	45	50	20	10
SYS3	0.5	45	50	20	5
SYS4	0.5	45	50	10	10
SYS5	0.5	45	50	10	5

accordingly. Furthermore, as file size is changed because of b_0, C_2 has to be scaled by the same factor; and to keep the staging time the same, B_3 also has to be scaled by the same factor as well.

We see that as b_0 is increased, if we keep the other parameters the same (compare SYS0 and SYS1), the system capacity is decreased approximately by a factor of two. We also see that SYS2 and SYS3 have similar delay performance under high traffic (less interesting region), i.e., $\lambda \geq 18$ req./hr, while SYS3 shows lower delay at light traffic due to its higher B_3 (note that $\lim_{\lambda \to 0} \overline{D}_{st} = (1-\alpha_2)C_f/B_3$). SYS4 doubles its B_2 to 40 MB/s, which leads to lower delay for high traffic in comparison with SYS2 and SYS3, but such a high B_2 does not lead to higher performance at low traffic when compared with SYS3. From above, we see that at low arrival rates, the staging process determines the overall delay of the users, and hence increasing B_3 is more effective in decreasing user delay compared with increasing B_2.

Figure 30 shows the effect in changing the video holding time T_h on the performance of the system. We consider the systems as shown in Table 3, in which we have kept $b_0 = 1.5$ Mb/s. SYS0 is our baseline system.

We see that SYS0 and SYS5 exhibit the same performance, showing that as T_h (and

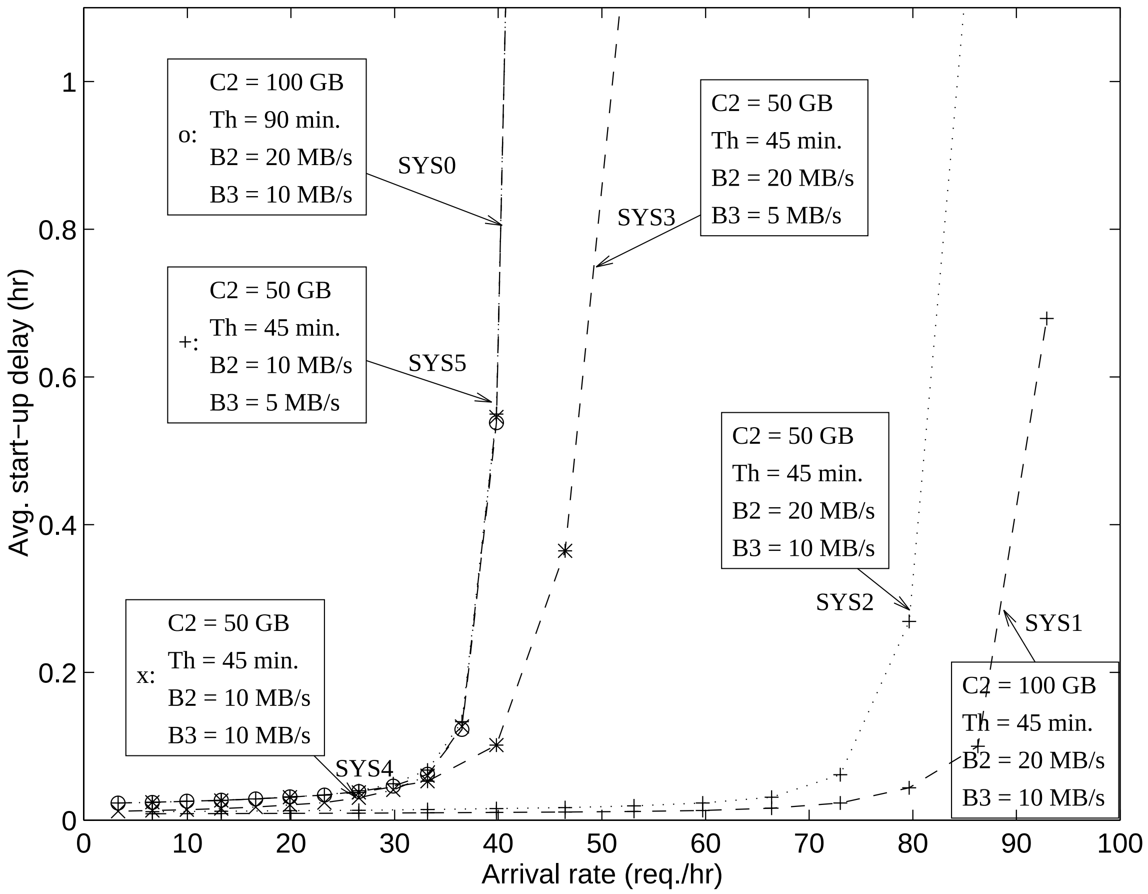

Figure 30: Performance of the hierarchical storage system as the video length, T_h, is changed from 90 minutes to 45 minutes ($N_v = 500$ and uniform video popularity)

the corresponding file size) is changed, C_2, B_2 and B_3 all have to be scaled by the same factor in order to preserve performance.

SYS5 and SYS4 perform similarly in high traffic ($\lambda \geq 36$ req./hr), but SYS4 performs better when traffic is low due to its higher B_3. Among all the systems examined, SYS1 has the best delay performance, followed by SYS2. By comparing SYS2 with SYS0, we see that a doubling in the holding time, keeping other factors the same, can decrease the streaming capacity of the system by more than half.

Figure 31 shows the effect of the distribution of the holding time, T_h, in the performance of the system. We consider three cases with the same mean:

1. Constant holding time with $T_h = 90$ min.;
2. Uniform holding time $\sim$ U[60 min., 120 min.]; and
3. Exponential holding time $\sim \exp[\mu]$, with $1/\mu = 90$ min.

We still use the baseline operations (except, of course, its holding time distribution) with $C_2 = 100$ GB, $B_2 = 20$ MB/s, $B_3 = 10$ MB/s, $N_v = 100$, and uniform video popularity.

Despite the differences in the holding time distribution, all systems have similar performance. Therefore, the holding time distribution is not a major factor in determining the start-up delay of the server. As a matter of fact, we have already seen from Fig. 3 that under low to moderate traffic, the major component in determining delay is the miss delay in the tertiary level. This delay depends on the tertiary characteristics and does not depend so much on the holding time distribution.

6 System Design

In Sect. 5, we show the performance characteristics of a hierarchical storage system for video server. We show in this section how such characteristics can be used to design a video server to meet some specific performance requirements. In particular, we address

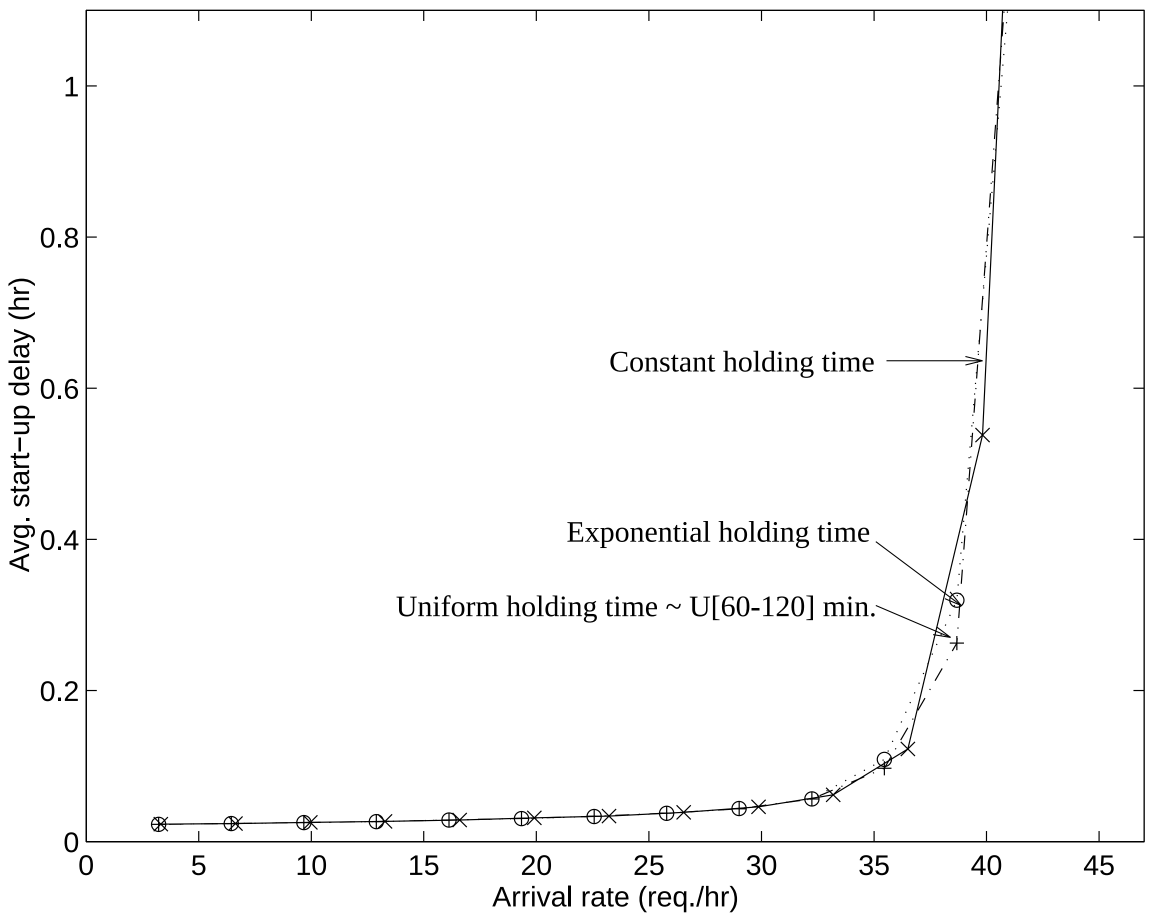

Figure 31: Influence of the holding time distribution on the delay performance of a hierarchical storage system (C_2 = 100 GB, B_2 = 20 MB/s, B_3 = 10 MB/s, N_v = 500, and uniform video popularity)

Table 4: Storage and bandwidth requirements to satisfy a 2-minutes average delay with $\lambda = 30$ req./hr ($B_3 = 10$ MB/s, $N_v = 500$, and uniform video popularity)

# Magnetic disks	GB/Disk	MB/s
8	19.4	2.5
16	9.7	1.25
32	4.8	0.625

how such a server can be designed so as to meet certain start-up delay requirement, based on our baseline specifications given in Sect. 4.4.

From Fig. 13, we see that given a certain arrival rate, one can always find pairs (B_2, C_2) in order to satisfy a certain average delay requirement. For example, given a certain arrival rate $\lambda = 30$ req./hr and delay requirement of 2 minutes, one such pair is (B_2, C_2) = (20 MB/s, 155 GB). These requirements can be satisfied by using different number of magnetic disks and disk capacity according to Table 4.

Figure 13 shows that if there is no relationship between disk capacity and disk bandwidth, a hierarchical storage system would be likely operated around the trade-off knee. However, a magnetic disk nowadays does have limited bandwidth and storage capacity. In this section, we will show how a hierarchical storage system can be designed given the current disk technology. We will see that such practical consideration would lead us to design a server away from the trade-off knee, leading to excess B_2.

A magnetic disk comes with its storage capacity C_{dsk} and a certain effective disk bandwidth B_{dsk}.[8] Several magnetic disks can be put together to achieve a higher total capacity, while at the same time increasing the total bandwidth (e.g., as in disk array). If we assume that the total storage and bandwidth is proportional to the number of disks, and let B_2 be the total effective bandwidth required in the secondary level, we then have the following

[8]Here, effective disk bandwidth is the bandwidth of a disk after taking into account of access latency, data layout, and access algorithm.

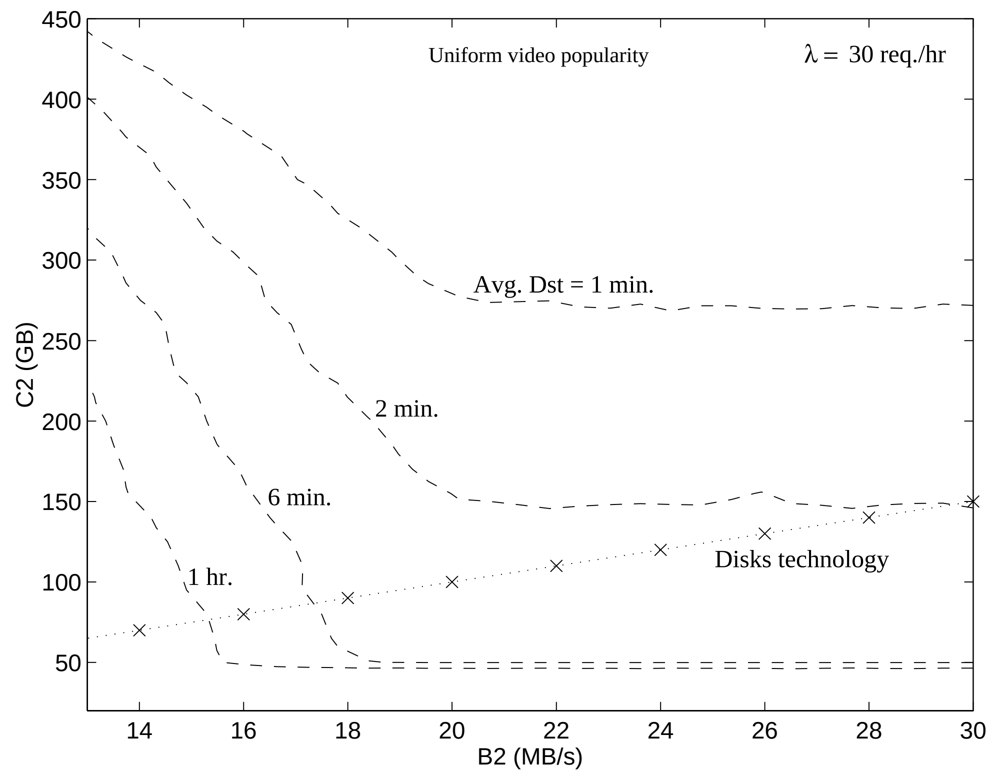

Figure 32: Trade-off between C_2 and B_2 with the "disk technology" line included ($\lambda = 30$ req./hr, $B_3 = 10$ MB/s, $N_v = 500$, negligible T_{ex}, and uniform video popularity)

"disk technology" relationship:

$$C_2 \quad = \quad \left(\frac{B_2}{B_{dsk}}\right) C_{dsk}. \tag{23}$$

Consider that $B_{dsk} = 2$ MB/s, and given that currently $C_{dsk} \leq 10$ GB, we therefore have $C_2 \leq (B_2/2)10 = 5B_2$. The line is shown as the "disk technology" line in Figure 32, reproduced from Fig. 13. The region below the line is the current "technologically feasible" region. Clearly, we see that good operating points of a server are not necessarily around the knees of the trade-off curves, after taking into account our current storage technology.

Here, we give examples of how a hierarchical storage system can be designed to satisfy certain delay requirements, say $\overline{D}_{st} = 30$ seconds and 2 minutes, under arrival rate $\lambda = 30$ req./hr. We show in Figure 33 specifically the delay contours corresponding to $\overline{D}_{st} = 30$ seconds and 2 minutes. We have also shown three different values of B_3: 8 MB/s, 10 MB/s and 12 MB/s, assuming $N_v = 500$, and uniform video popularity.

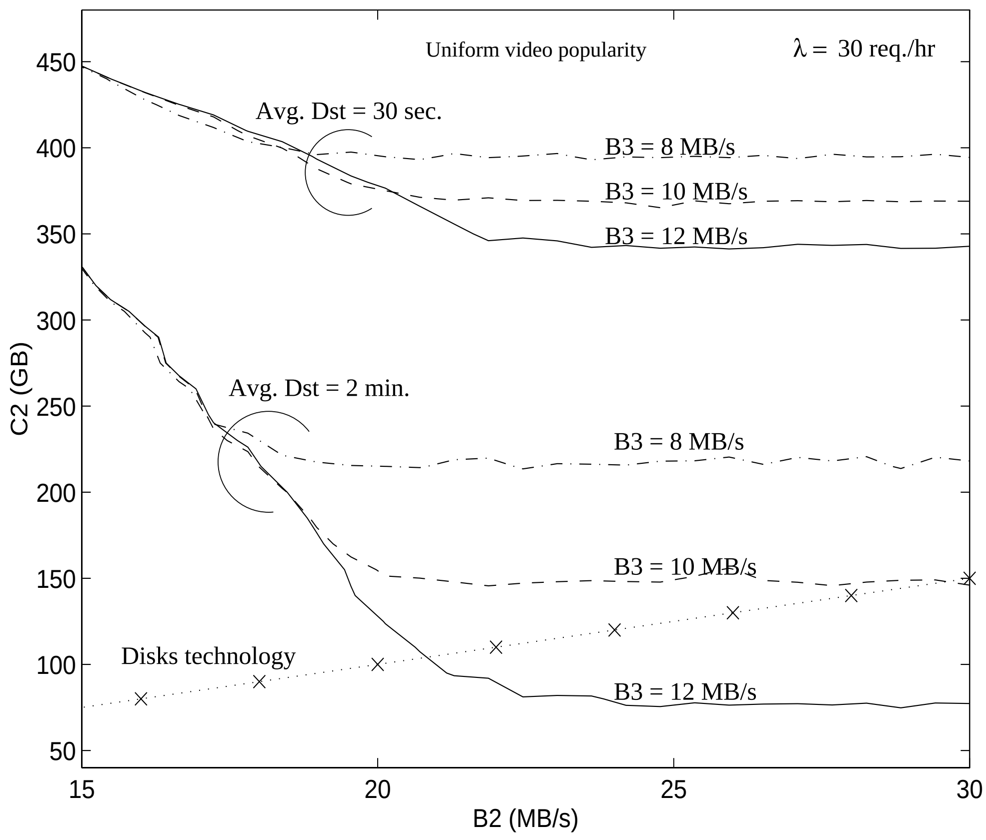

Figure 33: The design of a hierarchical storage system with $\overline{D}_{st} = 30$ seconds and 2 minutes, with $\lambda = 20$ req./hr ($N_v = 500$, and uniform video popularity)

From the interception between the "disk technology" line and the delay contours, we can deduce the requirements of the secondary level in order to satisfy the average delay requirement. Such requirements in terms of storage capacity and bandwidth are shown in Tab. 5. Note that in the case that $\overline{D}_{st} = 2$ minutes and $B_3 = 10$ MB/s, since the "technology" line intercepts the delay contour well into the horizontal asymptote region, we have a lot of excess B_2. We also see that a good operating point for a server satisfying 2 minutes delay with $\lambda = 30$ req./hr is $(C_2, B_2, B_3) = $ (80 GB, 22 MB/s, 12 MB/s). Given our current technology, the hierarchical storage system can hardly satisfy $\overline{D}_{st} = 30$ seconds with our parameters without incurring excessive storage and bandwidth requirements.

How does non-uniform video popularity affect our design? Figure 34 shows the similar plot as Fig. 33, but with non-uniform 80/20 video popularity. From the figure, we can

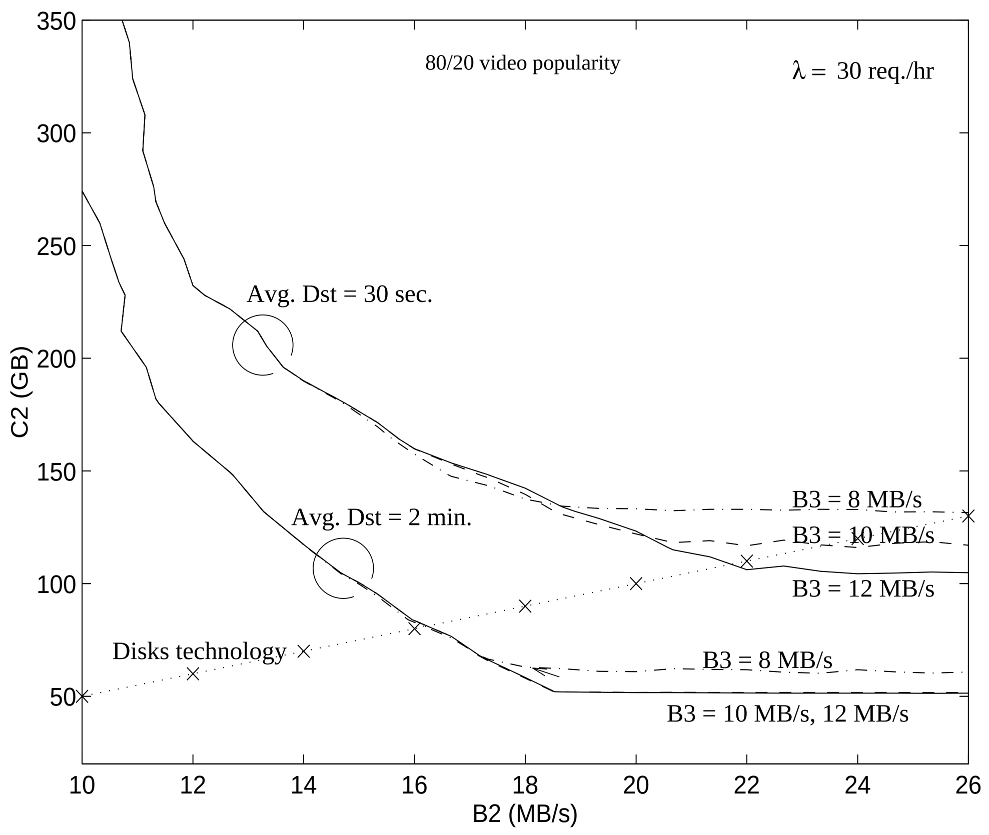

Figure 34: Trade-off between C_2 and B_2, with 80/20 video popularity ($\lambda = 30$ req./hr and $N_v = 500$)

obtain the storage and bandwidth requirements as shown in Tab. 5, in which the resources in excess are highlighted.

From the table, we see that non-uniform video popularity can tremendously reduce the requirements in both B_2 and C_2 compared with the uniform case, sometimes by more than half. For example, the design for a hierarchical storage system satisfying 30-seconds delay under uniform video popularity is almost impossible, but with non-uniform popularity, it is very feasible. In fact, with $B_3 = 12$ MB/s, the storage necessary is only about 20% of the total storage required to keep all the videos resident.

With $\overline{D}_{st} = 2$ minutes, under non-uniform popularity, we see that there is no much gain in C_2 and B_2 as B_3 is increased from 10 MB/s to 12 MB/s. Therefore, using $B_3 = 10$ MB/s is sufficient. For uniform case, we should use $B_3 = 12$ MB/s, as this does not lead

Table 5: Storage and bandwidth requirements for a server under a certain delay constraint constraint ($\lambda = 30$ req./hr, $B_3 = 10$ MB/s, and $N_v = 500$)

$\overline{D}_{st}$	B_3	Uniform popularity				80/20 video popularity			
		B_2 (MB/s)	C_2 (GB)	#disks	GB/disk	B_2 (MB/s)	C_2 (GB)	#disks	GB/disk
30 sec.	8	*78*	390	39	10	*26*	130	13	10
	10	*72*	360	36	10	*24*	115	12	9.6
	12	*68*	340	34	10	22	105	11	9.55
2 min.	8	*44*	220	22	10	18	60	9	6.7
	10	*30*	150	15	10	18	55	9	6.1
	12	22	85	11	7.73	18	$\approx$55	9	6.1

to excessive resources.

7 Conclusions and Future Works

Video-on-demand (VOD) refers to services in which users are able to request their videos on demand. It encompasses such applications as movie-on-demand, home-shopping, news-on-demand, various distributed/interactive training programs, etc. In this paper, we have presented some of the important VOD applications characteristics (characteristics in user demand, video files, user interactions, and performance requirements) and performance goals in designing a video server (meeting application performance requirements, cost-effectiveness and robustness/scalability).

Video servers based on hierarchical storage systems offer high-capacity and low-cost video storage. The hierarchical storage system consists of a secondary level for video presentation; and a tertiary level for video staging. The design space of a hierarchical storage system consists of numerous architectural parameters and operational procedures. Architectural parameters include bandwidths and storage capacities in both levels, while operational procedures deal with various scheduling policies and priority schemes. In this paper, we have studied the architecture and operation of a server based on hierarchical storage system which offers interactive capability to its users, i.e., a user has complete control and freedom in interacting with the playback video.

Using an event-driven simulation, we have addressed various performance issues of such a hierarchical storage system. The performance measure we mainly considered here is user delay. In terms of scalability, we found that the performance of the hierarchical storage system scales well with the number of video titles only under low bandwidth utilization. We have also found that, as the user holding time or the video streaming bandwidth changes, the secondary storage and the bandwidths in the secondary and tertiary levels have to be scaled accordingly in order to achieve similar performance. Such "invariance" would prove very useful in designing the server. We have also shown that the delay performance of the server does not depend sensitively on the distribution of the user holding time besides its mean.

In terms of operational procedures, we found that scheduling based on simple FCFS policies perform almost as well as other scheduling policies based on prioritized or more complicated schemes. For file replacement algorithm, random replacement performs almost as well as least-recently-used (LRU). However, in deleting a file, we found that decision taking into account the state of the queue, and hence knowing what video files would need to be displayed in the future, would achieve less overheads and higher performance.

Since there are two types of users in the system (hits and misses) whose delay may differ markedly, we have also studied the delay distribution of the system. Generally, hits enjoy much lower delay than misses. Therefore, total staging delay is the biggest component for overall user delay. Having an efficient tertiary level is therefore very important in designing a high-performance hierarchical storage system for video servers. To this end, we found that maximum drive parallelism in the tertiary level achieves lowest user start-up delay.

Our study shows that, to achieve a certain delay performance, bandwidths and storage capacity can generally be traded off with each other up to a certain "limit" beyond which no more beneficial "trade-off" is possible. However, bandwidths in the secondary and tertiary levels cannot be flexibly traded off with each other, and hence the design space of a hierarchical storage system can be reduce. We have also found that architectural parameters play a more important role in determining server performance than operational procedures.

We have shown how a video server based on hierarchical storage system can be designed given a certain start-up delay requirement, taking into account current storage technology. Indeed, a hierarchical storage system is able to provide low-cost storage. In our design examples, we show that even with uniform video popularity, we need less than 20% resident files in order to satisfy a 2-minutes delay. Non-uniform video popularity is able to reduce the storage and bandwidth requirements even more — in some cases, the reduction can be more than half.

We are currently investigating the influence of admission control policies and other staging schemes/ mechanisms. Some analytic models are also being developed, which facilitate the server design process and helps us extract performance trend. Through these studies, we are able to provide sound and comprehensive guidelines in designing an on-demand video server.

References

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

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

[3] 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.

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

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

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

[7] S.-H. G. Chan and F. A. Tobagi, "Hierarchical Storage Systems for On-Demand Video Servers," in *Proceedings of the SPIE (the International Society for Optical Engineering) High-Density Data Recording and Retrival Technologies*, (Philadelphia, PA), pp. 103–120, SPIE, October 1995.

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

[9] M. G. Kienzle, A. Dan, D. Sitaram, and W. Tetzlaff, "Using Tertiary Storage in Video-on-Demand Servers," in *Compcon: Digest of Papers*, pp. 225–232, IEEE, 1995.

[10] S. Ghandeharizadeh, A. Dashti, and C. Shahabi, "A Pipelining Mechanism to Minimize the Latency Time in Hierarchical Multimedia Storage Managers," *Computer Communication*, vol. 18, pp. 170–184, March 1995.

[11] L. Golubchik, J. C. S. Lui, and R. Muntz, "Reducing I/O Demand in Video-On-Demand Storage Servers," in *SIGMETRICS'95*, (Ottawa, Ontario, Canada), pp. 25–36, ACM, 1995.

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

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

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

[15] 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.

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

[17] 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.

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

[19] D. Patterson, G. Gibson, and R. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," in *ACM SIGMOD Conference on the Management of Data*, pp. 109–116, 1988.

[20] G. Ganger, B. Worthington, R. Hou, and Y. Patt, "Disk Arrays – High-Performance, High Reliability Storage Subsystems," *Computer Magazine*, pp. 30–36, March 1994.

[21] S. Ng, "Some Design Issues of Disk Arrays," in *Digest of Papers: Spring Compcon*, pp. 137–142, Feb 27 - Mar 3 1989.

[22] P. Chen and D. Patterson, "Maximizing Performance in a Striped Disk Array," in *The 17th IEEE Annual International Symposium on Computer Architecture*, pp. 322–331, May 28-31 1990.

[23] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "RAID: High-Performance, Reliable Secondary Storage," *ACM Computing Surveys*, vol. 26, pp. 145–185, June 1994.

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

[25] D. A. Ford, R. J. T. Morris, and A. E. Bell, "Redundant Arrays of Independent Libraries (RAIL): A Tertiary Storage System," in *COMPCON'96*, pp. 280–285, 1996.

[26] H. M. Vin, A. Goyal, and P. Goyal, "Algorithms for designing multimedia servers," *Computer Communications*, vol. 18, pp. 192–203, March 1995.

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

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

[29] S.-H. G. Chan and F. A. Tobagi, "Modeling and Analysis of Video Servers Based on Hierarchical Storage Systems." in preparation, CSL Technical Report, Stanford University, 1997.