Workload-Aware Resource Sharing and Cache Management for Scalable Video Streaming

Bashar Qudah, *Student Member, IEEE*, and Nabil J. Sarhan, *Member, IEEE*

***Abstract*—The required real-time and high-rate transfers for multimedia data severely limit the number of video streams that can be delivered concurrently. Resource-sharing techniques address this problem and can be classified into two main classes: stream merging and periodic broadcasting. We evaluate through extensive simulation major resource-sharing techniques from the two classes, considering different service models and video workloads. We utilize this extensive analysis in developing a *workload-aware hybrid solution* (WAHS) that combines the advantages of the best performers among resource-sharing techniques. Moreover, we propose a *statistical cache management* (SCM) approach and derive analytical models for optimal cache allocation to reduce further the demands on the disk I/O when various resource sharing techniques are used.**

***Index Terms*—Cache management, performance evaluation, periodic broadcasting, resource sharing, stream merging, video-on-demand (VOD), video streaming.**

I. Introduction

THE interest in scalable video streaming has increased dramatically. Unfortunately, the number of video streams that can be supported concurrently is highly constrained by the required real-time and high-rate transfers, which quickly consume server and network resources, including network bandwidth and disk I/O bandwidth. Resource-sharing techniques face this challenge by utilizing the multicast facility. The main classes of these techniques are *stream merging* [7], [9], [12], [26] and *periodic broadcasting* [14], [18], [20]–[23], [29]–[31], which deliver data in client-pull and server-push fashions, respectively. The decision as to which class and particular technique to apply greatly impacts the overall system performance and the perceived quality-of-service (QoS). With the many available techniques, it is unclear which one is the best to use in a target environment.

This paper provides a detailed analysis of resource sharing techniques, considering both the *True Video-on-Demand* (TVOD) and the *Near Video-on-Demand* (NVOD) models and two video workloads: *mixed-video* and *hot-video*. The first workload contains both hot (i.e., popular) and cold (i.e., unpopular) videos, whereas the second contains only hot videos. In TVOD, we compare the server resources required to service all requests immediately, whereas in NVOD we fix the resources and compare the number of customers that can be serviced concurrently, the waiting times, and unfairness towards unpopular videos. This study examines the impacts of many system and workload parameters and introduces and applies a framework for comparing stream merging and periodic broadcasting techniques, considering the ability of the latter to provide maximum waiting time guarantees.

Guided by this extensive analysis, this paper proposes an efficient *workload-aware hybrid solution* (WAHS) that combines the advantages of stream merging and periodic broadcasting. This solution is highly adaptive to variations in the workload and the available system resources. It first identifies the videos to be served by each technique by estimating and comparing their requirements. It then allocates resources intelligently to each technique by utilizing a proposed *prediction-based allocation* approach, which allocates resources by predicting the achieved customer defection (i.e., turn-away) probability and average waiting time by each technique. Finally, it divides the resources allocated for periodic broadcasting among the corresponding videos using a proposed *maximum-gain allocation*, which allocates channels, one at a time, to the video that gains the most from that channel in reducing its expected defection probability or waiting time. WAHS provides a promising generic solution and performs significantly better than *selective catching* [16], which is, to the best of our knowledge, the best existing hybrid scheme.

Moreover, this paper considers other specific resource-sharing aspects in terms of the disk I/O bandwidth and introduces this bandwidth as an additional dimension in comparing resource sharing techniques. Motivated by the widening gap between capacities and speeds of hard disk drives [17], cache management can be used to reduce further the cost of media streaming servers by minimizing the required disk I/O bandwidth. Unfortunately, little work has been done on cache management when scalable resource sharing is applied. The study in [5] analyzed the impact of caching by actual measurements, using limited caching schemes and simple resource sharing, and showed that the disk I/O bandwidth may become a bottleneck without proper caching. This paper proposes a *statistical cache management* (SCM) approach, which computes periodically video access frequencies and determines the data to be cached based on these statistics. In addition to its performance effectiveness, it is easy to implement and incurs small overhead as updates are triggered only when the workload varies considerably. We derive analytical models for allocating cache space for various resource sharing techniques.

Manuscript received January 05, 2008; revised May 12, 2008. First published February 13, 2009; current version published April 01, 2009. This work was supported in part by the National Science Foundation under Grant CNS-0626861. A preliminary version of a part of the paper was presented at MASCOTS 2006. This paper was recommended by Associate Editor D. Oliver.

The authors are with the Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI 48202 USA (e-mail: {bqudah,nabil}@wayne.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2009.2013498

This work differs significantly from previous work on proxy caching [24] (and references within), which has different objectives and factors affecting the allocation decisions. It also differs from low-level caching techniques, such as *least recently used* (LRU), *least frequently used* (LFU), and *interval caching* [10], [28], whereby the content of the cache changes frequently, incurring significant overheads. The remainder of this paper is organized as follows. Section II discusses major resource-sharing techniques. The proposed hybrid solution and cache management approach are presented in Sections III and IV, respectively. Section V discusses the performance evaluation methodology and presents the main results. Finally, conclusions are drawn in Section VI.

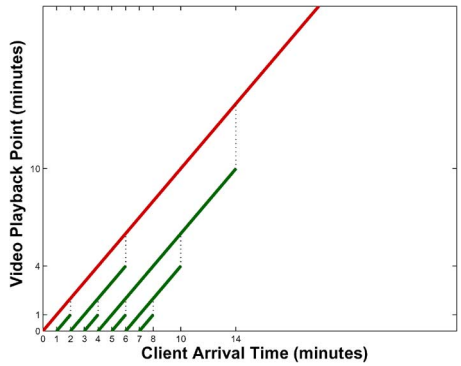

Fig. 1. Illustration of ERMT.

II. Background Information and Preliminary Analysis

A. Stream Merging Techniques

These techniques reduce the delivery costs by combining streams. They include, in increasing order of complexity and performance, *stream tapping/patching* [6], [8], *transition patching* [7], and *earliest reachable merge target (ERMT)* [12]. Each of these techniques requires two client download channels at the video playback rate. Stream tapping/patching expands the multicast tree dynamically to include new requests. A new request joins the latest *regular* (i.e., full) stream for the object and receives the missing portion as a *patch*. To avoid the continuously increasing patch lengths, regular streams are retransmitted when the required patch length exceeds a pre-specified value, called *regular window* (Wr).

Transition Patching allows some patches to be sharable by extending their lengths. It introduces another multicast stream, called *transition patch*. The threshold to start a regular stream is Wr as in patching, and the threshold to start a transition patch is called the *transition window* (Wt). The transition patch length is equal to the difference between the starting times of the transition patch and the last regular stream plus $2Wt$, whereas the length of the patch is equal to the difference between the starting times of the patch and the latest transition stream or regular stream. Hence, the maximum possible patch length is Wt, and the maximum possible transition patch length is $Wr + 2Wt$.

ERMT is a hierarchical stream merging technique, which builds a dynamic merge tree. A new client joins the closest reachable stream and receives the missing portion by a new stream. Reachable means the merge can occur before the target terminates. After the merger stream finishes and merges into the target, the latter can get extended to satisfy the playback requirement of new client(s), and this extension can affect its own merge target. For example, in Fig. 1, the third stream got extended from 2 min to 4 after the fourth stream had merged with it. ERMT performs better than other hierarchical stream merging techniques (such as those in [9], [12]) and very close to the optimal solution [4], [12].

A scheduling policy is used with stream merging to select the video to be serviced. All waiting requests for the video can be serviced using one stream. The main scheduling policies include *first come first serve* (FCFS) [11], *maximum queue length* (MQL) [11], and *maximum factored queue length* (MFQL) [1]. FCFS selects the video with the earliest request, whereas MQL selects the video with the largest number of requests, and MFQL selects the video with largest factored length (a weighted value of the queue length).

B. Periodic Broadcasting Techniques

Whereas stream merging suits a wider spectrum of video workloads, periodic broadcasting techniques can be more efficient in serving highly popular videos. These techniques divide each video into multiple segments and broadcast them periodically on dedicated channels. They include *skyscraper broadcasting* (SB) [20], *greedy disk-conserving broadcasting* (GDB) [14], *Fibonacci broadcasting* (FB) [18], and *harmonic broadcasting* (HB) [22], [25]. In SB, GDB, and FB, the segments are of variable length and the channels have equal bandwidth. The client waits until the beginning of the next broadcast of the first segment and then receives data concurrently from two broadcast channels. The relative length of the n^{th} segment compared to the first segment is determined using a technique-specific partitioning series, as shown in Table I. To illustrate the main concept, if a 30-min video is divided into three segments by FB, the length of the first segment and thus the maximum waiting time are $30/(1 + 2 + 3) = 5$ min. Note that the maximum waiting times decreases with the number of segments at the expense of increasing server bandwidth.

In contrast, HB-based protocols have uniform-length segments and the channel bandwidth decreases with the segment number. Despite their high effectiveness in reducing server bandwidth, they require the client to have download bandwidth equal to the server bandwidth allocated for the video and partition each video into a relatively large number of segments (to reduce the maximum delay).

C. Combining Stream Merging and Periodic Broadcasting

Catching uses a modified version of GDB (called here MGDB) to deliver hot videos, but, instead of making the client wait until the beginning of the next broadcast time, the client receives the small missed portion by a patch. A client initially listens to one broadcast channel and its own patch and then to two broadcast channels. The partitioning series of MGDB is shown in Table I. Selective catching extends catching by delivering cold videos using *controlled multicast* [15], which works essentially as patching. Another hybrid solution that combines batching (an older multicast scheme without stream merging) with SB was proposed in [19]. In contrast with these techniques, our hybrid solution combines the best performers

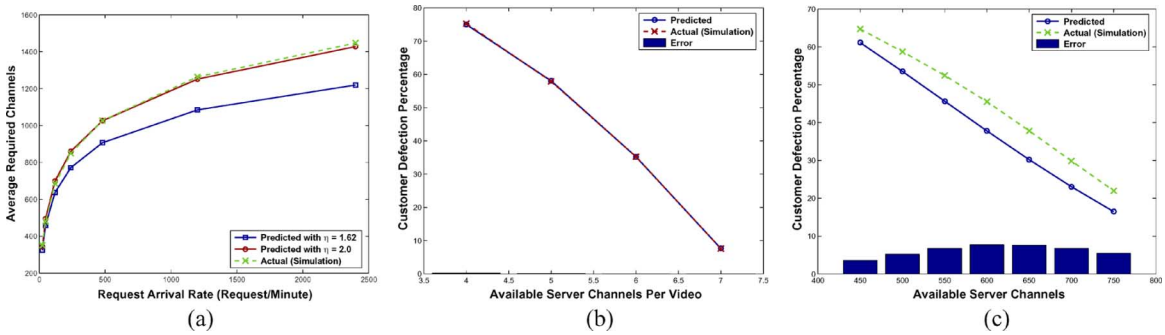

Fig. 2. Metric prediction accuracy [120 videos, video length = 120 min]. (a) Average server channel requirements by ERMT (TVOD). (b) Defection probability with FB. (c) Defection probability with ERMT (120 requests/min, FCFS).

TABLE I
SEGMENT PARTITIONING IN PERIODIC BROADCASTING

Technique	Partitioning Series
SB	$1, 2, 2, 5, 5, 12, 12, 25, 25, 52, 52, 105, 105, 212, \ldots$
FB	$1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, \ldots$
GDB	$1, 2, 2, 5, 5, 12, 12, 25, 25, 60, 60, 125, 125, 300, \ldots$
MGDB	$1, 1, 1, 2, 2, 5, 5, 12, 12, 25, 25, 60, 60, 125, 125, \ldots$

of stream merging and periodic broadcasting techniques and uses a comprehensive resource-allocation approach.

D. Analyzed Techniques

This paper analyzes patching, transition patching, ERMT, selective catching, FB, SB, and GDB. To conduct fair comparisons and account for limitations in client bandwidth, we do not consider techniques that require client download bandwidth more than double the video playback rate, such as *fast broadcasting* [23] and *enhanced fast broadcasting* [31]. To keep the discussion focused, we exclude techniques that are designed for heterogeneous receivers [3], [26], [29], [30].

III. WORKLOAD-AWARE HYBRID SOLUTION (WAHS)

We propose WAHS for scalable video streaming. The solution combines stream merging and periodic broadcasting. Without lack of generality, ERMT is used for stream merging and FB is used for periodic broadcasting. As will be shown later, each performs the best in its own class of techniques requiring only two download channels. The algorithm dynamically determines the videos to be served by each technique and allocates resources efficiently to the different techniques and also to the different videos served by periodic broadcasting. The main goals in decreasing order of importance are to minimize the customer defection probability and to minimize the average waiting time. (Customers defect without being served when their waiting time exceeds their waiting tolerance.) Table II summarizes the used symbols, and Table III shows a simplified algorithm. WAHS proceeds in the following three main steps.

A. Step 1: Identify the Videos to be Served by Each Technique

In this step, the videos served by each technique (ERMT or FB) are identified. This task translates to finding the first V_{FB} videos to be served by FB because the more popular the video is, the more likely it is to be suitable for service by FB. The

TABLE II
SYMBOLS USED IN ALGORITHM DESCRIPTION

Parameter Name(s)	Symbol(s)
Number of videos served by FB/ERMT	V_{FB}, V_{ERMT}
Number of server channels	Ch
ERMT requirement to provide TVOD for videos s till t/for the i_{th} Video	$Ch_{ERMT:TVOD_{s:t}}$, $Ch_{ERMT:TVOD_i}$
FB requirement to provide zero defections for videos s till t/for the i_{th} video	$Ch_{FB:Zero_{s:t}}$, $Ch_{FB:Zero_i}$
Channels allocated for all FB videos/for the i_{th} video	Ch_{FB}, Ch_{FB_i}
Channels shared by all ERMT videos	Ch_{ERMT}
Request arrival rate for FB/ERMT videos (req./s)	λ_{FB}, λ_{ERMT}
Defection prob. for all videos/for the i_{th} video	DP, DP_i
Defection prob. for FB/ERMT videos	DP_{FB}, DP_{ERMT}
Average waiting time for all videos/for the i_{th} video	WT, WT_i
Average waiting time for FB/ERMT videos	WT_{FB}, WT_{ERMT}

algorithm first determines whether ERMT can serve all video requests immediately without any customer defections and then whether FB can serve all the requests without defections. If the number of channels required by any one of these techniques is within server capacity, then only that technique is used. Otherwise, the algorithm compares the channel requirements to serve each video by ERMT and FB and selects for that video the technique that requires the least.

The channel requirement by ERMT to serve all videos immediately (i.e., achieve TVOD) can be found by adding up the requirement for each video: $Ch_{\mathrm{ERMT:TVOD}_{1:V}} = \sum_{i=1}^{V} Ch_{\mathrm{ERMT:TVOD}_i}$. In ERMT, all videos share the same pool of resources. However, when the number of videos is large enough, $Ch_{\mathrm{ERMT:TVOD}_i}$ can be substituted for the average server channel requirement for video i, which is derived in [13]. Thus

$$Ch_{\mathrm{ERMT:TVOD}_i} = \eta * \log(\lambda_i \times D_i/\eta) \tag{1}$$

where λ_i is the request arrival rate, D_i is the length (i.e., duration) of video i, and η is the deviation of the average number of channel requirements. η was found to be 1.62 for optimal stream merging [13], but for ERMT, we found that 2.0 is more accurate, as shown in Fig. 2(a).

TABLE III
SIMPLIFIED WAHS ALGORITHM

Step	Description
1	**Determine Number of Videos (V_{FB}) to be Served by FB:**
1.1	Estimate ERMT Channel Requirement ($Ch_{ERMT:TVOD_{1:V}}$) to provide TVOD: $V_{FB} = 0$; $\eta = 2.0$; $Ch_{ERMT:TVOD_{1:V}} = \sum_{i=1}^{V} \eta * log(N_i/\eta)$; *if* ($Ch_{ERMT:TVOD_{1:V}} < Ch$) *proceed to Step 2;*
1.2	Estimate FB Channel Requirement ($Ch_{FB:Zero_{1:V}}$) to provide zero defection; *if* ($Ch_{FB:Zero_{1:V}} < Ch$) { $V_{FB} = V$; *proceed to Step 2;* }
1.3	Estimate $Ch_{ERMT:TVOD_i}$ and $Ch_{FB:Zero_i}$ per video i *for* ($i = 1$, $i <= V$, $i++$) { *if* ($Ch_{FB:Zero_i} < Ch_{ERMT:TVOD_i}$) {$V_{FB}++$;} *else* {*break*;} }
2	**Determine Number of Channels (Ch_{FB}) for FB:**
2.1	Minimize Total Customer Defection Probability (DP): $Ch_{FB} = minimum(Ch \times \frac{\lambda_{FB}}{\lambda}, Ch_{FB:Zero_{1:V_{FB}}})$; $Ch_{ERMT} = Ch - Ch_{FB}$; $currentDP = DP$; *while* ($DP_{FB} > 0.0$ & $Ch_{ERMT} > 0$ & $DP < currentDP$) { $currentDP = DP$; $Ch_{FB}++$; $Ch_{ERMT}--$; *re-predict* DP_{FB}, DP_{ERMT}, *and* DP }
2.2	Minimize Total Customer Waiting Time (WT): $currentWT = WT$; *while* ($DP == 0.0$ & $Ch_{ERMT} > 0$ & $WT < currentWT$) { $currentWT = WT$; $Ch_{FB}++$; $Ch_{ERMT}--$; *re-predict* WT_{FB}, WT_{ERMT}, *and* WT }
3	**Distribute Broadcast Channels among FB Videos:**
3.1	Allocate minimum channels per video (*round robin from most popular to least*);

3.2	Minimize Total Customer Defection Probability (DP):
	while ($DP_{FB} > 0.0$ & $allocatedChannels < Ch_{FB}$) {
	/* *Pick the video to get the next channel* */
	$maxGain = 0.0$; $pickedVideo = 1$;
	for ($i = 1$, $i <= V_{FB}$, $i++$) {
	$gain = (DP_i(Ch_{FB_i}) - DP_i(Ch_{FB_i} + 1)) \times \lambda_i$;
	if ($gain > maxGain$) {
	$maxGain = gain$; $pickedVideo = i$; } }
	$Ch_{FB_{pickedVideo}}++$; $allocatedChannels++$; }

3.3	Minimize Total Customer Waiting Time (WT):
	while ($allocatedCh < Ch_{FB}$) {
	/* *Pick the video to get the next channel* */
	$maxGain = 0.0$; $pickedVideo = 1$;
	for ($i = 1$, $i <= V_{FB}$, $i++$) {
	$gain = (WT_i(Ch_i) - WT_i(Ch_{FB_i} + 1)) \times \lambda_i$;
	if ($gain > maxGain$) {
	$maxGain = gain$; $pickedVideo = i$; } }
	$Ch_{FB_{pickedVideo}}++$; $allocatedChannels++$; }

The channel requirement by FB is independent of the request rate. Moreover, FB has the advantage of encouraging customers to wait by providing them with highly accurate times of service. Hence, to achieve zero defections for a video, FB requires keeping the maximum waiting time (which is equal to the first segment length) for that video shorter than a certain value.

B. Step 2: Split the Resources

Once the videos to be served by each technique are determined, resources should be allocated intelligently to each technique. We propose a *prediction-based allocation approach*, which allocates resources by predicting the achieved customer defection probability and average waiting time by each technique. The algorithm first sets the number of channels for FB (Ch_{FB}) to the smaller value of the number of channels to achieve no defections and the number of channels based on its fair share in terms of the number of requests. The algorithm then tries to reduce the overall defection probability by incrementing Ch_{FB} repeatedly as long as the overall defection probability continues to decrease and remains greater than 0. Subsequently, the algorithm tries to minimize the average waiting time if the defection probability is 0 by incrementing Ch_{FB} repeatedly as long as the waiting time continues to decrease and the defection probability remains 0.

The overall defection probability (DP) and waiting time (WT) can be estimated by the weighted sums of the corresponding technique-level metrics

$$DP = DP_{\mathrm{FB}} \times \frac{\lambda_{\mathrm{FB}}}{\lambda} + DP_{\mathrm{ERMT}} \times \frac{\lambda_{\mathrm{ERMT}}}{\lambda} \quad (2)$$

$$WT = WT_{\mathrm{FB}} \times \frac{\lambda_{\mathrm{FB}}}{\lambda} + WT_{\mathrm{ERMT}} \times \frac{\lambda_{\mathrm{ERMT}}}{\lambda}. \quad (3)$$

The weighing is done based on the share of the number of requests. The technique-level metrics can be computed as the weighted sums of the corresponding per-video metrics. Hence, for FB, we have

$$DP_{\mathrm{FB}} = \sum_{i=1}^{V_{\mathrm{FB}}} DP_i \times \frac{\lambda_i}{\lambda_{\mathrm{FB}}} \quad (4)$$

$$WT_{\mathrm{FB}} = \sum_{i=1}^{V_{\mathrm{FB}}} WT_i \times \frac{\lambda_i}{\lambda_{\mathrm{FB}}}. \quad (5)$$

As discussed in Step 1, the estimations of the ERMT-level metrics are performed at the video level. Thus, DP_{ERMT} and WT_{ERMT} are computed as DP_{FB} and WT_{FB} in (4) and (5) except for using the ERMT metrics and summing over the videos from $V_{\mathrm{FB}} + 1$ and V.

1) Predicting Per-Video Metrics for FB: For FB, the defection probability for a video is a function of the waiting time, waiting tolerance model, and the number of channels allocated for that video. The first segment length (l_i), which represents the maximum waiting time for any customer requesting that video, can be easily calculated based on the number of allocated channels. Assuming equal probability for request arrival during that period, $DP_i = \int_0^{l_i} F(x).dx$, where $F(x)$ is the cumulative distribution function of the customer waiting tolerance model. This model can be built dynamically based on recent history. Step 3 determines the number of channels allocated for each one of the videos served by FB. That step is called iteratively in this step (before it is finally executed in Step 3 to obtain the actual channel allocation). The per-video average waiting time is a function of only the allocated channels. Thus, WT_i is simply $l_i/2$. Fig. 2(b) shows the estimation accuracy of the defection rate.

2) Predicting Per-Video Metrics for ERMT: The prediction here differs from that for FB in two main ways. First, because videos served by ERMT share the same pool of resources, the allocation of channels to different videos is hard to estimate. Under a fair scheduling policy, such as FCFS, each video is

expected over a long period of time to receive a fair number of channels, which is a function of its ratio of the overall number of video requests. Thus, the number of channels for video i can be estimated by

$$Ch_i = \frac{\log(\lambda_i \times D_i)}{\sum_{v=V_{\mathrm{ERMT}}}^{V} \log(\lambda_v \times D_v)} \times Ch_{\mathrm{ERMT}}. \quad (6)$$

The log represents the relation between the required channels and the request rate in case of ERMT, as shown in (1). Under a biased policy, such as MQL, hot videos can receive more than their fair share. Second, the average video inter-service time should also be estimated. The inter-service time for video i (Δ_{S_i}) is the reciprocal of that video's stream initiation rate (λ_{S_i}), which can be estimated using (1) as the maximum request arrival rate that can be served immediately by ERMT. The defection probability and waiting time can now be estimated as follows:

$$DP_i = \frac{\lambda_i - \lambda_{S_i}}{\lambda_i} \int_0^{\Delta_{S_i}} F(x).dx \quad (7)$$

$$WT_i = \frac{\lambda_i - \lambda_{S_i}}{\lambda_i} \times \frac{\Delta_{S_i}}{2}. \quad (8)$$

$F(x)$ with ERMT is likely to be different from that with FB because ERMT does not provide time of service guarantees. Multiplying by $(\lambda_i - \lambda_{S_i})/\lambda_i$ is because ERMT does not initiate streams unless they are requested. This indicates that a number of clients equal to the number of the video streams must be served and are exempted from the defection prediction process. The equations assume $\lambda_i > \lambda_{S_i}$, which should be the reason why this step is reached. As depicted in Fig. 2(c), the defection rate estimation is fairly accurate but not as accurate as that for FB due to the aforementioned complexities.

C. Step 3: Distribute the FB Resources

Instead of allocating the channels dedicated for FB uniformly to the corresponding videos or based on the fractions of request arrival rate, we propose a *maximum-gain allocation* approach, which allocates channels, one by one, to the video that gains the most from that channel in reducing its expected defection rate, weighted by its request rate. When all videos are expected to have zero defections, the allocation is based on the weighted gain in reducing the average waiting time.

IV. Statistical Cache Management (SCM)

Cache management can be applied with resource-sharing techniques to reduce further the cost of media streaming servers by minimizing the required disk I/O bandwidth. *Interval caching* [10], an efficient cache management technique, caches intervals between successive streams in the server main memory. It caches the data brought by a stream and reuses it in servicing a closely following stream. This technique incurs significant overhead and its effectiveness was not analyzed with scalable resource sharing techniques.

We propose and investigate an SCM approach. It periodically computes video access frequencies and determines the data to be cached based on these statistics. In addition to its performance effectiveness, it is easy to implement and incurs small overhead as updates are triggered only when the workload varies considerably. We first derive the access distribution equations for videos served by various techniques and then use these equations to determine the optimal cache allocation. For ease of reference, Table IV describes the main parameters used in the subsequent analysis.

TABLE IV
Main Parameters

Parameter Name	Symbol
Request arrival rate for video j (req./s)	λ_j
Inter-arrival time for video j = $1/\lambda_j$ (s)	Δ_j
Data with playback position at time t in video j	$data(j,t)$
Access Distribution: the access frequency of data d (req./s)	$Fr(d)$
Cache size (s)	S
Size of cached portion of video j (s)	S_j
Number of videos	V
Duration of video j (s)	D_j
Regular window for video j (s)	Wr_j
Transition window for video j (s)	Wt_j
Number of requests per video length for video j = $\lambda_j \times D_j$	N_j

A. Video Access Distributions

We derive next the access distribution equations for each playback point in a video delivered using various techniques. We assume here TVOD except for periodic broadcasting. Hence, the average service rate of video j is the same as its average request arrival rate (λ_j). For NVOD, the video's stream initiation rate (λ_{S_j}) can be used instead. We also assume that $Fr(data(j,t))$ is the rate (frequency) by which the tth point of time of video j is requested from the disks when there is no cache. The derived equations were validated by simulation. Additional derivation details and the validation results can be found in the technical report [27].

1) Patching: Because of having two types of streams in Patching and having a limit on the maximum patch length (Wr_j), the video can be divided into two regions: [0, Wr_j] and $[Wr_j, D_j]$. Since every regular stream delivers the full video data, all points in the second region will be requested with an equal rate: $1/Wr_j$. The situation is different for the first region because the patches vary in length. The closer the point to the beginning of the video, the more likely it will be missed and requested by patches. The access rate approaches the video request rate (λ_j) as the point becomes closer to the beginning. Consequently, the access distribution can be formulated as follows:

$$Fr\left(data(j,t)\right) = \begin{cases} \lambda_j - q\left(\frac{\lambda_j}{Wr_j} - \frac{1}{(Wr_j)^2}\right)t & 0 \le t < Wr_j, \\ \frac{1}{Wr_j} & Wr_j \le t \le D_j. \end{cases} \quad (9)$$

2) Transition Patching: The stream types here are patches with maximum length of Wt_j, transition patches with lengths between $3Wt_j$ and $Wr_j + 2Wt_j$, and regular streams with full video length. Therefore, the region $[Wr_j + 2Wt_j, D_j]$ is delivered by only regular streams. As in Patching, the rate for region [0, Wt_j] decreases linearly from λ_j to $1/Wt_j$. The region $[Wt_j, 3Wt_j]$ is delivered by every transition patch and regular stream, $[3Wt_j, 4Wt_j]$ is delivered by every regular stream and

transition patch but the first transition patch in every regular window (Wr_j), and so on. Consequently, the access distribution can be formulated as follows:

$$Fr\left(data(j,t)\right) = \begin{cases} \lambda_j - \left(\frac{\lambda_j}{Wt_j} - \frac{1}{(Wt_j)^2}\right)t, & 0 \le t < Wt_j \\ \frac{1}{Wt_j}, & Wt_j \le t < 3Wt_j \\ \frac{1}{Wt_j} - \frac{n}{Wr_j}, & 3Wt_j \le t < Wr_j + 2Wt_j \\ \frac{1}{Wr_j}, & Wr_j + 2Wt_j \le t \le D_j \end{cases} \tag{10}$$

where $n = \lfloor t/Wt_j \rfloor - 2$.

3) ERMT: By studying the average behavior of ERMT over a long period of time with N_j requests per video length, it can be noticed that almost every stream delivers independently the first part of the video from the beginning to around Δ_j, and almost $N_j - N_j/2$ streams deliver the data from approximately Δ_j to $4 \times \Delta_j$ per video length, and so on. In general, the access distribution is given by

$$Fr\left(data(j,t)\right) \approx \frac{\lambda_j}{2^{(1+\lfloor \log_2(t/\Delta_j+2)/3 \rfloor)}}. \tag{11}$$

4) Periodic Broadcasting: For each video segment, the access distribution function here is simply the inverse of the segment repeat time. For GDB, SB, and FB, the segment repeat time is the segment length.

5) Hybrid Techniques: As with periodic broadcasting, the access frequency with Catching for data in any segment other than the first is equal to the inverse of the segment length. Data in the first segment receives additional access due to the patches. The additional access frequency decreases linearly up to the last point in the first segment. Consequently, the access distribution is given by

$$Fr\left(data(j,t)\right) = \begin{cases} \lambda_j + \frac{h(K_j)}{D_j} - \lambda_j \frac{h(K_j)}{D_j}t, & 0 \le t \le \frac{D_j}{h(K_j)} \\ \frac{h(K_j)}{f(n)D_j}, & \frac{D_j}{h(K_j)} < t \le D_j \end{cases} \tag{12}$$

where $h(m) = \Sigma_{i=1}^{m} f(i)$, $f(i)$ is the ith-segment's relative size compared with the first segment in MGDB partition series, and n can be found using the inverse of function $h(m)$ as $n = h^{-1}((h(K_j)/D_j)t)$.

For selective catching, (12) can be used for hot videos, whereas (9) can be used for cold videos.

For WAHS, (11) can be used for the ERMT videos, whereas the calculations in Section IV-A4 apply for FB videos.

B. Cache Allocation Model

The following two conditions must be met for optimal cache allocation:

$$S = \sum_{j=1}^{V} S_j \qquad \forall d_{\text{in}} \in \text{cached-data}$$
$$Fr(d_{\text{in}}) \ge Fr(d_{\text{out}}) \qquad \forall d_{\text{out}} \notin \text{cached-data}. \tag{13}$$

Because the derived access distribution models exhibit the property $Fr(data(j,t_1)) \ge Fr(data(j,t_2))$ for all $t_1 < t_2$ in the jth video, if the optimal cache size for the jth video is found to be S_j, this implies that the first S_j of data from the beginning of the video must be cached. The allocation can be performed by proceeding with the video blocks (across all videos) in decreasing order of access frequency and caching them one at a time until the cache reaches its maximum capacity.

TABLE V
PARAMETER VALUES

Parameter	Mixed-Video	Hot-Video
Request arrival rate (req./min)	5 - 240	60 - 2400
Number of videos	120	1, 10, 20
Video duration (min)	120	
Video skewness (θ)	0.271	
Cache size	5% of total size of videos	
Scheduling policy	MQL	MQL, FCFS
Waiting tolerance model	Exponential, TSG	

V. PERFORMANCE EVALUATION AND MAIN RESULTS

We have developed a validated simulator for streaming servers. It uses optimal tuning of design parameters for transition patching and selective catching based on our derived analytical models provided in the technical report [27]. Simulation stops after a steady state analysis with 95% confidence interval is reached.

We consider two service models (TVOD and NVOD) and two video workloads (*mixed-video* and *hot-video*), leading to four target environments. In the TVOD model, we compare the server resources required to service all requests immediately, whereas in NVOD, we fix the server resources and compare the customer defection probability, average waiting time, and unfairness towards cold videos. The first metric translates directly to the number of customers that can be serviced concurrently. The two considered server resources are *server channels* and *disk channels*, which correspond to the network bandwidth and disk I/O bandwidth requirements, respectively. A channel can support only one stream at a time. Without caching, the numbers of these channels are equal. The mixed-video workload contains both hot and cold videos. The hot-video workload is used mainly to evaluate periodic broadcasting techniques, which are designed mainly for hot videos.

A. Workload Characteristics

We assume that the arrival of requests follows a Poisson process and that the accesses to videos follow a Zipf-like distribution with skewness parameter θ. The mixed-video workload contains 120 2-h videos, whereas in the hot-video workload, we consider three numbers of 2-h videos: 1, 10, and 20. Table V shows the values of the input parameters in both workloads. For scheduling, we use MQL and FCFS and do not show the results of MFQL because it does not perform well in the stream merging environment.

We consider two models of customer waiting tolerance. In general, we assume as in prior work that the tolerance follows the exponential distribution with a mean of 2 min. For periodic broadcasting techniques, we borrow and apply the *time-of-service guarantee* (TSG) model [2] to capture the ability of these techniques to provide time-of-service guarantees by upper limiting the waiting times. With TSG, customers who receive time-of-service guarantees shorter than 2 min will wait for service, whereas the waiting tolerance of all other customers follows the exponential distribution with a mean of 2 min.

TABLE VI
PERCENTAGE OF CUSTOMERS RECEIVING TIME-OF-SERVICE GUARANTEES WITH WAHS [720 CHANNELS, MQL]

Request Arrival Rate (Requests/Minute)	Received TSG (Percentage)	No TSG (Percentage)
90	63%	37%
120	74%	26%
150	83%	17%
180	91%	9%

To study the effectiveness of WAHS in handling dynamic variations in the request arrival rate and video popularity, we use the following dynamic workload. The request rate has a 24-h cycle with a maximum (peak) and a minimum period, separated by 12 h, and the minimum request rate being half of the maximum. The rate changes gradually in 3-h steps by dividing/multiplying by 2 1/4. The video popularity has a daily pattern and a day-to-day pattern. Within the day, when the system goes from the maximum to the minimum rate, each video popularity drops one rank every 90 min and the least popular becomes the most popular, and when going from the minimum to the maximum rate, each video gains a rank every 180 min and the most popular becomes the least popular. Hence, most videos loose four ranks from a day to the next. This workload evaluates WAHS in comparison with ERMT and FB when significant changes in the request rate and video popularity happen in short periods of time. It is synthetic and meant to be much worse than that expected in reality, where changes are more gradual and do not happen to all videos simultaneously.

B. Analysis of Existing Techniques

1) Environment I: Mixed-Video and TVOD: This environment helps in comparing stream merging and composite techniques in terms of the number of server and disk channels required to achieve TVOD (i.e., zero defection and waiting time). We vary here the request arrival rate from 5 to 100 requests per minute and examine the average and the maximum server channels required by the different techniques. The average results represent the average requirements over long periods, while the maximum results represent the actual required number of channels for providing TVOD for all requests. Fig. 3 plots these results. As expected, the gaps among the techniques widen as the arrival rate increases because the workload offers increasingly higher degrees of resource sharing, which are exploited to different levels by different techniques. Generally, ERMT has the lowest requirements, and Patching the highest. Interestingly, selective catching does not perform relatively well. We have experimented with using ERMT instead of patching in selective catching for serving cold videos but the performance benefits are low.

2) Environment II: Hot-Video and TVOD: In this environment, we compare the performance of various periodic broadcasting techniques (SB, GDB, and FB) and the best performer in Environment I (ERMT). Since periodic broadcasting cannot provide a zero waiting time, we assume that a maximum waiting time below 2 s constitutes a TVOD service.

Fig. 4(a) plots the average and maximum required server channels per video to provide TVOD by the different techniques. Unlike periodic broadcasting, the number of server

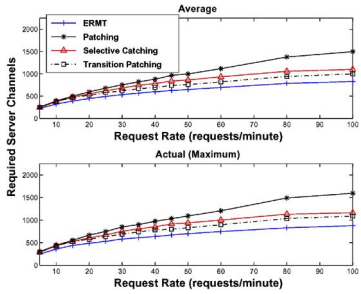

Fig. 3. Environment I: mixed-video and TVOD. Effect of request rate on required channels.

channels per video with ERMT depends on the arrival rate and the number of supported videos. Whereas periodic broadcasting techniques reserve channels for each video, ERMT deals with all the resources as one pool and distributes them dynamically to the requests. While the average helps in understanding the overall system load, the maximum indicates the actual bandwidth to be reserved to achieve TVOD. The distinction between the two metrics was always ignored in previous studies, where only the average results were usually reported. The results demonstrate that ERMT scales very well with the increase in request rate even in terms of both the average and actual (maximum) required bandwidth. For a workload of 20 2-h videos, the request rate must be constantly over 900 requests/minute for the best periodic broadcasting technique (FB) to be a better choice than ERMT. Even when that is the case, while ERMT requires more guaranteed bandwidth, it does not consume it all the time, which may enable the system to face short periods of drops in server capacity. Since the required bandwidth with ERMT is a function of the number of requests per video length (i.e., $N = \lambda D$), the breakpoint is higher than 900 requests/min when the videos are shorter.

To highlight the effect of the difference between the average and actual required number of channels, Fig. 4(b) plots the average and maximum waiting times if only the average required bandwidth is provided. Therefore, the number of provided channels varies with the request rate and with the number of videos as well. The service is obviously not TVOD, especially for the relatively low request rates because some customers had to wait few minutes for service. Since scheduling policies can impact the performance of ERMT compared with periodic broadcasting, both MQL and FCFS are examined. MQL achieves higher throughput and shorter average waiting time, whereas FCFS is fairer and can significantly reduce the maximum waiting time.

3) Environment III: Mixed-Video and NVOD: By varying the number of server channels, we can use this environment to compare stream merging and composite techniques in terms of the achieved average customer defection probability, waiting time, and unfairness. Fig. 5 plots these three metrics versus the number of server channels for various techniques. The results here demonstrate once again that ERMT performs the best and Patching the worst among stream merging and composite techniques. There is a clear correlation among the three metrics. The correlation between unfairness and the other two metrics is due to using MQL. MQL selects for service the longest video queue, trading off fairness for performance. The more limited

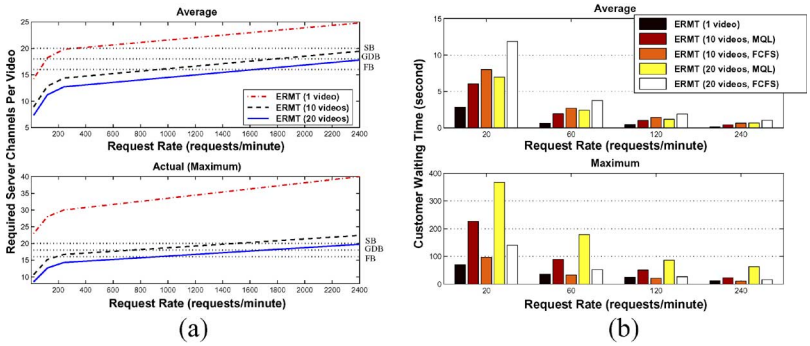

Fig. 4. Environment II: hot-video and TVOD. (a) Wffect of request rate on required channels. (b) Effect of request rate on waiting time (average required channels provided).

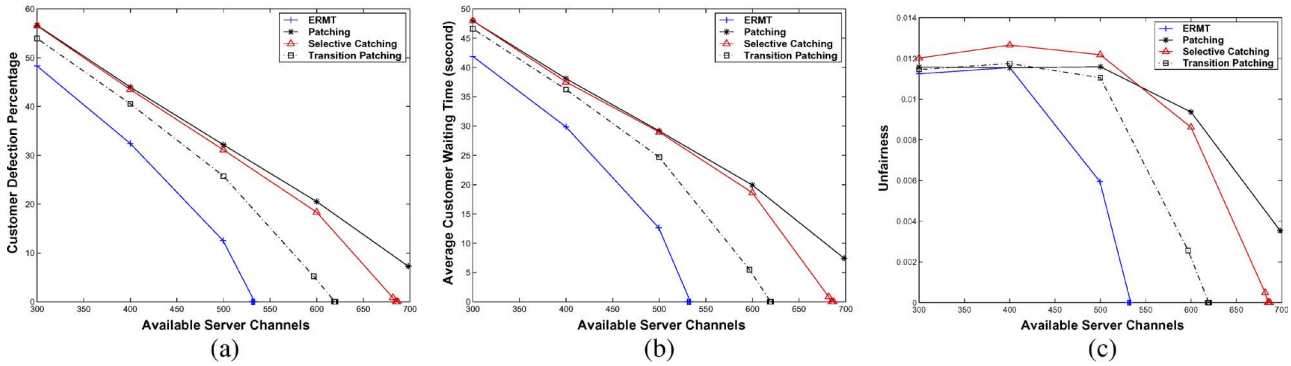

Fig. 5. Environment III: mixed-video and NVOD [30 requests/minute]. (a) Effect of server channels on defection probability. (b) Effect of server channels on average waiting time. (c) Effect of server channels on unfairness.

the number of channels becomes, the longer customers have to wait, and the faster the waiting queues build up. Consequently, both the defection probability and the bias against cold videos increase.

4) Environment IV: Hot-Video and NVOD: In this environment, we also compare the performance of various periodic broadcasting techniques with ERMT, considering two waiting tolerance models: exponential and TSG. Fig. 6(a)–(d) depicts the defection probability and average waiting time under the exponential model for different server capacities, request rates, and numbers of videos. These results confirm that ERMT is very competitive, compared with the best periodic broadcasting technique, even in servicing very hot videos and under very limited resources.

Fig. 6(e) and (f) compare various techniques in terms of the defection probability and average waiting time under the TSG model, which captures the ability of the broadcasting techniques in providing waiting time guarantees. The results demonstrate that broadcasting techniques can achieve much lower defection probabilities than ERMT, especially when the available server bandwidth is not very limited. With only 8 server channels per video, all broadcasting techniques cause no defections, whereas ERMT causes more than 20% defections. However, ERMT achieves shorter average waiting time for those clients who were serviced, partially due to the high defection percentage.

C. Effectiveness of the Proposed Hybrid Solution

Under the mixed-video workload, Fig. 7 demonstrates that WAHS outperforms both ERMT and FB in throughput, the most important metric, and its performance in the average waiting time lies between ERMT and FB. In addition, WAHS provides the majority of customers with time-of-service guarantees, whereas ERMT does not. As expected, FB is the best in fairness (the least important metric). Selective catching performs poorly and thus its results are only shown for 720 channels. Its relative performance is worse with 840 channels.

Let us now discuss the impact of dynamic workload. Fig. 8 explains how the per-video channel allocation transition is handled when the number of allocated channels for a video is changed due to workload changes. The example shows a switch from six-channel allocation to five-channel for a *2-h* video. The required five channels in the new allocation will be required all the time, during the transition period and afterward, but an additional channel is needed for short periods during the transition. (The additional number of needed channels is the difference between the two allocations. These channels are required to change the allocation without impacting existing clients.) Although each additional channel can be used for several video transitions, to be conservative, a maximum of two transitions were allowed to share such a channel. Under the dynamic workload, Fig. 9 demonstrates that WAHS still keeps its lead in the defection rate.

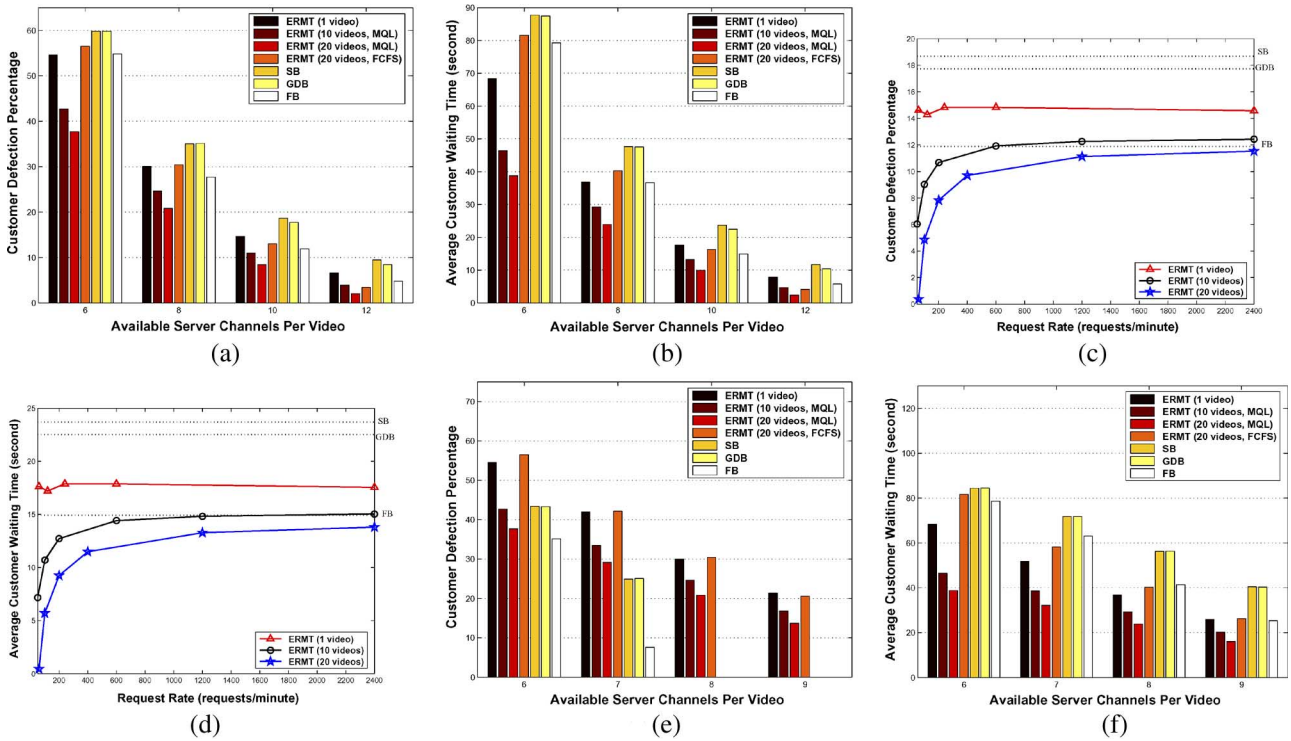

Fig. 6. Environment IV: hot-video and NVOD [defaults are 240 requests/minute, 10 channels per video, MQL]. (a) Effect of server channels per video on defection probability (exponential). (b) Effect of server channels per video on average waiting time (exponential). (c) Effect of request rate on defection probability (exponential). (d) Effect of request rate on average waiting time (exponential). (e) Effect of server channels per video on defection probability (TSG). (f) Effect of server channels per video on average waiting time (TSG).

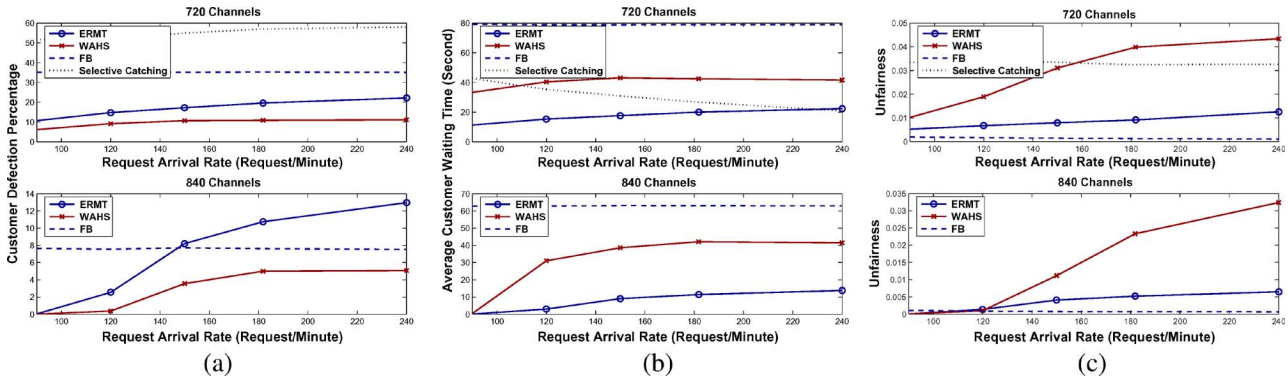

Fig. 7. WAHS compared with ERMT and FB [MQL, TSG]. (a) Defection. (b) Waiting. (c) Unfairness.

D. *Effectiveness of the Proposed Cache Management Scheme*

Finally, let us evaluate the effectiveness of SCM. Fig. 10 compares SCM and interval caching in terms of the reduction in the average and maximum I/O requirements when using WAHS, ERMT, and FB. The results show that SCM performs significantly better than interval caching. With a cache size of 2% of the total video size, the reductions achieved by SCM are up to 18%. The reductions in the maximum I/O requirements in the case of ERMT are exceptionally low (but still much better than interval caching) because of the dynamic nature of ERMT and the irregularity in resource consumption and allocation among videos. Interval caching causes no gain with FB because it exploits the data overlap between streams, and no two streams that coexist in FB carry the same data.

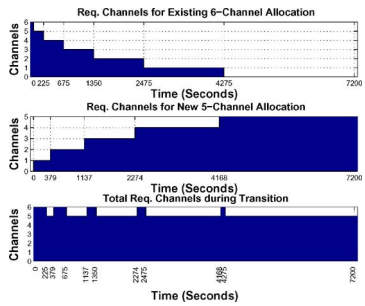

Fig. 8. Illustration of support for per-video channel allocation with FB (six to five channels).

Fig. 11 compares the effectiveness of SCM with other resource-sharing techniques, including patching, transition

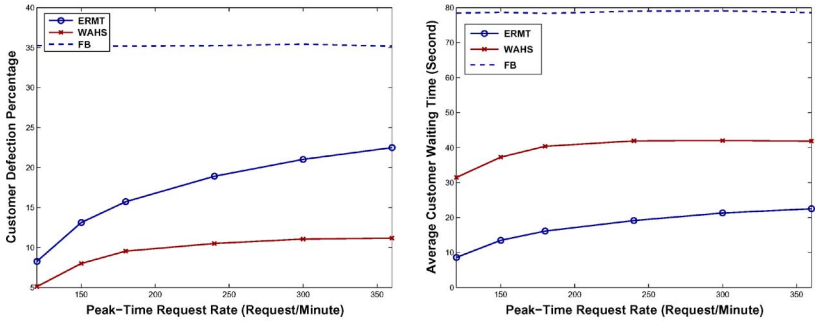

Fig. 9. WAHS compared to ERMT and FB under dynamic workload [720 channels, MQL, TSG]. (a) Defection. (b) Waiting.

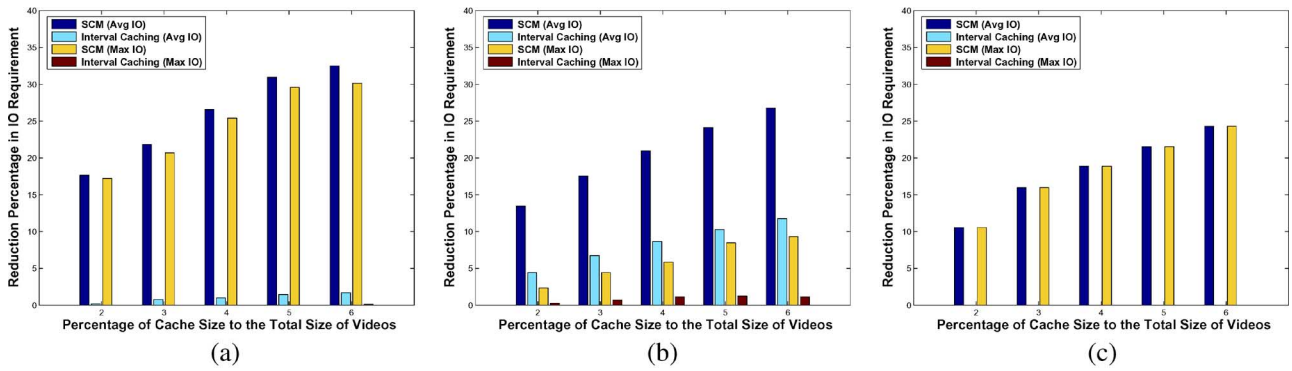

Fig. 10. Effectiveness of SCM versus interval caching [720 channels, 150 requests/minute, MQL, TSG]. (a) WAHS. (b) ERMT. (c) FB.

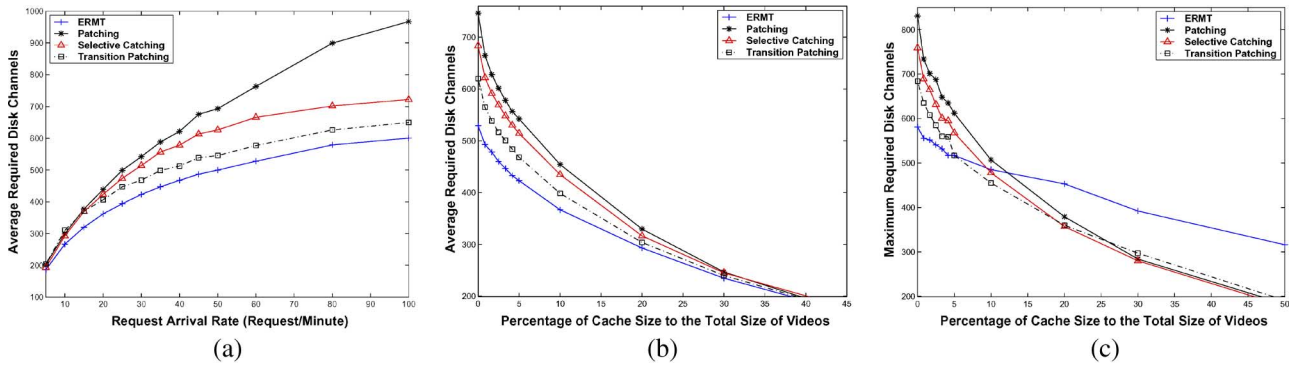

Fig. 11. Impact of caching [environment I, 30 requests/minute, SCM]. (a) Effect of request rate on average disk channels. (b) Effect of cache size on average disk channels. (c) Effect of cache size on maximum disk channels].

patching, and selective catching. ERMT requires the lowest average disk I/O channels, and patching the highest. However, as expected, the effectiveness of caching decreases, and the gaps among various techniques diminish as we keep increasing the cache size. Interestingly, whereas ERMT requires the smallest actual number of disk channels with no cache, the situation changes when the cache is introduced as shown in Fig. 11(c). With a cache size greater than 5% of the total size of videos, transition patching starts requiring fewer disk channels than ERMT. With a cache size greater than 12%, all of the other techniques require fewer disk channels than ERMT. ERMT benefits the least from caching and patching the most because of the access distribution curves in Section IV-A.

VI. Conclusion

The main results can be summarized as follows.

1) In workloads containing both hot and cold videos, ERMT generally performs the best among existing techniques.
2) In workloads containing only hot videos, FB performs better than both SB and GDB in terms of the waiting times. For NVOD, FB achieves higher throughput than ERMT, assuming customers are encouraged to wait when informed with their times of service. ERMT, however, achieves shorter waiting times. It is also more capable of providing TVOD.
3) WAHS outperforms both ERMT and FB in server throughput. It can eliminate more than half the defec-

tions compared to the best alternative among ERMT and FB, and more than 70% compared to the other. WAHS also outperforms FB in the average customer waiting time. ERMT can lead to shorter waiting times, but it does not provide TSGs while WAHS provides these guarantees to the majority of customers.

4) SCM is very effective in further reducing the disk I/O bandwidth requirements. The reductions can be up to 18% and 30%, when the cache sizes are 2% and 6% of the total size of videos, respectively.

Consequently, WAHS is the best choice under a wide range of workloads and server resources. SCM is highly recommended with WAHS to reduce further the disk I/O requirements.

REFERENCES

[1] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "The maximum factor queue length batching scheme for Video-on-Demand systems," *IEEE Trans. Comput.*, vol. 50, no. 2, pp. 97–110, Feb. 2001.

[2] M. Alsmirat, M. Al-Hadrusi, and N. J. Sarhan, "Analysis of waiting-time predictability in scalable media streaming," in *Proc. ACM Multimedia*, Sep. 2007, pp. 791–794.

[3] O. Bagouet, K. A. Hua, and D. Oger, "A periodic broadcast protocol for heterogeneous receivers," in *Proc. Multimedia Computing Networking Conf. (MMCN)*, Jan. 2003, vol. 5019, pp. 220–221.

[4] A. Bar-Noy, G. Goshi, R. Ladner, and K. Tam, "Comparison of stream merging algorithms for media-on-demand," *Multimedia Syst. J.*, vol. 9, pp. 211–223, 2004.

[5] M. Bradshaw, B. Wang, S. Sen, L. Gao, J. Kurose, P. Shenoy, and D. Towsley, "Periodic broadcast and patching services: Implementation, measurement and analysis in an internet streaming media testbed," in *Proc. ACM Multimedia*, Oct. 2001, pp. 280–290.

[6] Y. Cai and K. A. Hua, "Sharing multicast videos using patching streams," *Multimedia Tools Appl. J.*, vol. 21, no. 2, pp. 125–146, Nov. 2003.

[7] Y. Cai, W. Tavanapong, and K. A. Hua, "Enhancing patching performance through double patching," in *Proc. 9th Int. Conf. Distrib. Multimedia Syst.*, Sep. 2003, pp. 72–77.

[8] S. W. Carter and D. D. E. Long, "Improving Video-on-Demand server efficiency through stream tapping," in *Proc. Int. Conf. Comput. Commun. Networks (ICCCN)*, Sep. 1997, pp. 200–207.

[9] E. Coffman, P. Jelenkovic, and P. Momcilovic, "The dyadic stream merging algorithm," *J. Algorithms*, vol. 43, no. 18, pp. 120–137, Apr. 2002.

[10] A. Dan, D. M. Dias, R. Mukherjee, D. Sitaram, and R. Tewari, "Buffering and caching in large-scale video servers," in *Proc. IEEE Int. Comput. Conf.*, Mar. 1995, pp. 217–225.

[11] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling policies for an on-demand video server with batching," in *Proc. of ACM Multimedia*, Oct. 1994, pp. 391–398.

[12] D. L. Eager, M. K. Vernon, and J. Zahorjan, "Optimal and efficient merging schedules for Video-on-Demand servers," in *Proc. ACM Multimedia*, Oct. 1999, pp. 199–202.

[13] D. L. Eager, M. K. Vernon, and J. Zahorjan, "Bandwidth skimming: A technique for cost-effective Video-on-Demand," in *Proc. Multimedia Computing and Networking Conf. (MMCN)*, Jan. 2000, pp. 206–215.

[14] L. Gao, J. Kurose, and D. Towsley, "Efficient schemes for broadcasting popular videos," in *Proc. Int. Workshop Network and Operating Syst. Support for Digital Audio and Video (NOSSDAV)*, Jul. 1998, pp. 317–329.

[15] L. Gao and D. Towsley, "Supplying instantaneous Video-on-Demand services using controlled multicast," in *Proc. IEEE Multimedia Computing Syst.*, Jun. 1999, pp. 117–121.

[16] L. Gao, Z.-L. Zhang, and D. F. Towsley, "Catching and selective catching: Efficient latency reduction techniques for delivering continuous multimedia streams," in *Proc. ACM Multimedia*, Oct. 1999, pp. 203–206.

[17] J. Gray and P. Shenoy, "Rules of thumb in data engineering," in *Proc. IEEE Int. Conf. Data Eng.*, Feb. 2000, pp. 3–10.

[18] A. Hu, "Video-on-Demand broadcasting protocols: A comprehensive study," in *Proc. IEEE INFOCOM*, Apr. 2001, pp. 508–517.

[19] K. A. Hua, J. Oh, and K. Vu, "An adaptive video multicast scheme for varying workloads," *Multimedia Syst.*, vol. 8, no. 4, pp. 258–269, 2002.

[20] K. A. Hua and S. Sheu, "Skyscraper broadcasting: A new broadcasting scheme for metropolitan Video-on-Demand system," in *Proc. ACM SIGCOMM*, Sep. 1997, pp. 89–100.

[21] C. Huang, R. Janakiraman, and L. Xu, "Loss-resilient on-demand media streaming using priority encoding," in *Proc.f ACM Multimedia*, Oct. 2004, pp. 152–159.

[22] L. Juhn and L. Tseng, "Harmonic broadcasting for Video-on-Demand service," *IEEE Trans. Broadcasting*, vol. 43, no. 3, pp. 268–271, Sep. 1997.

[23] L. Juhn and L. Tseng, "Fast data broadcasting and receiving scheme for popular video service," *IEEE Trans. Broadcasting*, vol. 44, no. 1, pp. 100–105, Mar. 1998.

[24] J. Liu and J. Xu, "Proxy caching for media streaming over the internet," *IEEE Commun. Mag.*, vol. 42, no. 8, pp. 88–94, Aug. 2004.

[25] J.-F. Pâris, S. W. Carter, and D. D. E. Long, "Efficient broadcasting protocols for video on demand," in *Proc. Int. Symp. Modeling, Anal. Simulation Comput. Telecommun. Syst. (MASCOTS)*, Jul. 1998, pp. 127–132.

[26] B. Qudah and N. J. Sarhan, "Towards scalable delivery of video streams to heterogeneous receivers," in *Proc. ACM Multimedia*, Oct. 2006, pp. 347–356.

[27] B. Qudah and N. J. Sarhan, Workload-Aware Resource Sharing and Cache Management for Scalable Video Streaming Dept. Elect. Comput. Eng., Wayne State Univ., Detroit, MI, Oct. 2007, Tech. Rep..

[28] N. J. Sarhan and C. R. Das, "Caching and scheduling in NAD-based multimedia servers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 10, pp. 921–933, Oct. 2004.

[29] P. Sessini, L. Shi, A. Mahanti, Z. Li, and D. L. Eager, "Scalable streaming for heterogeneous clients," in *Proc. ACM Multimedia*, Oct. 2006, pp. 337–346.

[30] M. A. Tantaoui, K. A. Hua, and T. T. Do, "Broadcatch: A periodic broadcast technique for heterogeneous Video-on-Demand," *IEEE Trans. Broadcasting*, vol. 50, no. 3, pp. 289–301, Sep. 2004.

[31] Y.-C. Tseng, C.-M. Hsieh, M.-H. Yang, W.-H. Liao, and J.-P. Sheu, "Data broadcasting and seamless channel transition for highly-demanded videos," in *Proc. IEEE INFOCOM*, Mar. 2000, pp. 727–736.

Bashar Qudah (S'08) received the B.S. degree in electrical engineering from the University of Jordan in 1997 and the M.S. degree in computer engineering from the University of Michigan, Ann Arbor, in 2003. He is currently working toward the Ph.D. degree at the Electrical and Computer Engineering Department, Wayne State University, Detroit, MI.

His research areas include multimedia computing and networking, video streaming, parallel computing, performance evaluation, and software engineering. He has more than ten years of professional experience in software development.

Nabil J. Sarhan (M'04) received the B.S. degree in electrical engineering from Jordan University of Science and Technology and the M.S. and Ph.D. degrees in computer science and engineering from the Pennsylvania State University, University Park.

In 2003, he joined Wayne State University, Detroit, MI, where he is an Assistant Professor with the Department of Electrical and Computer Engineering and directs the Wayne State Media Research Laboratory. His main research areas of interest include multimedia computing and networking, video streaming, storage, computer architecture, and performance evaluation. He has a strong publication record in top conferences and journals and has served as a technical program committee member of premier international conferences and as a panelist for the National Science Foundation and the National Institutes of Health.

Dr. Sarhan was the recipient of the 2008 Outstanding Professional of the Year Award from the IEEE Southeastern Michigan Section for many accomplishments in multimedia research and university teaching profession.