

**A Statistical Admission Control Scheme for Continuous Media
Servers Using Caching**

Jin B. Kwon and Heon Y. Yeom
School of Computer Science and Engineering
Seoul National University
Seoul, 151-742, Korea

* TO APPEAR IN "*MULTIMEDIA TOOLS AND APPLICATIONS*" IN 2003.

A Statistical Admission Control Scheme for Continuous Media Servers Using Caching

Jin B. Kwon Heon Y. Yeom

School of Computer Science and Engineering

Seoul National University, Korea

{jbkwon,yeom}@dclab.snu.ac.kr

Abstract

In continuous media servers, disk load can be reduced by using buffer cache. In order to utilize the saved disk bandwidth by caching, a continuous media server must employ an admission control scheme to decide whether a new client can be admitted for service without violating the requirements of clients already being serviced. A scheme providing deterministic QoS guarantees in servers using caching has already been proposed. Since, however, deterministic admission control is based on the worst case assumption, it causes the wastage of the system resources. If we can exactly predict the future available disk bandwidth, both high disk utilization and hiccup-free service are achievable. However, as the caching effect is not analytically determined, it is difficult to predict the disk load without substantial computation overhead. In this paper, we propose a statistical admission control scheme for continuous media servers where caching is used to reduce disk load. This scheme improves disk utilization and allows more streams to be serviced while maintaining near-deterministic service. The scheme, called *Shortsighted Prediction Admission Control* (SPAC), combines exact prediction through on-line simulation and statistical estimation using a probabilistic model of future disk load in order to reduce computation overhead. It thereby exploits the variation in disk load induced by VBR-encoded objects and the decrease in client load by caching. Through trace-driven simulations, it is demonstrated that the scheme provides near-deterministic QoS and keeps disk utilization high.

1 Introduction

Recent advances in computing and communication technologies have made it feasible as well as economically viable to provide on-line access to continuous media objects such as video or audio over high speed networks. The fundamental problem in developing continuous media servers is that audio and video differ from image and text in their characteristics, and hence require different techniques for their organization and management. *Continuous Media*(CM) objects such as audio and video objects which consist of sequences of media quanta(audio samples or video frames), convey meaning only when presented continuously in time. In order to ensure continuous delivery of video or audio, various resources, such as disk bandwidth, memory, network bandwidth and so on, should be reserved before commencing playback of a stream.

Read-ahead buffering, where blocks are read and buffered ahead of the time they are delivered to the network, has been introduced[6]. Since CM objects usually require high disk bandwidth, it is important to reduce disk I/O in continuous media servers to service the CM objects. Storage systems can reduce the number of disk I/O operations by sharing the data already retrieved from disk among all of the clients, using buffer cache[13]. For the purpose of reducing disk I/O, recent works[7, 11] have proposed the use of a global buffer cache similar to the buffer cache in traditional storage systems. Owing to the access pattern for CM objects, however, LRU and MRU, which are regarded as good replacement algorithms, do not yield a high cache hit ratio in continuous media servers[2]. A stream, a client of a continuous media server, reads sequentially from the first block to the last block of CM objects in time. Thus, continuous media servers can determine when and which block should be read. Several caching schemes that consider the access pattern have been presented[3, 12]. As they focus only on reducing disk I/O by employing a high hit ratio, they cannot provide *quality-of-service*(QoS) guarantees which CM servers must provide to users. It is evident that the higher hit ratio is, the lower average disk load is. Though the low average disk load lowers the probability of disk overload, it never means that it provides QoS guarantees. Note that the hit ratio alone cannot capture the effectiveness of a caching scheme for such servers without QoS guarantees.

Therefore, the CM servers using caching need to have an admission control that enables them to utilize the disk bandwidth saved by caching with QoS guarantees.

CM servers organize the storage of CM objects on disk in terms of *media blocks*, and they service multiple clients simultaneously by proceeding in rounds. If the total time spent in retrieving media blocks during a round exceeds the *round length*, at least one client experiences *hiccup* due to disk overload. In order to provide QoS guarantees for the clients, CM servers employ an *admission control scheme* to decide whether a new client can be admitted for service without violating the requirements of the clients being serviced. So, it is critical that admission control schemes have information on the available amount of resources and the requirements of a new client. Admission control schemes can be classified into *deterministic schemes* and *statistical schemes* according to the way the CM server handles the QoS requirements of the clients[14]. Since deterministic admission control schemes are characterized by worst-case assumptions which needlessly constrain the number of clients that can be serviced simultaneously, they lead to severe under-utilization of server resources. Statistical schemes provide a probabilistic QoS guarantee instead of deterministic one, resulting in higher resource utilization due to statistical multiplexing gain[4, 14]. An admission control scheme in CM servers using caching must consider the caching effect in order to utilize the saved disk bandwidth by caching. If we can exactly predict the future available disk bandwidth instead of assuming the worst case situation, both high disk utilization and hiccup-free service(deterministic QoS) are achievable. Though CM servers can determine the access patterns of the streams currently being serviced, they cannot do when and which objects new clients would request in the future. Moreover, the caching effect is variable while proceeding in rounds, i.e., *non-deterministic*. Thus, the exact future disk load is hard to predict exactly without substantial computation overhead.

Although admission control schemes in servers that do not use caching have been developed[1, 5, 15, 16], there has been little work on how to effectively utilize the saved disk bandwidth due to cache hits. In this paper, we propose a statistical admission control scheme for continuous media servers which employs *round-level caching*, called *Shortsighted Prediction Admission Control*(SPAC). Our scheme integrates exact prediction through on-line simulation and prob-

abilistic prediction through the statistical estimation of the future disk load so that it provides near-deterministic QoS with tolerable computation overhead. SPAC determines whether admitting a new stream(client) causes disk overload through on-line simulation of disk activity from the starting point of the new stream to the point at which any of the existing streams ceases. Then the possible disk overload that can occur after the end of the simulation is tested statistically. Our probabilistic model is shown to match the workload very satisfactorily.

The remainder of this paper is organized as follows. Section 2 briefly introduces related studies and section 3 describes our system model including object striping, round-level caching and assumptions on the disk load. We propose a statistical admission control scheme, SPAC, in section 4 and the experimental results are shown in section 5. Finally we present our conclusions and introduce future work in section 6.

2 Related Work

Dan et al. presented a caching scheme for video-on-demand servers named *Interval Caching*[3]. It exploits temporal locality of the requests accessing the same CM object by caching *intervals* that comprise all the data blocks between successive requests for the same object. Two streams, S_i and S_j are defined to be *consecutive* if S_j is the stream that next reads the data blocks that have just been read by S_i . S_i and S_j are referred to as the *preceding* stream and the *following* stream, respectively. When an interval is cached, its following stream does not need any disk I/O because all the needed data are in the buffer cache. Since caching granules in *Interval Caching* are large, it only requires a small computational overhead. However, it results in a somewhat low hit ratio. Özden et al. proposed another caching scheme, BASIC, whose caching granules are blocks[12]. Whenever there is a need to remove a block from the cache, it evicts the block not to be referenced for the longest duration by the clients being serviced, based on the nearest re-reference time(*future*). Although it yields a high hit ratio, it introduces a great deal of computation caused by computation of the *future* values of all the blocks in buffer cache in each round. In order to decrease the computational overhead of BASIC, another scheme DISTANCE was proposed. In DISTANCE, caching granules are

intervals rather than blocks as in *Interval Caching*. It selects the longest interval as the victim. Owing to the granularity of caching, the hit ratio of DISTANCE is lower than that of BASIC.

However, all the works mentioned concentrate only on increasing the hit ratio without considering how to effectively use the reduction in the disk load by caching. They did not provide any means to service more clients with QoS guarantees using the saved disk bandwidth. *Preemptive but Safe Interval Caching*(PSIC) [9] based on *Interval Caching* proposes a method of utilizing the caching gain by employing deterministic admission control for *hiccup-free* services. The victim for replacement is a stream with the largest interval among all the cached intervals whose following streams can access disks without violating the streams being serviced. The high performance of PSIC was demonstrated through simulation on CBR objects, but the performance on VBR objects was not dealt with. In the case of VBR objects, since the difference between the worst case and average case is considerable, the performance may be degraded by low resource utilization. Therefore, we propose a statistical admission control scheme that provides near-deterministic QoS to all clients on VBR objects in a CM server using caching.

3 System Model

In this section we describe the system model on which SPAC is based. Though there are various system resources, we focus on the disk bandwidth which is the main bottleneck.

3.1 Striping CM Objects

CM servers usually organize the storage of CM objects on disks in terms of *media blocks*. We divide each CM object into media blocks whose playback durations are the round length \mathcal{B} . These media blocks are striped along the disks. It is called *Constant Time Length*(CTL) striping. In the case of VBR-encoded objects, the playback durations of media blocks are the same and their sizes are variable. The servers service multiple clients simultaneously by proceeding in *rounds*. If k active streams are being serviced from disk i at the current round,

then these streams will be serviced from disk $(i + 1) \bmod n$ at the next round, where n is the number of disks. We define a *service group* as the streams serviced by a disk. Then, there always exist n service groups. Since we assume that there is no VCR-operations, there is no movement among the service groups. Accordingly, a service group can think of as a disk, since a service group is not changed except admission and departure of streams. Therefore, though we assume a single homogeneous disk for simplicity of presentation, it is easily applied to the case of having multiple disks.

3.2 Round-level Caching

A block(or page), which is the caching unit in a general-purpose file system, is the smallest unit of caching in block-based disk I/O. As mentioned in section 1, since streams read CM objects sequentially in time, CM servers can use TTNR(Time To Next Reference) of blocks in cache to replace a block. Thus if a replacement algorithm selects a block to have the largest TTNR as the victim for replacement, it can achieve an optimal hit ratio. For this purpose, it is necessary to sort each block in each round according to the TTNR values of blocks. Since buffer cache usually contains tens or hundreds of thousands of blocks(e.g. a 125MB-sized cache contains 122,000 1KB blocks), the computational overhead for sorting the blocks is considerable.

Figure 1 shows a snapshot of a CM server where the numbers on each object indicate the TTNRs of media blocks to be retrieved by streams at a time. Since we assume there is no VCR-operation in our system models, the TTNRs of the media blocks that belong to intervals are deterministic or predictable. In the figure the media block consumed by S_1 in round t is always consumed by S_2 in round $t + 2$, i.e., its TTNR is always 2 because the length of the interval between the preceding stream S_1 and the following stream S_2 is 2. A caching scheme where a media block is the caching unit is defined as a *round-level caching* scheme. In CM servers, since the unit of TTNR is a round and each stream reads a media block in a round, a TTNR value is assigned to each media block in cache. Hence the servers can substantially decrease the computation for selecting victims. Although enlarging the granularity of caching may bring about *external fragmentation*, the decrease in hit ratio which may result is not

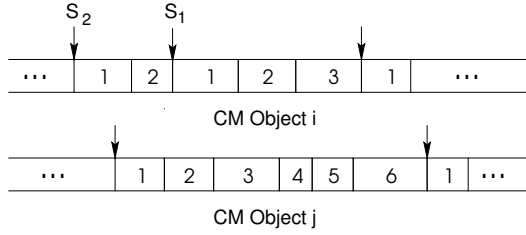


Figure 1: Snapshot of a CM server

significant. There is no decrease in hit ratio originating from the difference on the caching unit since physical blocks in a media block have the same TTNR. Therefore, we use round-level caching as the basis of our admission control scheme.

3.3 Characteristics of the Disk Load

Consider a CM server that is servicing n clients (say S_1, S_2, \dots, S_n , respectively). The CM objects it services are encoded using the MPEG compression technique resulting in a VBR stream. Let a random variable X_i be the number of blocks that a stream S_i requires in a round, and μ_i and σ_i^2 denote the mean and the variance of X_i , respectively. When the CM server does not use caching, X_i is $N(\mu_i, \sigma_i^2)$. That is, X_i is normally distributed with mean μ_i and variance σ_i^2 . Thus, based on the Central Limit Theorem, the sum of the number of blocks that each stream should read from disk in a round follows the normal distribution[10]:

$$X \sim \mathcal{N}(\mu, \sigma^2), \text{ where } X = \sum_{i=1}^n X_i, \mu = \sum_{i=1}^n \mu_i, \sigma^2 = \sum_{i=1}^n \sigma_i^2,$$

μ_i 's and σ_i^2 's are calculated from *priori* information on the media block sizes within n objects.

Another important factor to be considered for modeling the disk load is the seek overhead which depends on the number of disk seeks. Since CM servers retrieve data from disks by media block, we assume that a media block is placed contiguously on the disk. This placement eliminates the disk seeks within a media block and only induces disk seeks between media blocks. Hence, we can model the seek overhead easily using the number of active streams which is equal to the number of media blocks retrieved in a round. Vin et al. proposed a disk load model based on an empirical value[14], but the maximum overhead assumption is

needed to model the disk load for deterministic admission control. However, since the disk overhead distribution strongly depends on the data placement scheme, we omit the details and just employ the maximum overhead assumption.

Let n be the number of currently active streams and τ the worst case disk overhead. Then the disk overhead in a round becomes $n \cdot \tau$. If t_{tr} denotes the block transfer time, a random variable L , which is the disk load in a round, is given by: $L = X \cdot t_{tr} + n \cdot \tau$. Since τ is a constant and L is a linear function of X which is normally distributed, L is also given by[8]:

$$L \sim \mathcal{N}(\mu t_{tr} + n \cdot \tau, t_{tr}^2 \sigma^2).$$

That is, the disk load in a round has a normal distribution.

4 Shortsighted Prediction Admission Control (SPAC)

An admission control scheme should have information on the amount of system resources that will be available in the future in order to effectively provide QoS guarantees. However, it is difficult to predict exactly the future utilization of resources especially in our system model, where round-level caching is used and objects are VBR-encoded. It is mainly because the disk load for each stream may vary from one round to another. The server can provide deterministic QoS by exact prediction but the computational overhead is too great for practical consideration. The admission control scheme SPAC proposed in this paper introduces statistical theory to reduce the computational overhead and provides a statistical but near-deterministic QoS. This section describes the on-line simulation for the exact prediction and the inference for the statistical estimation.

4.1 Exact Prediction of Disk Load

The *stream hit ratio*, which is defined as the proportion of the number of *cached* streams to the number of all the streams being serviced in a round, is important in terms of QoS. If the stream hit ratio of each round, that is, the saved disk bandwidth by caching, can be calculated in advance, admission control schemes can provide hiccup-free(or deterministic)

service to all clients. In *interval-level caching* such as *Interval Caching* and DISTANCE, since the following streams of cached intervals should not access the disk until it is replaced from cache, we can identify the streams to be cached in future rounds and hence calculate the future disk load simply. PSIC exploits this characteristic of interval-level caching to provide hiccup-free service[9]. On the other hand, in round-level caching, prediction of the future disk load is not simple because it is hard to determine which streams cause cache hits. Fortunately, as the server determines which media blocks to read and when they are read, it can exactly predict the disk load through on-line simulation and theoretically provide hiccup-free service with the worst case assumption. We consider first an admission control scheme executing the on-line simulation, including the new stream, of the cache status and disk load to make sure that there is no disk overload until the new stream is finished. The problem is whether the server should simulate future disk load until the new stream is finished for admitting it. It is evident that the disk load should be less as the more requests depart from the system. We may safely conclude that the admission will not cause the disk overload if no disk overload has been observed up to some future round during the simulation. The following explains how we can find such a round. Let \mathcal{B} be a round length and n_t be the number of streams being serviced in round t . Let us assume that $t_{tr}(B_i^{max})$ denotes the transfer time of the largest media block, B_i^{max} , of the object accessed by stream S_i , and τ denotes the worst-case disk overhead. If the following equation holds at some future round t_v , we can safely stop the simulation.

$$\sum_{i=1}^{n_v} t_{tr}(B_i^{max}) + n_v \cdot \tau \leq \mathcal{B}.$$

Letting t_l be the length of the video requested by the new client, the scheme admits a new client for service only if there is no disk overload during the simulation up to round $\min\{t_v, t_l\}$. It provides deterministic QoS, since it ensures that the disk overload never occurs during the service for the new stream even in the worst case where all streams retrieve the largest media block B_i^{max} from disk in the same round owing to cache misses. The computational overhead for this admission control is dependent on $\min\{t_v, t_l\}$, which is too large since the above condition is excessively conservative. For example, consider the server that services one CM object whose round length(\mathcal{B}) is 1 second, running time is 30 minutes and size of

the largest media block(B_i^{max}) is 2Mbits. Let this server have the capability to support 100 streams with stream hit ratio 0.2 in the heavy-loaded case. The number of non-cached streams, which invoke disk I/O in a round is 80 on average. Assuming the disk bandwidth(= $1/t_{tr}$) to be 160Mbps and the disk overhead to be zero(i.e., $\tau = 0$), the admission control requires on-line simulation until the number of streams being serviced, n_v , reduces below 80 (= $160Mbps/2Mbits$) according the above equation. In other words, the simulation runs until 20 streams out of 100 streams are finished. It ensures that there is no disk overload in the worst case that 80 streams induce disk I/O for B_i^{max} in the same round due to cache misses. Since there are 100 requests being serviced at a time with a total duration of 30 minutes, one stream would be finished every 18 seconds ($30\ min./100\ streams$) if the streams are assumed to be evenly distributed over the 30-minute object. A new request may arrive at any round during 18 seconds after the last stream started. Thus, the distance between the new request and the last admitted request is 9 seconds on average. Finally, the scheme should simulate about $351(= 9\ sec. \times 1\ stream + 18\ sec. \times 19\ streams)$ rounds before about 20 streams are finished. The computational overhead for one-round simulation is $O(n \log n)$ where n is the number of media blocks in the cache. Hence, an m -round simulation requires $O(mn \log n)$ time complexity. Since the admission control scheme should decide whether to admit a new stream in one round, for large n or m , it is hard to provide deterministic service.

SPAC introduces statistical estimation to reduce the computational overhead. Based on the idea that the available disk bandwidth may increase when one stream departs from the system, this scheme integrates an exact prediction through on-line simulation and statistical estimation of future disk load. SPAC determines whether admitting a new stream causes disk overload using on-line simulation round by round only up to the round where a stream first departs the system, hence the name *shortsighted prediction*. We call the round with the first departure the *visible round*. Then, the possibility of disk overload which may be experienced after the *visible round* is tested statistically. That is, the exact test by *shortsighted prediction* deterministically ensures no *starvation* due to disk overload up to the *visible round* and the statistical test by estimation does so after the *visible round*. It is reasonable to assume that the probability of disk overload while servicing $n - 1$ streams is very low since no disk overload

occurs while servicing n streams.

Let L_{new} denote the load to be introduced by a new stream, L_{old} the load to be reduced by a departing stream and L_{tot} be the current aggregated load. SPAC invokes the statistical test to admit the new stream only after the exact test is passed in order to test if the admission control criterion

$$\mathcal{P}(L_{tot} - L_{old} + L_{new} \geq \mathcal{B}) < \alpha$$

is satisfied, where α is a QoS level (e.g. 0.05). That is, SPAC admits a new stream if the probability of disk overload after the *visible round* is smaller than α . The statistical inference for derivation of the probability is dealt within the section 4.2. Since the overhead for statistical testing is negligible compared to exact testing, SPAC requires only about 1/39 (=9/351) of the computational overhead of the deterministic scheme in the above example.

4.2 Statistical Inference

4.2.1 Load Reduction by a Departing Stream

In order to model the reduction of the disk load by a departing stream, we must estimate the disk bandwidth used by the stream during its lifetime. Figure 2 shows a snapshot of the cache right after the disk service of a round has finished when using round-level caching. The horizontal bars indicate the intervals between two consecutive streams. The streams are classified into three types in terms of the length of the intervals, as depicted in Figure 2. If the interval is sufficiently small, the media blocks that the preceding stream reads from disk or cache always remain in the cache for the following stream to read, and hence the following stream is serviced from the cache at every round. These following streams are called *Steadily Cached(SC)* streams. On the other hand, if the interval is too large, the TTNR of the media block read by the preceding stream is also large. The following streams of those intervals are called *Non-Cached(NC)* streams, which have never been serviced from the cache. The streams which are serviced both from disk and from cache are called *Unsteadily Cached(UC)* streams. Since SC streams never access disks, the disk load induced by an SC stream is 0. On the other hand, the number of blocks read from disks by an NC stream S_i , X_i , has a

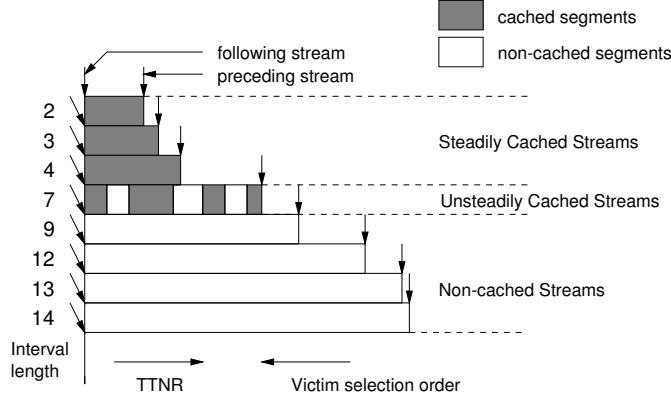


Figure 2: The Types of streams

normal distribution and accordingly the disk load L_i induced by S_i is a normal distribution if we assume the maximum disk overhead, as mentioned in section 3.2. Thus, the probability density function (p.d.f.) of L_i is given by:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma_l} e^{-\frac{(x-\mu_l)^2}{2\sigma_l^2}}, \quad (1)$$

where μ_l and σ_l^2 are the mean and the variance of the disk service time for the media blocks included in the CM object requested by S_i . We define the *access ratio* p_i as the following in order to derive the p.d.f. of L_i in the case that S_i is a UC stream.

$$p_i = \frac{\text{the number of rounds in which } S_i \text{ has invoked disk I/O}}{\text{the number of total rounds for which } S_i \text{ is serviced}}$$

When S_i is a UC stream, from the normal distribution and *access ratio*, the p.d.f. of L_i is defined by:

$$f(x) = \begin{cases} p_i \cdot \frac{1}{\sqrt{2\pi}\sigma_l} e^{-\frac{(x-\mu_l)^2}{2\sigma_l^2}} & \text{if } x \neq 0 \\ 1 - p_i & \text{if } x = 0. \end{cases} \quad (2)$$

Equation (2) generalizes the distribution of load by SC streams and NC streams as well as UC streams. p_i of an NC stream S_i is 0. Substituting 0 for p_i , Equation (2) is equal to Equation (1) which is the p.d.f. of a normal distribution with mean μ_l and variance σ_l^2 . Since p_i of an SC stream S_i is 1, Equation (2) trivially denotes that L_i is zero as mentioned above. Since the load reduction L_{old} caused by a departing stream S_d is equal to L_d , which S_d has induced

during its lifetime, the p.d.f. of L_{old} is given by Equation (2) which is equivalent to: (*w.p.* denotes *with probability*)

$$L_{old} \sim \begin{cases} \mathcal{N}(\mu_d, \sigma_d^2) & \text{w.p. } p_d \\ 0 & \text{w.p. } 1 - p_d, \end{cases} \quad (3)$$

where p_d is the *access ratio* of S_d and μ_d and σ_d^2 are the mean and variance of the disk service time for the media blocks included in the object requested by S_d .

4.2.2 Load Increase by an Arriving Stream

The *access ratio* p_d of a departing stream S_d is determined by keeping track of the number of *hits* through its lifetime. Thus, we could derive the distribution of the load reduction by the stream. Since, however, we cannot determine p_a of an arriving stream S_a before servicing it, we must estimate p_a to derive a random variable L_{new} denoting the disk load increased by S_a . The type of a stream depends on the length of the interval that the stream follows.

Theorem 1 *Let V_i and V_j be two intervals whose lengths are l_i and l_j , respectively, and S_i and S_j be the following streams of V_i and V_j , respectively. (1) If $l_i < l_j$ and S_j is an SC stream, S_i is also an SC stream. And (2) if $l_i > l_j$ and S_j is NC stream, S_i is also an NC stream.*

Proof Assume that S_i is not an SC stream when $l_i < l_j$ and S_j is an SC stream. Then, the media blocks consumed by the preceding stream of V_i may be replaced before being consumed by S_i . And, assuming that $TTNR_k$ is the TTNR of a media block B_k , $\max_{B_k \in V_i} \{TTNR_k\} < \max_{B_k \in V_j} \{TTNR_k\}$ since $l_i < l_j$. Thus, if the media blocks in V_i are replaced in a round, those in V_j would have already been replaced. That is, S_j is not an SC stream. By contradiction, (1) is true. We can also prove (2) in a similar manner to (1) \square

Consequently, the streams are divided into SC streams and NC streams based on a threshold which depends on cache size, and a small number of streams straddling the threshold are UC streams (see Figure 2).

Figure 3 shows the distribution of p of the departing streams from simulation results. The result that most p 's are close to 0 or 1 is consistent with Theorem 1. Hence it is reasonable

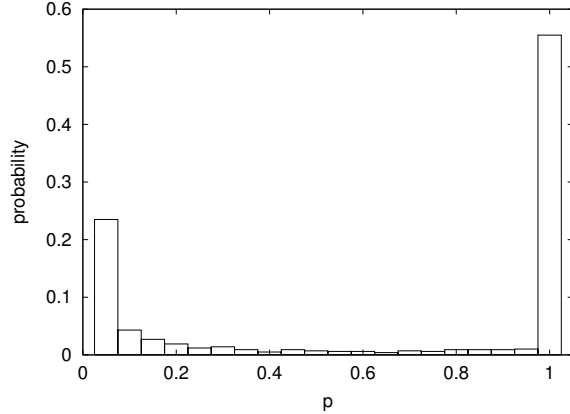


Figure 3: Distribution of p

to estimate p to be either 0 or 1. Assuming that l_i denotes the length of an interval V_i whose the following stream is S_i , the estimator of p is given by:

$$\hat{p}_a = \begin{cases} 0 & \text{if } \{S_i | l_a < l_i \wedge p_i \leq \epsilon\} \neq \phi \\ 1 & \text{otherwise} \end{cases} \quad (4)$$

where the *type threshold* ϵ is a tunable parameter equal to the point that decides whether \hat{p}_a is determined to be 0 or 1. L_{new} has the distribution given by the p.d.f. substituting \hat{p}_a for p_d in Equation (2).

4.2.3 Estimation of Aggregated Disk Load

Aggregated disk load L_{tot} means the total disk load induced by all the streams being currently serviced. We can model L_{tot} by estimating the load L_i of each stream S_i probabilistically as in section 4.2.2 and aggregating them. However, as the calculation requires many parameters to be estimated, there may result erroneous values due to cumulated error. Moreover, we can use the past actual disk load that servers maintain. Hence, the estimate based on the past samples of L_{tot} is more reasonable than that based on the estimates of L_i . We may determine the empirical distribution of L_{tot} with the samples, $x_1, x_2, \dots, x_\omega$, of the latest ω rounds, where ω is a *window size*. However, if L_{tot} is estimated from this empirical distribution, admission and termination of streams is not reflected in estimating L_{tot} . For example, consider the system

that can serve a maximum of n streams. Assume that there has been no stream in the system from $(t - \omega)^{th}$ round to current round (t^{th} round) and then $n + 1$ streams arrive in the t^{th} round. Although L_{tot} should be estimated to be larger than \mathcal{B} , it will be determined to be zero ($\sum_{i=1}^{\omega} x_i / \omega$) as there has been no stream during the last ω rounds. Hence, all of the $n + 1$ streams will be admitted. Consequently, even if the server is actually heavily loaded, it may accept a new request by wrong estimation as a result of not considering events such as arrival and departure.

To solve this problem, SPAC combines the actual load and the estimated load to make an estimate of L_{tot} responsive to such events. We decompose the aggregated total load L_{tot} into components. We refer to the streams whose *ages* are larger than ω rounds as *mature* streams and the others as *immature* streams. The set of NC streams, \mathcal{NC} , can be divided into the set of *mature* streams \mathcal{NC}_m and the set of *immature* streams \mathcal{NC}_i , i.e., $\mathcal{NC} = \mathcal{NC}_m \cup \mathcal{NC}_i$ and $\mathcal{NC}_m \cap \mathcal{NC}_i = \phi$. Since, as mentioned in section 4.2.2, the number of UC streams is very small, we classify all the streams into either SC streams or NC streams in terms of the *typethreshold* ϵ for the purpose of modeling L_{tot} . SC streams are not concerned in L_{tot} because they do not induce disk load. If L_i denotes the load induced by a stream S_i in a round, L_{tot} can be given by:

$$L_{tot} = \sum_{S_i \in \mathcal{NC}_m} L_i + \sum_{S_i \in \mathcal{NC}_i} L_i + e,$$

where e is the error caused by fluctuation of the load induced by UC streams. With $L_{mat} = \sum_{S_i \in \mathcal{NC}_m} L_i$ and $L_{imm} = \sum_{S_i \in \mathcal{NC}_i} L_i$, the aggregated disk load is given by $L_{tot} \approx L_{mat} + L_{imm}$ since it is reasonable that e is assumed to be small. As L_{mat} and L_{imm} are linear functions of random variables X_i , which, as described in section 3.2, are normal distributions and are mutually independent, they can be assumed to be normal distributions. Here we define additional sample data, $x'_1, x'_2, \dots, x'_\omega$, as the actual load caused by *mature* streams in a round. Since the load of *mature* streams is reflected in all ω samples, we can properly estimate L_{mat} from their empirical distribution. Then, L_{mat} is $\mathcal{N}(\mu_M, \sigma_M^2)$, where μ_M and σ_M^2 are estimated at the sample mean and sample variance of $x'_1, x'_2, \dots, x'_\omega$. On the other hand, since the load of an *immature* stream is not reflected in all ω samples, the theoretical distribution from Equation (3) and \hat{p}_a from Equation (4) are used to estimate L_{imm} . Hence, L_{imm} is $\mathcal{N}(\mu_I, \sigma_I^2)$,

where μ_I and σ_I^2 are estimated by $\hat{\mu}_I = \sum_{S_i \in \mathcal{NC}_i} \hat{\mu}_i$ and $\hat{\sigma}_I^2 = \sum_{S_i \in \mathcal{NC}_i} \hat{\sigma}_i^2$, where $\hat{\mu}_i$ and $\hat{\sigma}_i^2$ are the estimators for the mean and variance of the load induced by S_i . Consequently, since aggregated disk load L_{tot} is the sum of two normal distributions,

$$L_{tot} \sim \mathcal{N}(\mu_M + \mu_I, \sigma_M^2 + \sigma_I^2).$$

SPAC updates x_i' 's and a random variable L_{imm} to adjust T_{tot} . On departure of a stream S_i in round t , the load l_{ij} 's which S_i actually induced in the latest $(t - j + 1)^{th}$ round are subtracted from x_j . That is, $x_j^{t+1} = x_j^t - l_{i,t-j+1}$ for $j = 1, \dots, \omega$. When a stream S_i matures, L_i induced by S_i is subtracted from L_{imm} (i.e., $L_{imm} = L_{imm} - L_i$), and then the load l_{ij} 's which S_i actually induced in the latest $(\omega - j + 1)^{th}$ round are added to x_j . That is, $x_j^{t+1} = x_j^t + l_{i,t-j+1}$ for $j = 1, \dots, \omega$.

4.3 Statistical Test

We have derived L_{tot} , L_{old} and L_{new} in section 4.2. In this section, we describe how SPAC uses their distributions for admission control. SPAC examines the disk overload through on-line simulation up to a *visible round* and, if there is no overloaded round, tests the rest of the rounds statistically. $L(= L_{tot} - L_{old} + L_{new})$ provides a statistics to be used for testing. Since $L_{tot} - L_{old}$ and L_{new} are normally distributed, L , the sum of them, also follows a normal distribution. From Equation (3),

$$L \sim \begin{cases} \mathcal{N}(\mu_T - \mu_D + \mu_A, \sigma_T^2 + \sigma_D^2 + \sigma_A^2 - 2 \cdot \sigma_T \cdot \sigma_D) & w.p. \ p_d \\ \mathcal{N}(\mu_T + \mu_A, \sigma_T^2 + \sigma_A^2) & w.p. \ 1 - p_d, \end{cases}$$

where p_d is the *access ratio* of the departing stream S_d corresponding to L_{old} . If $f_D(x)$ is the p.d.f. when L is $L_{tot} - L_{old} + L_{new}$ and f_C is the p.d.f. when L is $L_{tot} + L_{new}$, F_D and F_C , which are the probability of overload on each case are given by

$$F_D = \int_{\mathcal{B}}^{\infty} f_D(x) dx, \quad F_C = \int_{\mathcal{B}}^{\infty} f_C(x) dx,$$

respectively. Finally, if the probability of overload

$$\mathcal{P}(L > \mathcal{B}) = F_D \cdot p + F_C \cdot (1 - p) < \alpha$$

is satisfied, the server admits the new stream corresponding to L_{new} .

Buffer Cache Size	32 MB
Number of CM Objects	20
Length of CM Objects	10 ~ 50 min
Avg. Bit Rate of CM Objects	0.272 ~ 0.768 Mbps
Round Length (\mathcal{B})	1 sec
Max. Disk overhead	22.99 msec
Disk Bandwidth	167 Mbps
Inter-arrival Time	5 sec
Zipf Parameter	0.2

Table 1: System parameters

5 Experimental Results

In this section, we demonstrate the performance of SPAC through trace-driven simulations. Table 1 shows the system parameters. For the purpose of the simulations, each video object is assumed to be encoded using VBR compression techniques. The trace data for frame size variation used for the simulation was obtained from [17]. These are MPEG1 traces of various movies and all data are VBR-encoded. We also assumed that the requesting pattern for objects has a Zipf distribution with parameter 0.2 (when the Zipf parameter is 0, the popularity is most highly skewed) and the arrival process of requests is Poisson with interarrival time of 5 sec.

In the simulations, the default values for the three parameters of SPAC, *type threshold* ϵ , *QoS level* α and *window size* ω , are 0.1, 0.05 and 100 rounds respectively. We observed disk utilization(or disk load) while proceeding in rounds, and the figures(Figure 4, Figure 5, Figure 6, and Figure 8) on disk utilization depict the results of part of total 15000 rounds of simulation. In [9], it is demonstrated that PSIC, which is a caching scheme providing deterministic QoS in CM servers, was better than other schemes devised for comparison with PSIC. Thus we have compared the performance of SPAC proposed in this paper with that of PSIC.

Figure 4 depicts disk utilization of PSIC and SPAC. SPAC achieves about 34% increase in

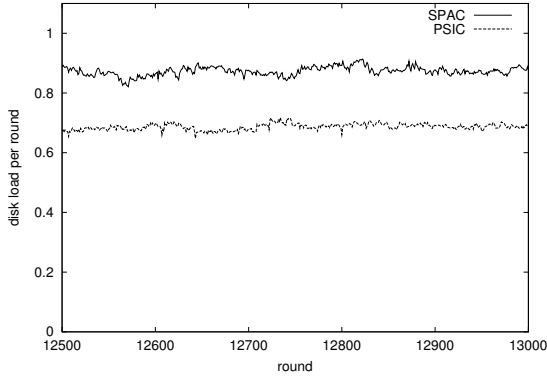


Figure 4: Disk Utilization: SPAC vs. PSIC

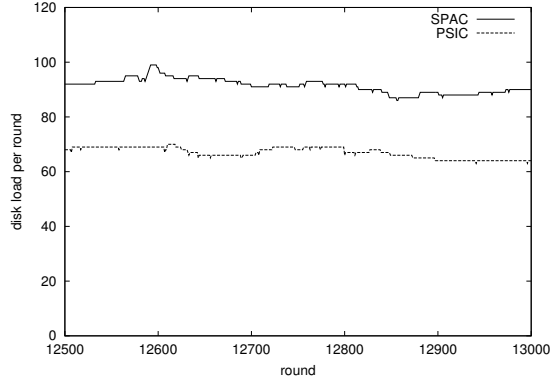


Figure 5: Number of Clients: SPAC vs. PSIC

the disk utilization per round compared to PSIC. Here, the disk utilization of SPAC was 0.91 and that of PSIC was 0.67 on average. In other words, the server using SPAC decreases the idle time of the disk by servicing more clients. PSIC does not utilize 33% of disk bandwidth in order to provide hiccup-free service in the worst case that is unlikely to take place. Owing to the effect of statistical multiplexing, the disk utilization of PSIC hardly changed from 0.67 and we did not observe any occurrence of the worst case where the disk load is close to 1. Although SPAC is a statistical admission control scheme which may experience occasional starvation, we have not encountered any disk overload under the default values of the parameters during the simulation. As depicted in Figure 5, SPAC achieves a 50% increase in the number of clients that can be serviced simultaneously by the server. Higher utilization leads to an increase in the number of clients and subsequently the lengths of the intervals become shorter. Since the shorter lengths of intervals result in a higher cache hit rate, the number of SC streams becomes greater. That is why the increase in the number of clients is larger than that for disk utilization.

Since the type of a new stream, which should be decided in advance, affects the estimation of L_{tot} as well as that of L_{new} , the *type threshold* ϵ greatly affects the disk utilization and the overload probability (see Equation (4)). Figure 6 shows the variation of the disk load with rounds when ϵ is 0.1 and 1.0 respectively. Theoretically, the closer ϵ gets to 1, the more UC streams SPAC regards as SC streams, i.e., the more aggressive it is. Thus the overload

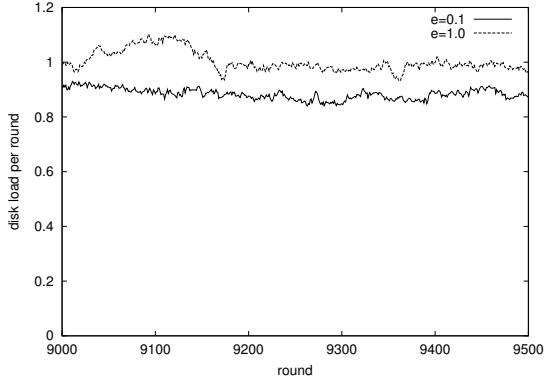


Figure 6: Effect of *type threshold* ϵ

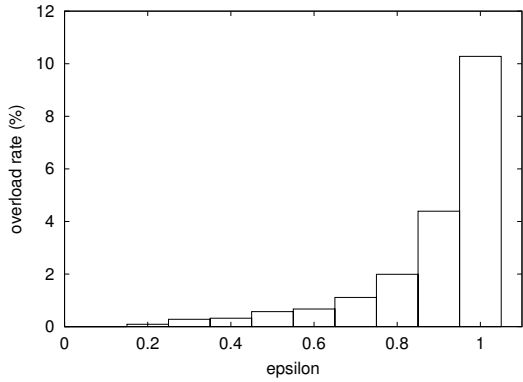


Figure 7: Variation of Disk Overload with ϵ

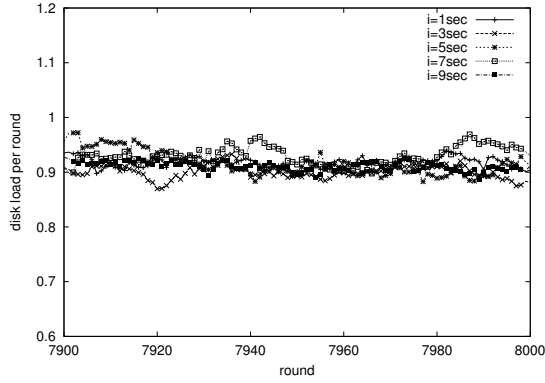


Figure 8: Effect of Interarrival Time

probability is high when ϵ is close to 1 and this is confirmed by the simulation result. The disk utilization is very high when ϵ is 1.0, and relatively low when ϵ is 0.1. Note that SPAC with ϵ equal to 1.0 induced disk overload in 10% of 15000 rounds, which is clearly unacceptable since disk overload results in the *starvation* of some streams due to aggressive admission.

We need to optimize performance considering the tradeoff between server capacity and QoS by tuning ϵ . We observed the overloaded probabilities while varying ϵ . As depicted in Figure 7, the overload ratio stays fairly small when ϵ is less than 0.6 and exponentially increases as ϵ gets close to 1. Using this as a guideline, the designer can decide the value of the *type threshold* ϵ for the system. In order to show that SPAC keeps disk utilization at a regular level irrelevant of the change of workload, we measured disk utilization while varying interarrival time. As depicted in Figure 8, the pattern of all the lines of utilization are

cache size	avg. util.	hit ratio	# of clients
32 MB	0.91	0.21	94
64 MB	0.90	0.32	107
96 MB	0.90	0.39	119
128 MB	0.92	0.43	126
256 MB	0.89	0.56	160

Table 2: Effect of Cache Size

alike. The results demonstrate that interarrival time does not affect the robustness of SPAC. Table 2 shows how cache size has an effect on the behavior of SPAC. We observed average utilization, the number of client that can serviced simultaneously (i.e., server throughput) and hit ratio, while varying cache size. The results that hit ratio is higher, as cache size increases. SPAC improves the throughput by admitting more requests as the amount of disk bandwidth saved by caching increases. However, the average utilization is insensitive to cache size. That is, our scheme is also robust against cache size.

6 Concluding Remarks

In continuous media systems, the disk load can be reduced by using caching. In order to utilize the saved disk bandwidth resulting from caching, a continuous media system must employ an admission control scheme. In this paper, we have presented a statistical admission control scheme for continuous media servers which employ caching, called *Shortsighted Prediction Admission Control*(SPAC), which combines exact prediction through on-line simulation and statistical estimation by probabilistic modeling of future disk loads. SPAC determines whether admitting a new stream(or client) causes disk overload due to on-line simulation which continues until one stream is finished. Then the possibility of disk overload which may occur subsequently is tested statistically. This scheme exploits the variation in disk load induced by VBR-encoded objects and the reduction in server load by caching.

We have demonstrated the effectiveness of SPAC through trace-driven simulations. Our simulation results show that, as compared with its deterministic counterpart PSIC, SPAC

achieves about a 50% increase in the number of streams serviced simultaneously and is able to provides near-deterministic QoS guarantees.

References

- [1] D. Anderson, Y. Osawa, and R. Govindan. A File System for Continuous Media. *ACM Transactions on Computer Systems*, 10(4):311–337, November 1992.
- [2] Pei Cao and Sancy Irani. Cost-aware WWW Proxy caching Algorithms. In *Proc. of the USENIX Symposium on Internet Technologies and Systems*, pages 193–206, December 1997.
- [3] A. Dan, R. Mukherejee, D.M. Dias, D. Sitaram, and R. Tewari. Buffering and Caching in Large-scale Video Servers. In *Compton-Technologies for the Information Superhighway*, pages 217–224, Los Alamitos, CA, June 1995.
- [4] E.Knightly and H.Zhang. Traffic Characterization and Switch Utilization Using Deterministic Bounding Interval Dependent Traffic Models. In *Proc. of IEEE INFOCOM '95*, pages 1137–1145, Boston, MA, 1995.
- [5] J. Gemmell and S. Christodoulakis. Principles of Delay-sensitive Multimedia Data storage and Retrieval. *ACM Trans. of Information Systems*, 10(1):51–90, January 1992.
- [6] J. Gemmell, Harrick M. Vin, Dilip D. Kandlur, and Venkat Rangan. Multimedia Storage Servers: A Tutorial and Survey. *IEEE Computer*, 28(5):40–49, May 1995.
- [7] S. Gollapudi and A. Zhang. Buffer Management in Multimedia Database Systems. In *Proc. of IEEE International Conference on Multimedia Computing and Systems*, pages 186–190, Hiroshima, Japan, June 1996.
- [8] Robert V. Hogg and Elliot A. Tanis. *Probability and Statistical Inference*. Prentice Hall, Eaglewood Cliffs, New Jersey, 1993.

- [9] KyungOh Lee, Jin B. Kwon, and Heon Y. Yeom. Exploiting Caching for Realtime Multimedia Systems. In *Proc. of IEEE International Conference on Multimedia Computing and Systems*, Florence, Italy, June 1999.
- [10] Raymond T. Ng and Rita Dilek. Statistical Modeling and Buffer Allocation for MPEG Streams. In *Multimedia Information Storage and Management*, pages 147–162. Kluwer Academic Publishing, 1996.
- [11] Raymond T. Ng and Jinhai Yang. An Analysis of Buffer Sharing and Prefetching Techniques for Multimedia Systems. *ACM Multimedia Systems*, 4(2):55–69, April 1996.
- [12] Banu Özden, Rajeev Rastogi, and Avi Silberschatz. Buffer Replacement Algorithms for Multimedia Storage Systems. In *Proc. of IEEE International Conference on Multimedia Computing and Systems*, pages 172–180, Hiroshima, Japan, 1996.
- [13] David A. Patterson and John L. Hennessy. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publisher, San Francisco, California, 1996.
- [14] Harrick M. Vin, Pawan Goyal, Alok Goyal, and Anshuman Goyal. A Statistical Admission Control Algorithm for Multimedia Servers. In *Proc. of ACM Multimedia*, pages 33–40, San Francisco, October 1994.
- [15] Harrick M. Vin and P. Venkat Rangan. Designing a Multi-user HDTV Storage Server. *IEEE Journal on Selected Areas in Communications*, 11(1):153–164, January 1993.
- [16] P.S. Yu, M.S. Chen, and D.D. Kandlur. Design and Analysis of a Grouped Sweeping Scheme for Multimedia Data. In *Proc. of 3rd International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 38–49, San Diego, November 1992.
- [17] Informatik MPEG-I traces. <ftp://ftp-info3.informatik.uni-wuerzburg.de/pub/MPEG>.