A Performance Model of Effective Memory Management in HYDRA: a Large Scale Data Stream Recording System

Kun Fu and Roger Zimmermann
Integrated Media Systems Center, University of Southern California
Los Angeles, CA, USA 90089-2561
[kunfu, rzimmerm]@usc.edu

ABSTRACT
Presently, digital continuous media (CM) are well established as an integral part of many applications. Scant attention has been paid to servers that can record such streams in real time. However, more and more devices produce direct digital output streams. Hence, the need arises to capture and store these streams with an efficient recorder that can handle both recording and playback of many streams simultaneously and provide a central repository for all data. Because of the continuously decreasing cost of memory, more and more memory is available on a large scale recording system. Unlike most previous work that focuses on how to minimize the server buffer size, this paper investigates how to effectively utilize the additional available memory resources in a recording system. We propose an effective resource management framework that has two parts: (1) a dynamic memory allocation strategy, and (2) a deadline setting policy (DSP) that can be applied consistently to both playback and recording streams, satisfying the timing requirements of CM, and also ensuring fairness among different streams. Furthermore, to find the optimal memory configuration, we construct a probability model based on the classic $M/G/1$ queueing model and the recently developed Real Time Queuing Theory (RTQT). Our model can predict (a) the missed deadline probability of a playback stream, and (b) the blocking probability of recording streams. The model is applicable to admission control and capacity planning in a recording system.

Keywords: continuous media recording system, memory management, EDF scheduling

1. INTRODUCTION
Digital continuous media (CM) are an integral part of many new applications. A considerable amount of research has focused on the efficient retrieval of CM. However, more and more devices produce direct digital output streams. Hence, the need arises to capture and store these streams with an efficient data stream recorder that can also support many concurrent playback streams simultaneously. Memory and disk bandwidth are two of the most important resources in a recording system. Most of the previous studies\cite{1-3} are based on the minimum buffer settings. With the continuously decreasing cost of the memory, a large memory capacity can be available on a large scale recorder. In this paper, we study the memory management issue in a recording system, HYDRA\cite{4}, which has limited, but more than minimal, memory. More specifically, we are interested in the following questions: (1) How should we allocate the buffers among playback and recording streams? (2) Can we allocate more buffers to one stream compared with others so that the stream with more memory gets better QoS? (3) Can we find an optimal buffer allocation scheme to maximize the system throughput while satisfying the client’s QoS requirements?

To answer these questions, we propose an effective resource management framework that has two parts. The first component is a dynamic memory allocation strategy. The second part is a deadline setting policy (DSP) that can be applied consistently to both playback and recording streams, satisfying the timing requirements of CM streams, and also ensuring fairness among different streams. Furthermore, to find the optimal memory configuration, we construct a probability model based on the classic $M/G/1$ model\cite{5} and the recently developed Real Time Queuing Theory (RTQT)\cite{6}. Our model can predict (a) the missed deadline probability of a playback stream with a given memory allocation, and (b) the blocking probability of recording streams. The model is applicable to admission control and capacity planning in a recording system. To the best of our knowledge, it is the first model that applies RTQT to the analysis of a streaming system. Specifically, our model characterizes: (1) deadline driven disk scheduling, (2) random data placement, (3) variable bit rate (VBR) streams, (4) concurrent reading and writing streams, (5) the difference in disk read/write bandwidth, (6) the transfer rate variability.

---

This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC), CMS-0219463 (ITR), and equipment gifts from the Intel Corporation, Hewlett-Packard, Sun Microsystems, and Raptor Networks Technology.
in multi-zoned disks, and (7) the variable seek time and rotational latency of disk I/O operations. The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 describes our proposed dynamic buffer sharing scheme and in Section 4 we discuss our design of deadline setting policies. Section 5 includes our probability model. In Section 6 we present evaluation results. Finally, Section 7 concludes the paper with some future work.

2. RELATED WORK

There are only a few papers\(^{1-4,7,8}\) that have studied both disk read and write issues for streaming architectures. Most of these techniques assume a minimum server buffer environment (i.e., double buffering) except\(^{7,8}\). Aref et al.\(^8\) proposed a technique that dynamically computes the deadlines for disk writing requests for video editing systems. Their technique assigns the same deadline to all the writing requests and consistently adjusts the deadline based on the available amount of buffers in the system. Therefore, the overhead of reorganizing the request waiting queue can be very high when the system experiences workload fluctuations. Moreover, a higher priority is given to read requests compared with write requests. Rangaswami et al.\(^2\) investigated the design of an interactive media server based on a fine-grained device management strategy. They designed a very constrained data placement technique for multi-zone disks and adopted the double buffering scheme in their system. In our initial memory management paper\(^5\) we studied the MSB problem, which minimizes the server buffer size under a given service requirement. Ghaderifarizadeh and Kim\(^7\) proposed a multi-buffer technique to minimize the hiccup probability in a CM editing server. To the best of our knowledge, no prior work has investigated the fairness issue of DSP and the issue of dynamic allocation of the buffers to different streams in the context of recording systems. Furthermore, no prior work has constructed a probability model that can evaluate performance trade-offs between different DSPs and buffer allocation on the recording system.

3. DYNAMIC MEMORY BUFFER SHARING

![Diagram]

**Figure 1.** A data stream recording architecture.

Fig. 1 shows the simplified architecture of HYDRA.\(^4\) There are two generally accepted paradigms to assign data blocks to the magnetic disk drives that form the storage system: in a *round-robin* sequence,\(^9\) or in a *random* manner.\(^10\) Traditionally, the round-robin placement utilizes a cycle-based approach to scheduling of resources to guarantee the service quality, while the random placement utilizes a deadline-driven approach. The latter provides a number of advantages such as support for multiple or variable delivery rates with a single storage data block size, easy support for interactive applications, and support for data reorganization during storage system scaling. All these features may be supported with cycle-based scheduling. However, it results in a complex implementation and – most importantly – many disk parameters must be assumed with their worst case values. Therefore, deterministic guarantees are obtained at the expense of efficiency. Deadline-driven scheduling can be configured to be both very efficient and to incur a very low probability of disruptions. HYDRA adopts random data placement with an *earliest deadline first* (EDF) disk scheduling algorithm, which can achieve high system utilization and provide statistical service guarantee. To effectively utilize server memory, we identify the following desirable Design Goals for the buffer sharing scheme: (1) share the buffers among playback and recording streams as much as possible, and (2) dynamically allocate the available buffer CM streams on demand. We consider each in turn.

**Buffer Sharing Scheme:** Given \(n\) buffers, when a recording system is under normal workload, the buffer usages for a playback or recording stream are shown in Fig. 2. Based on the typical buffer usage comparison, we make several important observations: (1) Sharing buffers among several playback streams is not feasible because the buffers allocated to each playback stream are usually occupied by movie data as shown in Fig. 2a. (2) Sharing buffers among a playback and a recording stream is difficult due to their almost conflicting usage of buffers. (3) Sharing buffers among several recording streams is possible, since under normal workload, recording streams usually have plenty of empty buffers, which could be easily utilized by other recording streams.
Based on these observations, we decided to partition the total memory buffers dynamically for each playback stream respectively and for all the recording streams as a shared buffer pool. Let $M$ denote the total number of buffers available in the system. Let $M_R$ and $M_W$ denote the number of buffers allocated to playback and recording streams, respectively. Let $M_R(i)$ denote the number of buffers allocated to playback stream $i$, where $i$ is an index of playback streams. Another important result we have obtained from our simulation (for details see11) is that each playback stream may require a different buffer size to achieve the same QoS requirement, i.e. the probability of a request missed deadline. Intuitively, this is because each stream may have different bandwidth requirements. For this reason, it is not desirable to evenly allocate the reading buffers $M_R$ among all playback streams. Next, we need to determine the appropriate values for $M_R$, $M_W$, and $M_R(i)$, where $i \in [1, N_r]$. Let $N_{rs}$ and $N_{ws}$ denote the number of concurrent playback and recording streams in the system. Our first approach is to allocate the memory resources according to streams’ bandwidth requirements. Thus,

$$M_W = M \frac{\sum_{i=1}^{N_{rs}} \mu_i^w}{\sum_{i=1}^{N_{rs}} \mu_i^w + \sum_{i=1}^{N_{ws}} \mu_i^w}, \quad M_R(i) = M \frac{\mu_i^r}{\sum_{i=1}^{N_{rs}} \mu_i^r + \sum_{i=1}^{N_{ws}} \mu_i^w} \quad (1)$$

where $\mu_i^r$ and $\mu_i^w$ denote the average bandwidth requirement for playback stream $i$ and recording stream $i$, respectively.

Our second approach is to construct a probability model (in Section 5) that can be used to find the optimal configurations.

4. DEADLINE SETTING POLICY FOR N-BUFFERING SCHEME

When a stream is allocated with $N$ buffers, the buffer allocation scheme is termed an $N$-buffering scheme. With more buffers allocated, there is more flexibility in configuring the disk I/O deadlines. In this section, we first establish the timing requirements for playback and recording streams respectively. Based on these timing requirements, we deduce the timing requirements for the deadline setting. Then, we propose several DSPs that can satisfy these timing requirements.

4.1. Timing Analysis for Continuous Media Streams with Multiple Buffers

From the storage system point of view, a playback/recording stream is essentially a sequence of disk read/write requests with real-time requirements. The real-time requirement for a disk read or write request can be characterized by two parameters: (1) the request issue time, and (2) the true deadline of the request.

**Definition 4.1.** The True Deadline of a disk I/O request is the deadline that must be satisfied in order to ensure the continuous playback and recording of streaming applications.

One of the scheduler design principles we chose is to ensure the work-conserving property. Thus, the disk I/O requests are always issued as early as possible. Detailed timing requirements are included in a technical report.11

4.2. Valid Deadline Setting Policies (V-DSP)

We define the two very important concepts below:

**Definition 4.2.** The Virtual Deadline of a disk I/O request is the deadline that is used by the deadline driven disk scheduling algorithm to choose which request should be serviced from the request waiting queue in the system.

**Definition 4.3.** A Deadline Setting Policy (DSP) is an algorithm that is used to compute the issue times and virtual deadlines for every disk request of a playback or recording stream.

Furthermore, we define the DSP that satisfies the timing requirements for continuous playback or recording streams as a Valid Playback Deadline Setting Policy (VP-DSP) or a Valid Recording Deadline Setting Policy (VR-DSP) respectively. The timing requirement overview in Section 4.1 provides a useful guideline to the design of a valid DSP. Since we adopt the
work-conserving scheduling design principle, for a playback or recording stream, when the movie start time \( T_{\text{movieStart}} \) and recording start time \( T_{\text{recordStart}} \) are determined, the disk I/O requests' issue times can be determined. Thus, the key design space for a DSP is the determination of the virtual deadline. We propose three sets of DSPs: (1) a true deadline setting policy (T-DSP), (2) a dynamic deadline setting policy (D-DSP), and (3) a fair deadline setting policy (F-DSP). T-DSP sets the virtual deadline of each request to their true deadline. D-DSP dynamically sets the virtual deadline of each request according to the available memory resources at the request issue time. F-DSP sets the virtual deadline of each request based on the consumption or fill duration of 2 buffers of data. We have proved that all three proposed DSPs are valid DSPs. More detailed definitions and proofs are included in a technical report.\(^{11}\)

4.3. Measuring Fairness in Deadline Setting Policies

Since the HYDRA system provides statistical service guarantees, the system might be temporarily overloaded, which may cause some of the disk I/O requests to miss their deadlines. In such a system, a fair DSP is necessary, because (1) it can ensure that those streams that have been allocated with similar memory and disk bandwidth resources can receive similar QoS in terms of the missed deadline probability, and (2) it can also provide better QoS to a specific stream by allocating more system resources to it. We introduced two performance metrics to evaluate the fairness of DSP in HYDRA: (1) a throughput measure to evaluate the fairness in disk I/O throughput of streams, and (2) a system time measure to capture the fairness on timing requirements, which is inspired by an earlier paper.\(^ {12}\)

**Definition 4.4.** The System Time Measure (STM) captures the fairness in satisfying the timing requirement by measuring the distribution of \( T_{\text{system}} \) (system time) of a request of a targeted stream.

In HYDRA, a closely matched \( T_{\text{system}} \) distribution from any two streams implies that the DSP ensures a better fairness in satisfying the timing requirements. In our simulation experiments, we found that all the proposed valid DSP could ensure the fairness in the disk throughput allocation. Therefore, we only focus on the second measure in this paper.

5. QUEUEING SYSTEM MODEL

![Figure 3. The conceptual queueing model of the recording system.](image)

5.1. Characterization of Queueing System

The HYDRA system shown in Fig. 1 can be modelled as a queueing system. We assume that the disk I/O requests arrive according to a Poisson distribution. We prove that this assumption is valid through a hypothesis test of the measured samples from streaming workloads generated from real-movie traces in Section 6. We construct a realistic service time model by considering detailed disk I/O characteristics, such as variable transfer rates in multi-zoned disks and nonlinear seek time for disk I/O operations. The disk I/O requests are served one by one according to the EDF scheduling policy, and we consider a single server (disk) at this point\(^{1}\). Therefore, the queueing system can be conceptually converted into the queueing model shown in Fig. 3. Because of the buffer sharing scheme (Section 3) and bandwidth sharing policy\(^{1}\) in HYDRA, the queueing system can also be treated as two logical queues: a reading request queue (RQueue) and a writing request queue (WQueue). We modelled the RQueue and WQueue as M/G/1/\( \infty \)/Deadline and M/G/1/K/Deadline queueing systems, respectively. We chose a general service time distribution because the disk service time is not exponentially distributed.

For a playback stream, a disk reading request is driven by the consumption of movie data. When the system is heavily loaded, a disk may not be able to retrieve data blocks in time, which could result in a long reading request waiting queue. From a read request point of view, there is unlimited waiting space in the RQueue. When a system is overloaded, some

\(^{1}\)Since HYDRA employs a random data placement scheme, the workload will be naturally balanced across multiple disks. Therefore, it is fairly easily to extend our model to a multi-server environment.
requests may miss their deadline. Therefore, we are interested to compute the probability of missing a request deadline for a playback stream. For a recording stream, data is continuously accumulated in server write buffers. When the system is overloaded, it may not be able to service write requests in time and writing buffers may be used up. From a writing request point of view, WQueue has limited waiting room. Recall that all the recording streams share a writing buffer pool. When new data is arriving, if all the buffers are used, the server has two choices: (Method 1: Blocking) throw away new data; or (Method 2: Overwriting) overwrite the data buffer that has not been written to disk. In HYDRA, we adopted the Blocking approach and we are interested in finding the blocking probability of a new writing request.

5.2. Deriving $P_B$: the Probability of Blocking for Recording Streams

Since the WQueue is modeled as a $M/G/1/K/Deadline$ system, by applying the formulas in the classical $M/G/1/K$ model, we can obtain the probability of a new disk request being blocked due to writing buffer limitation $M_W$.

$$P_B = P_{\text{req \& w}}(\mathcal{M}_W) = \frac{(1 - \rho_w)q_{M_W}}{1 - q_{M_W}\rho_w}$$

(2)

where $\rho_w$ is the system utilization of WQueue and $q_{M_W}$ denotes the tail probability of the corresponding $M/G/1/\infty$ WQueue (for the detailed derivation see11).

5.3. Deriving $P_{\text{read,missd}}(i)$: the Probability that a Disk Read Request Will Miss its Deadline

Let $P_{\text{read,missd}}(i)$ denote the probability that a reading request from playback stream $i$ missed its deadline in RQueue, where $i$ is an index of all the playback streams. We assume that all the requests we discussed in this section are requests from playback stream $i$ unless explicitly stated otherwise. Therefore, we omit the index $i$ in the discussion. Fig. 4 depicts the lifetime of a disk I/O request. Note that we define the request system time as $T_{\text{system}} = T_{\text{wait}} + T_{\text{service}}$. One important concept we want to introduce is the Lead Time of a request.

**DEFINITION 5.1.** The Lead Time of a request $(T_{\text{leadtime}})$ at time instant $T_0$ is defined as the remaining time until the deadline of a disk I/O request since $T_0$. That is $T_{\text{leadtime}} = T_{\text{deadline}} - T_0$.

The Lead Time of a request decreases linearly as time passes. Let $T_{\text{leadtime,arrival}}$ denote the $T_{\text{leadtime}}$ for a request at the time instant when the request just arrived into the system. With some derivation,11 we have

$$P_{\text{read,missd}}(i) = P[T_{\text{system}} > T_{\text{leadtime,arrival}}]$$

(3)

$T_{\text{leadtime,arrival}}$ is determined by the inherent bandwidth characteristics for playback stream $i$ and DSP. Next, we derive the distribution of $T_{\text{system}}$ by utilizing the real time queuing theory (RTQT).6

**Deriving the Distribution of $T_{\text{system}}$ for a request in RQueue**

Let $\lambda$ denote the mean customer arrival rate. $Q$ is the length of the waiting queue at time $T_0$. Let $G(x)$ denote the cumulative distribution function (CDF) of the lead time of arriving customers. For a $M/G/1$ queue with ED queueing discipline, at time $T_0$, given $\lambda, Q$, $G(x)$, RTQT computes the lead time profile of all the requests in the queue as a probability distribution with a probability density function (pdf) given by: $f_X(x) = \frac{\lambda}{Q}(1 - G(x))$ if $L(Q) \leq x \leq \infty$. $L(Q)$ is the departing customer’s (the next customer to be serviced) lead time at $T_0$, which can be determined by $\frac{\lambda}{Q} \int_{L(Q)}^{\infty} (1 - G(x)) dx = 1$. By applying RTQT, we can obtain

$$T_{\text{system}} = T_{\text{leadtime,arrival}} - L(Q) + T_{\text{service}}$$

(4)
where $T_{\text{leadtime, arrival}}$ denotes the initial lead time of a request from any playback streams. Substituting $T_{\text{system}}$ in Equation 3, we obtain

$$P_{\text{read,miss}}(i) = P \left[ T_{\text{leadtime, arrival}} + L(Q) + T_{\text{service}} > T_{\text{stream, }i} + T_{\text{leadtime, arrival}} \right]$$

(5)

Note in Equation 3, the $T_{\text{leadtime, arrival}}$ denotes the lead time of a read request from a specific playback stream $i$, while in Equation 4, the $T_{\text{leadtime, arrival}}$ denotes the lead time of a read request from any playback stream. To avoid confusion, we distinguish them with $T_{\text{read,miss},i}$ and $T_{\text{read,miss}}$ respectively. (A detailed derivation can be found in11).

6. PERFORMANCE EVALUATION

We implemented our proposed DSPs and dynamic buffer allocation scheme in a simulation system shown in Fig. 5. The WorkLoad Generator produces client playback or recording requests based on a Poisson process with a mean inter-arrival time of $\frac{1}{\lambda} = 5$ seconds. The movie blocks are randomly placed onto a disk and block requests are scheduled based on the EDF scheduling policy. Deadlines for block requests are computed based on (a) DSP, (b) buffer allocation, and (c) real movie traces. The block requests are forwarded to the Disk by the Disk Access Scheduler at the set times. The WorkLoad Generator has several configurable parameters: the mean inter-arrival time $\frac{1}{\lambda}$, the number of retrieval streams $n_{rs}$ and the number of recording streams $n_{wrs}$. Our storage system simulates a Seagate disk drive (Model ST336752LC). Disk block size $B_{\text{disk}}$ is set to 1.0 MB.

6.1. Fairness Comparison between T-DSP, D-DSP and F-DSP

Fig. 6 shows the experimental results with the three proposed deadline setting policies: true deadline setting policy (T-DSP), dynamic deadline setting policy (D-DSP), and fair deadline setting policy (F-DSP). The comparison is based on the System Time Measure (STM). In the simulation, 58 concurrent playback streams are running, each stream is generated from a trace of the DVD movie “Twister”. Half of the 58 streams are each allocated with 2 buffers, and each of the other 29 streams has 30 buffers. Each figure shows two curves of the relative frequency histogram of the system time $T_{\text{system}}$. One of the curve is from a stream with 2 buffers, the other is from a stream with 30 buffers. Fig. 6(a), (b) and (c) show the results with T-DSP, D-DSP and F-DSP, respectively. With T-DSP, the $T_{\text{system}}$ for the stream with 2 buffers is significantly less than the stream with 30 buffers as shown in Fig. 6(a) in terms of both the mean value and the standard deviation. This implies that the T-DSP is unfair for these two streams. On the other hand, with F-DSP, the $T_{\text{system}}$ distribution curves match reasonable well for those two streams. Note that the two streams used in the comparison are randomly selected from those two sets of streams with different buffer sizes. The results of D-DSP lie between T-DSP and F-DSP. Therefore, we can conclude that F-DSP is the fairest among all three proposed DSPs (see8 for more results and explanations).
6.2. Evaluation of Buffer Sharing Scheme for Recording Streams

Recall that a write buffer pool will be shared among all the recording streams in HYDRA, and we call this scheme Shared Buffer Scheme (SBS). In this section, we evaluate the effectiveness of our proposed SBS by comparing it with a Partitioned Buffer Scheme (PBS), where the server buffer capacity is equally partitioned for each recording stream. The comparison metric is the blocking probability for recording streams. We conducted recording experiments with both SBS and PBS. In all the recording experiments, we adopted the F-DSP as discussed in the previous section.

Fig. 7 shows how the blocking probability of incoming data changes as a function of the server memory capacity with both buffer allocation schemes with two different system workloads, 38 streams and 40 streams. In these experiments, the recording streams are generated from the DVD movie trace “Saving Private Ryan”. In both figures, the blocking probability decreases gradually to zero as the server buffer capacity increases for both schemes. Moreover, SBS consistently results in a lower blocking probability than PBS throughout all the memory configurations. In general, the benefit of SBS increases as memory capacity increases. In summary, compared to PBS, our proposed SBS improves the HYDRA system performance by taking advantage of the statistical multiplexing of different I/O bandwidth requirements across different recording streams. See[11] for more results and explanations.

6.3. Hypothesis Test of Disk I/O Request Arrival Process

To verify our model assumption that the interarrival time of the disk I/O requests follows an exponential distribution, we conducted a set of $\chi^2$ Chi-Square Goodness-of-Fit Test[11] on a set of measured samples generated from real movie traces in our simulation system. The measured data samples are divided into $k$ bins and the test statistics is defined as $\chi^2 = \sum_{i=1}^{k} \left( \frac{O_i - E_i}{E_i} \right)^2$ where $O_i$ is the observed frequency of measured samples for bin $i$, $E_i$ is the expected frequency of samples for bin $i$, which can be computed as $E_i = N \frac{e^{-\lambda T_i} - e^{-\lambda T_{i-1}}}{T_i - T_{i-1}}$. $N$ denotes the total number of measured data samples, $T_0$ and $T_k$ denote the upper and lower limits for each bin, $\lambda$ is the mean arrival rate of disk I/O requests, which can be estimated by computing the average arrival rate from the measured samples. Basically, the smaller values of the $\chi^2$ test statistics show that the observed and expected distributions are more similar (a better fit). Fig. 8 shows the observed measured samples and the computed expected values in different experiments. In all these experiments, the computed expected frequencies fit the measured sample data surprisingly well. See[11] for more results and explanations.

6.4. Verification of RTQT in HYDRA

Since nobody has applied RTQT to the streaming environment where the VBR streams can generate an arbitrary deadline distribution for arrival requests, we have conducted extensive simulations to verify the applicability of RTQT in HYDRA.
Fig. 9 compares the RTQT predicted lead time ($T_{\text{leadtime}}$) profile with the measured average lead time snapshots when the request waiting queue length is 100. Fig. 9(a), (b) and (c) show the results when the workloads are generated from different movie traces. Each figure contains 5 curves, the last two curves labelled with “predicted gX CDF” and “avg LT snapshot” are the RTQT prediction and measurement results (the other 3 curves are intermediate computation results of RTQT). The average lead time values are collected from a large number of snapshots. For example, 1879 snapshots for the experiment with VCD streams. In all figures, RTQT predictions closely match the simulation results. Therefore, we argue that RTQT is applicable to our timing analysis in HYDRA. More results can be found in.\textsuperscript{11}

7. CONCLUSIONS

We presented an effective resource management framework that is composed of a dynamic memory allocation strategy and a novel deadline setting policy (F-DSP) in a large scale recording system. Moreover, we constructed a probability model based on the classic M/G/1 queueing model and the recently developed Real Time Queuing Theory (RTQT) that can be used to evaluate the performance trade-off of different buffer allocation and DSPs. Our next step is to apply the developed model to find the optimal memory allocation, and to provide admission control and capacity planning.

REFERENCES