Randomized Data Allocation in Scalable Streaming Architectures*

Kun Fu and Roger Zimmermann

Integrated Media Systems Center
University of Southern California
Los Angeles, California 90089
[kunfu, rzimmerm]@usc.edu

Abstract. IP-networked streaming media storage has been increasingly used as a part of many applications. Random placement of data blocks has been proven to be an effective approach to balance heterogeneous workload in multi-disk steaming architectures. However, the main disadvantage of this technique is that statistical variation can still result in short term load imbalances in disk utilization. We propose a *packet level randomization* (PLR) technique to solve this challenge. We quantify the exact performance trade-off between PLR approach and the traditional *block level randomization* (BLR) technique through both theoretical analysis and extensive simulation. Our results show that the PLR technique can achieve much better load balancing in scalable streaming architectures by using more memory space.

1 Introduction

Large scale digital continuous media (CM) servers are currently being deployed for a number of different applications. Magnetic disk drives are usually the storage devices of choice for such streaming servers and they are generally aggregated into arrays to enable support for many concurrent users. Multi-disk CM server designs can largely be classified into two paradigms: (1) Data blocks are striped in a *round-robin* manner [1] across the disks and retrieved in *cycles* or *rounds* for all streams. (2) Data blocks are placed *randomly* [5] across all disks and the data retrieval is based on a *deadline* for each block. The first paradigm attempts to guarantee the retrieval or storage of all data. It is often referred to as *deterministic*. With the second paradigm, a disk may briefly be overloaded, leading to a few missed deadlines. This approach is often called *statistical*.

We focused on the statistical approach because of its many advantages. For example, a much higher resource utilization can be achieved. Moreover, the statistical approach can be implemented on widely available platforms such as Windows or Linux that do not provide hard real time guarantees. It can also naturally support a variety of different media types that require different data rates (both constant (CBR) or variable (VBR)) as well as interactive functions such as pause, resume and fast-forward. Moreover, it has been shown that the performance of a system based on the statistical method is on par with that of a deterministic

* This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC), IIS-0082826 and CMS-0219463, and unrestricted cash/equipment gifts from Intel, Hewlett-Packard, Raptor Networks Technology and the Lord Foundation.

system [6]. Finally, it can support on-line data reorganization more efficiently [3], which is very crucial for scalable storage systems. Even though the statistical approach is very resource efficient and ensures a balanced load across all disk devices over the long term, short term load fluctuations may occur because occasionally consecutive data blocks may be assigned to the same disk drive by the random location generator. During the playback of such a media file, the block retrieval request queue may temporarily hold too many requests such that not all of them can be served by their required deadline.

In this paper we introduce a novel *packet-level randomization* (PLR) technique that significantly reduces the occurrence of deadline violations with random data placement. PLR is the focus of the remainder of this paper which is organized as follows. Section 2 reviews the related work. Section 3 presents our proposed design. Performance analysis and evaluation are contained in Section 4 and 5, respectively. Finally, Section 6 outlines our future plans.

2 Related Work

Three techniques have been proposed to achieve load balancing in striped multi-disk multimedia storage systems. One approach is to use large stripes that access many consecutive blocks from all disks at a single request for each active stream [8]. It provides perfect load balancing because the number of disk I/Os is the same on all the devices. However, it results in extremely large data requests with large number of disks. Furthermore, it does not efficiently support unpredictable access patterns. The second approach uses small requests accessing just one block on a single disk, with sequential requests cycling over all the disks [1]. This technique does not support unpredictable access patterns well. The third technique randomly allocates data blocks to disks blocks [6, 7], and therefore supports unpredictable workloads efficiently. To our knowledge, no prior work has quantified the exact trade-off between the randomization at the packet and block levels when fine-grained load balancing is desired or required.

3 Design Approach

We assume a multi-disk, multi-node streaming media server cluster design similar to the one used in our previous research activities. Our first prototype *Yima* [7], was a scalable streaming architecture to support applications such as video-on-demand and distance learning on a large scale. Our current generation system, termed the *High-performance Data Recording Architecture* (HYDRA) [10] improves and extends Yima with real time recording capabilities. For load-balancing purposes, without requiring data replication, a multimedia object X is commonly striped into blocks , e.g., $X_0, X_1, \ldots, X_{n-1}$, across the disk drives that form the storage system [4, 8]. Because of its many advantages, we consider randomly allocating data to the disk drives.

3.1 Packets versus Blocks

Packet-switched networks such as the Internet transmit relatively small quanta of data per packet (for example 1400 bytes). On the other hand, magnetic disk drives operate very inefficiently when data is accessed in small amounts. This is

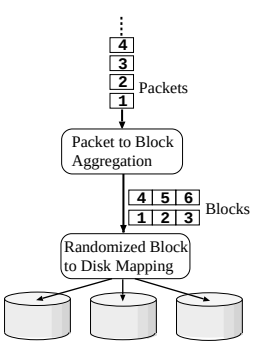

Fig. 1(a): Block Level Randomization (*BLR*)

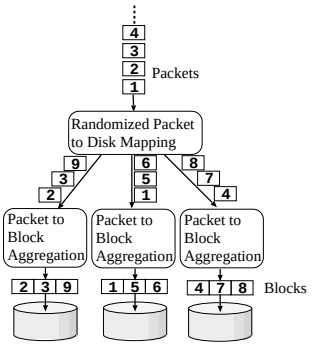

Fig. 1(b): Packet Level Randomization (*PLR*)

Fig. 1. Two different randomization schemes that can be applied in a Recording System, e.g. HYDRA [10]. Note that in this example, each block contains 3 packets.

due to the fact that disk drives are mechanical devices that require a transceiver head to be positioned in the correct location over a spinning platter before any data can be transferred. The seek time and rotational latency are wasteful [10]. Consequently, media packets need to be aggregated into larger data blocks for efficient storage and retrieval. Traditionally this is accomplished as follows.

Block-Level Randomization (BLR): Media packets are aggregated in sequence into blocks (see Figure 1(a)). For example, if m packets fit into one block then the data distribution algorithm will place the first m sequential packets into block X_0, the next m packets into block X_1, and so on. As a result, each block contains sequentially numbered packets. Blocks are then assigned randomly to the available disk drives. During retrieval, the deadline of the first packet in each block is essentially the retrieval deadline for the whole block. The advantage of BLR is that only one buffer *at a time* per stream needs to be available in memory across all the storage nodes. In order to allow high disk utilization while still reducing the probability of hot-spots we propose a novel technique as follows.

Packet-Level Randomization (PLR): Each media packet is randomly assigned to one of the storage nodes, where they are further collected into blocks (Figure 1(b)). One advantage is that during playback data is retrieved randomly from all storage nodes at the granularity of a packet. Therefore, load-balancing is achieved at a very small data granularity. The disadvantage is that memory buffers need to be allocated concurrently on all nodes per stream. In the next section we quantify the load-balancing properties of both BLR and PLR.

4 Performance Analysis

We evaluate *PLR* and *BLR* with three metrics: (1) the uniformity of data distribution on each disk, (2) the disk I/O imbalance across every disk during a streaming experiment, and (3) the memory size and potential caching effects.

4.1 Data Placement Imbalance Analysis

With *PLR*, let the random variable X denote the number of packets assigned to a specific disk. As proposed in [3], uniformity of data distribution can be measured

Term	Definition	Units
N_D	Total number of disks	
M	Total data size	MB
S_P	Packet size	MB
S_B	Block size	MB
M_B	Total data size in blocks	
M_P	Total data size in packets	
D_{PLR}	The amount of data assigned to a disk in PLR	MB
D_{BLR}	The amount of data assigned to a disk in BLR	MB
X	The number of packets assigned to a disk in PLR	
Y	The number of blocks assigned to a disk in BLR	
R	Ratio of block size and packet size, i.e., $\frac{S_B}{S_P}$	
N_C	The number of concurrent clients supported by the storage server	
N_S	Total number of storage server nodes	
${S_{mem}}^{blr}$	Memory size required for BLR	MB
${S_{mem}}^{plr}$	Memory size required for PLR	MB
α	The number of disks attached to each server node [1], i.e. $\alpha = \frac{N_D}{N_S}$	

Table 1. List of terms used repeatedly in this study and their respective definitions.

by the standard deviation and the coefficient of variation (CV) of X, represented by σ_X and $CV(X)$, respectively. $CV(X)$ can be derived from dividing σ_X by the mean value of X, μ_X. If we consider the random assignment of a packet to a disk as a Bernoulli trial, which has probability $p = \frac{1}{N_D}$ to be successfully allocated to a specific disk, then the total number of successful Bernoulli trials could be naturally mapped to X. Intuitively, X follows a Binomial distribution, where the number of Bernoulli trials is the total number of packets to be assigned, and denoted as M_P. Note that M_P can be computed as $M_P = \frac{M}{S_P}$, where M is the total data size and S_P is the packet size. Therefore, we can obtain

$$\mu_X = \frac{M}{S_P \times N_D}\ ,\ \sigma_X = \sqrt{\frac{M \times (N_D - 1)}{S_P \times N_D^2}}\ ,\ CV(X) = \sqrt{\frac{(N_D - 1) \times S_P}{M}} \times 100 \quad (1)$$

With BLR, let Y denote the number of blocks assigned to a disk. Furthermore, M_B represents the total data size in blocks. M_B can be calculated as $M_B = \frac{M}{S_B}$, where S_B denotes the block size. Similar to PLR, we obtain

$$\mu_Y = \frac{M}{S_B \times N_D}\ ,\ \sigma_Y = \sqrt{\frac{M \times (N_D - 1)}{S_B \times N_D^2}}\ ,\ CV(Y) = \sqrt{\frac{(N_D - 1) \times S_B}{M}} \times 100 \quad (2)$$

Let D_{PLR} and D_{BLR} denote the amount of data assigned to a disk with PLR and BLR, respectively. Then, D_{PLR} and D_{BLR} can be calculated by

$$D_{PLR} = XS_P\ ,\ D_{BLR} = YS_B \quad (3)$$

Using Equations 1 and 3, we obtain the mean, standard deviation and coefficient of variation of D_{PLR} as expressed in Equation 4.

$$\mu_{D_{PLR}} = \frac{M}{N_D}\ ,\ \sigma_{D_{PLR}} = \sqrt{\frac{S_P^3 M(N_D - 1)}{N_D^2}}\ ,\ CV(D_{PLR}) = \sqrt{\frac{(N_D - 1) \times S_P^3}{M}} \times 100 \quad (4)$$

Similarly, with Equation 3 and 2, we can obtain the mean, standard deviation and coefficient of variation of D_{BLR} as Equation 5.

$$\mu_{D_{BLR}} = \frac{M}{N_D}\ ,\ \sigma_{D_{BLR}} = \sqrt{\frac{S_B^3 M(N_D - 1)}{N_D^2}}\ ,\ CV(D_{BLR}) = \sqrt{\frac{(N_D - 1) \times S_B^3}{M}} \times 100 \quad (5)$$

Finally, from Equations 4 and 5, we obtain:

$$\mu_{D_{BLR}} = \mu_{D_{PLR}} \ , \ \frac{\sigma_{D_{BLR}}}{\sigma_{D_{PLR}}} = \frac{CV(D_{BLR})}{CV(D_{PLR})} = R^{\frac{3}{2}} \tag{6}$$

where $R = \frac{S_B}{S_P}$. There are two important observations we obtain from Equation 6. First, the mean values of D_{BLR} and D_{PLR} are the same, which confirms that both the *BLR* and *PLR* schemes achieve the same level of load balancing in the long run as expected. Second, with respect to short term load balancing, *PLR* has a significant advantage over *BLR*.

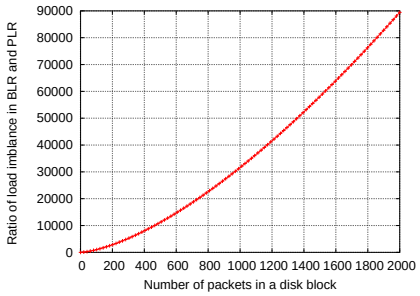

Fig. 2. Load imbalance ratio $\frac{\sigma_{D_{BLR}}}{\sigma_{D_{PLR}}}$, with different block to packet size ratio $R = \frac{S_B}{S_P}$.

Impact of Block to Packet Size Ratio R: Figure 2 shows the ratio of load imbalance $\frac{\sigma_{D_{BLR}}}{\sigma_{D_{PLR}}}$ as a function of the block to packet size ratio R. When R increases from 1 to $2,000$, $\frac{\sigma_{D_{BLR}}}{\sigma_{D_{PLR}}}$ increases sharply from 1 to approximately $90,000$. The figure clearly indicates the significant performance gain of *PLR* over *BLR*. In fact, $2,000$ packets in one block is not unusual. For example, a 512 byte packet size and a 1 MB block size are a quite common configuration in streaming servers [7].

Parameters	Configurations
Test movie "Twister"	MPEG-2 video, AC-3 audio
Average bandwidth	698,594 bytes/sec
Length	115 minutes
Throughput std. dev.	308,283.8
RTP packet size	512 bytes
Total number of RTP packets	10,740,000

Table 2. Parameters for movie "Twister" used in analysis.

Impact of the Number of Disks N_D: Figure 3(a) shows the standard deviation of the amount of data assigned to a disk in *BLR* and *PLR*, i.e., $\sigma_{D_{BLR}}$ and $\sigma_{D_{PLR}}$, respectively, as a function of the number of disks N_D on a logarithmic scale. Note that the data assigned is from the DVD movie "Twister" (see Table 2). As shown in the Figure 3(a), $\sigma_{D_{BLR}}$ is larger than $\sigma_{D_{PLR}}$ by several orders of magnitude, which implies that *PLR* allocates the movie data much more evenly across all the disks than *BLR*. Furthermore, when the total number of disks N_D increases from 2 to 200, $\sigma_{D_{BLR}}$ decreases from 35 MB to 5 MB.

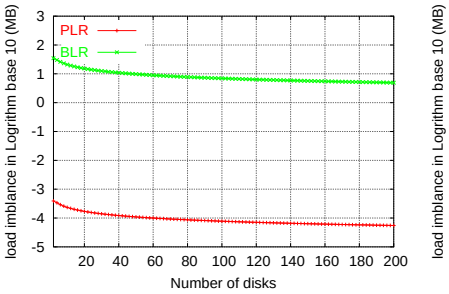

Fig. 3(a): Impact of the number of disks N_D based on DVD movie "Twister" (see Table 2).

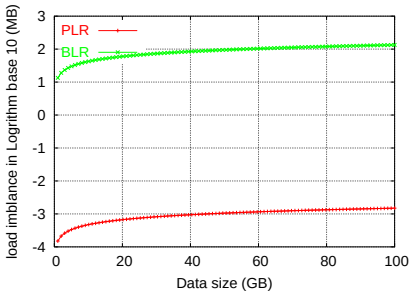

Fig. 3(b): Impact of data size M, with $N_D = 4$, $S_P = 512$ bytes, and $R = 2,000$.

Fig. 3. Impact of the number of disks N_D and data size M on the standard deviation $\sigma_{D_{BLR}}$ and $\sigma_{D_{PLR}}$. Note that the figures are logarithmic in scale.

Similarly, $\sigma_{D_{PLR}}$ also follows this trend. In fact, we can formally prove that, if $N_D \geq 2$ and as N_D increases, $\sigma_{D_{BLR}}$ and $\sigma_{D_{PLR}}$ both decrease monotonically as given in Lemma 42.

Lemma 41 *Let* $A(n) = \frac{\sqrt{n-1}}{n}$,$\forall n \geq 2$, $A(n+1) < A(n)$.

Proof. $\forall n \geq 2$, $A(n) > 0$, thus, we need to prove Equation 7,

$$\frac{A(n+1)}{A(n)} < 1 \Longleftrightarrow \sqrt{\frac{n^3}{(n+1)^2(n-1)}} < 1 \tag{7}$$

To prove Equation 7, we need to show Equation 8.

$$n^3 - (n+1)^2(n-1) < 0 \tag{8}$$

Equation 8 can be rewritten as $-[(n-1)^2 + (n-2)] < 0$, which is always true for all $n > 2$.

Lemma 42 $\forall N_D \geq 2$, *if* M, S_P *and* S_B *are fixed, both* $\sigma_{D_{PLR}}$ *and* $\sigma_{D_{BLR}}$ *monotonically decrease as* N_D *increases.*

Proof. Because S_P and M are constant, using Equation 4 and Lemma 41, it is straightforward to prove that, as N_D increases, $\sigma_{D_{PLR}}$ monotonically decreases. Similarly, since S_B and M are fixed, and using Equation 5 and Lemma 41, we can prove that, as N_D increases, $\sigma_{D_{BLR}}$ monotonically decreases.

Impact of the Data Size M**:** Figure 3(b) shows the load imbalance metric for both schemes as a function of the data size M. In the analysis, the total number of disks $N_D = 4$, packet size $S_P = 512$ bytes for *PLR*, and block to packet ratio $R = 2,000$ for *BLR*. As illustrated, as the data size M increases from 0 to 100 GB, the load imbalance metric of *BLR* $\sigma_{D_{BLR}}$ increases sharply from 0 to more than 120 MB. Similarly, the load imbalance metric of *PLR* $\sigma_{D_{PLR}}$ also increases, but because it is several orders of magnitude smaller than $\sigma_{D_{BLR}}$, $\sigma_{D_{PLR}}$ is still less than 2 Kbytes for $M = 100$ GB. This figure confirms that, when more data are loaded or recorded into the storage system, the imbalance of the amount of data assigned across all the disks will increase significantly, which explicitly shows the great performance gain of *PLR*. Next, we compare the disk load imbalance through the analysis of a streaming experiment.

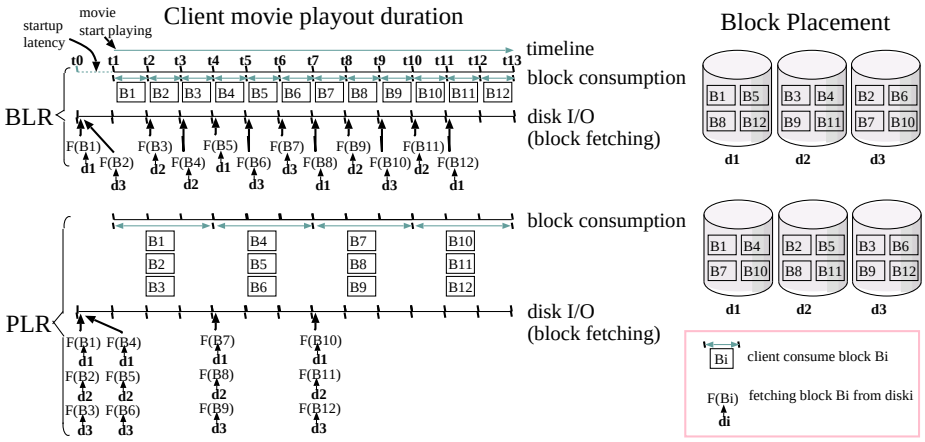

Fig. 4. Illustration of the disk I/O load imbalance during the playback of a CBR movie with $N_D = 3$ and $M_B = 12$ for both the *BLR* and *PLR* schemes.

4.2 Disk I/O Imbalance Time Analysis

As suggested by [2], the disk load imbalance during a predefined measurement period is characterized by the *Global Standard Deviation* $\sigma_{B_{disk}}$, which is defined as the utilized disk I/O bandwidth of all the N_D disks [2], shown in Equations 9.

$$\sigma_{B_{disk}} = \sqrt{\frac{\sum_{i=1}^{N_D}(L_i - \mu_{B_{disk}})^2}{N_D}} \tag{9}$$

where L_i denotes the utilized disk I/O bandwidth during the measurement period for disk i, $i \in [1, N_D]$ and $\mu_{B_{disk}}$ represents the mean value of the utilized disk I/O bandwidth and can be computed as:

$$\mu_{B_{disk}} = \frac{1}{N_D}\sum_{i=1}^{N_D} L_i \tag{10}$$

Our experimental setup is as follows. Three disks ($N_D = 3$) are attached to a server. A client streams a constant bit rate (CBR) movie from the server, and the movie contains 12 blocks ($M_B = 12$) of size $S_B = 1$ MB. The movie consumption rate is 1 MB/s. Therefore, a block is consumed every second during the movie playback, and the movie length is 12 seconds. The server employs random data placement with deadline driven disk scheduling algorithm. Recall that the deadline of each block is set to the first packet in each block. A simple double buffering scheme is adopted for memory management.

Figure 4 shows a detailed analysis of all the disk I/O events during the playback of the movie. During the movie startup period, which is between time

[2] We believe that in the analysis of storage systems, the absolute values are intuitively more understandable. Thus, we do not normalize the *Global Standard Deviation* by the mean value.

t0 and t1, the server prefetches some blocks. Because of double buffering, in *BLR* two blocks, $B1$ and $B2$, are prefetched, while in *PLR* six blocks are prefetched. A client starts the movie playback at time t1. With *BLR*, after every second when one block is consumed, the server fetches the next block. This process continues until the end of the movie. In *PLR*, the process is similar to *BLR* except that it takes 3 seconds from t1 to t4 for the client to consume blocks $B1$, $B2$ and $B3$. Note that because the randomness granularity is at the packet level, these three blocks are consumed almost at the same time. Subsequently, at time t4, blocks $B7$, $B8$ and $B9$ are fetched in parallel. A similar procedure is repeated at time t7 for blocks $B10$, $B11$ and $B12$. Note that *PLR* exploits the disk I/O parallelism naturally in this multi-disk environment. Table 3 summarizes the

Parameters		Time Slots												
Scheme	Statistics (MB/s)	t0 -t1	t1 -t2	t2 -t3	t3 -t4	t4 -t5	t5 -t6	t6 -t7	t7 -t8	t8 -t9	t9 -t10	t10 -t11	t11 -t12	t12 -t13
BLR	L_1	1	0	0	0	1	0	0	1	0	0	0	1	0
	L_2	0	0	1	1	0	0	0	0	1	0	1	0	0
	L_3	1	0	0	0	0	1	1	0	0	1	0	0	0
	$\mu_{B_{disk}}$	0.67	0	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0
	$\sigma_{B_{disk}}$	0.471	0	0.471	0.471	0.471	0.471	0.471	0.471	0.471	0.471	0.471	0.471	0
PLR	L_1	2	0	0	0	1	0	0	1	0	0	0	0	0
	L_2	2	0	0	0	1	0	0	1	0	0	0	0	0
	L_3	2	0	0	0	1	0	0	1	0	0	0	0	0
	$\mu_{B_{disk}}$	2	0	0	0	1	0	0	1	0	0	0	0	0
	$\sigma_{B_{disk}}$	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3. Disk I/O imbalance computation results during the playback of a 12 seconds CBR movie for both *BLR* and *PLR* scheme.

computed load imbalance for *BLR* and *PLR* based on Equations 9 and 10 for each second during the movie playback. Throughout the movie playback session, *PLR* perfectly balances the load, while *BLR* suffers from load imbalance with a global standard deviation value of 0.471 MB/s during more than 80% of the playout time. Next, we compare the impact of the memory size and its usage.

4.3 Memory Usage Analysis

Memory Size Requirement. Assuming that the same number of disks are attached to each storage server node and let N_S denote the total number of server nodes. Let us further assume that double buffering techniques are adopted in the system. In the *PLR*, the blocks from multiple disks within one server node can be accessed in parallel even for a single stream. Therefore, two buffers per disk are necessary for each *stream*. However, in *BLR*, because the blocks from multiple disks within one server node will be accessed sequentially for a single stream, only two buffers per server *node* are required for each stream. Accordingly, ${S_{mem}}^{blr}$ and ${S_{mem}}^{plr}$, the memory size required for BLR and PLR, can be computed as

$$ {S_{mem}}^{blr} = 2 \times S_B \times N_C \times N_S \ , \ {S_{mem}}^{plr} = 2 \times S_B \times N_C \times N_D \qquad (11) $$

where N_C denotes the number of concurrent clients. We define α as the ratio between the total number of disks N_D and N_S, i.e, $\alpha = \frac{N_D}{N_S}$. Therefore, we obtain $\frac{S_{mem}^{plr}}{S_{mem}^{blr}} = \alpha$, which means that *PLR* requires more memory resources than *BLR* when more than one disk is attached to each server node.

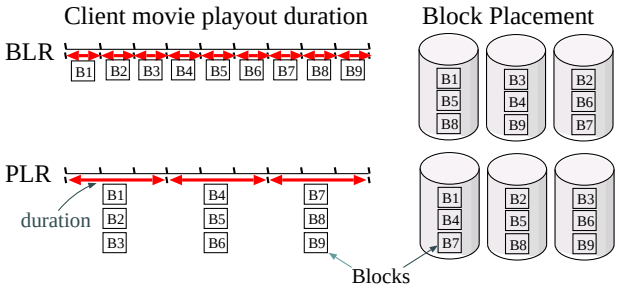

Fig. 5. Illustration of the memory access duration during the playback of a CBR (constant bit rate) movie with $N_D = 3$ and $M_B = 9$ for both the BLR and PLR techniques.

Memory Access Duration. Figure 5 shows the *memory access duration* during the playback of a CBR (constant bit rate) movie with $N_D = 3$ and $M_B = 9$ for both the BLR and PLR techniques. It clearly shows that the access duration for a block in PLR is N_D times that of the BLR scheme. Intuitively, due to the finer granularity of randomness, the buffered blocks from multiple disks are used simultaneously, which naturally leads to much longer consumption time for each memory buffer. Because the memory access duration is the minimum time that the corresponding blocks must be kept in memory, we believe that PLR could potentially result in greater caching effects in the server.

5 Performance Evaluation

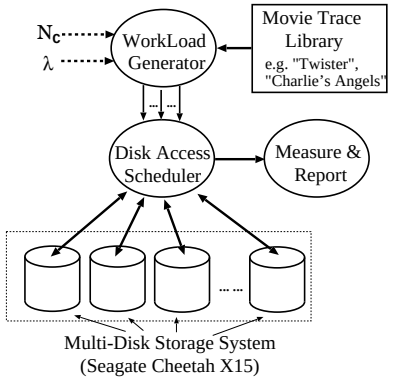

Fig. 6. Experimental system setup.

5.1 Experimental Setup

To evaluate the performance of PLR and BLR in a more practical environment, we integrated both the BLR and PLR methods into a simulation system. Fig. 6 illustrates the structure of our experimental setup. Note that we did not integrate a full fledged streaming server into our simulation system to reduce the number of factors that would influence the results. The WorkLoad Generator

produces stream requests based on a Poisson process with a mean inter-arrival time of $\lambda = 2$ seconds. Each stream retrieval produces data block requests based on either the PLR or BLR schemes with associated disk I/O deadlines according to movie traces from the Movie Trace Library. The movie blocks are allocated to disks in BLR or PLR schemes. The block requests are forwarded to the corresponding disk by the Disk Access Scheduler at the set times. The Measure & Report module generates the measured result. In a deadline driven streaming system, one of the most important parameters is the probability of a disk I/O request deadline miss, denoted p_{iodisk}. In the output report, both the number of requests with missed deadlines and the total number of disk block requests are collected. Furthermore, the ratio between these two numbers, which represents the fraction of the missed deadline requests, is interpreted as the probability of missed deadlines p_{iodisk}. The WorkLoad Generator has two configurable parameters: the mean inter-arrival time λ and the number of movie streams N_C. In the experiments, we used the DVD movie "Saving Private Ryan," whose profile is shown in Fig.1(a) in [9]. Our disk system simulates four independent Seagate Cheetah X15 disk drives. Table 4 summarizes all the used parameters.

Parameters	Configurations
Test movie "Saving Private Ryan"	MPEG-2 video, AC-3 audio
Average bandwidth	757,258 bytes/sec
Length	50 minutes
Throughput std. dev.	169,743.6
Disk Model "Seagate Cheetah X15"	Model ST336752LC
Capacity	37 GB
Spindle speed	15,000 rpm
Avg. rotational latency	2 msec
Worst case seek time	$\approx$ 7 msec
Number of Zones	9
Transfer rate	See Fig.1(b) in [9]
Mean inter-arrival time λ of streaming request	2 seconds
Data Packet size S_P	0.5 KB
Disk block size S_B	1.0 MB
Number of disks N_D	4
Number of concurrent clients N_C	1, 2, 3, ..., 230

Table 4. Parameters used in the experiments.

5.2 Experimental Results

Comparison based on *Global Standard Deviation* $\sigma_{B_{disk}}$: Figure 7 shows the *global standard deviation* measured during streaming experiments with the number of concurrent clients N_C being 10, 20, 50, and 100, respectively. In all these four scenarios, PLR significantly improves the load balancing over BLR in the multi-disk system. For example, with 50 concurrent streams, PLR decreases the global standard deviation from 1.0721 MB/s to 0.3263 MB/s with $N_C = 50$ and from 1.4982 MB/s to 0.4593 MB/s with $N_C = 100$.

Figure 8(a) compares the general trend of the average global standard deviation during each experiment as a function of the number of concurrent streams N_C. As N_C increases, the average global standard deviation increases, which verifies our analysis results in Section 4.1. That is, as the data size M increases the imbalance also increases. Note that in all these measurement, PLR reduces the load imbalance significantly.

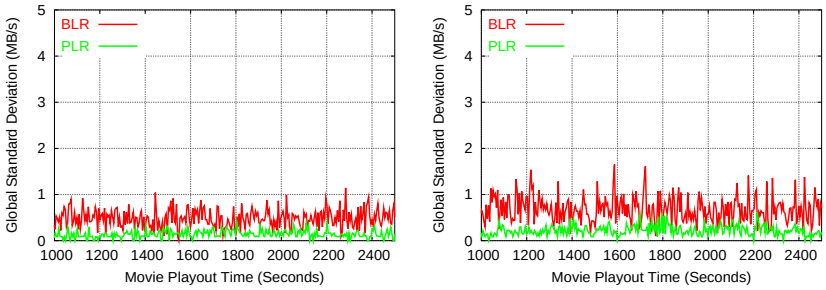

Fig. 7(a): $N_C = 10$ streams.

Fig. 7(b): $N_C = 20$ streams.

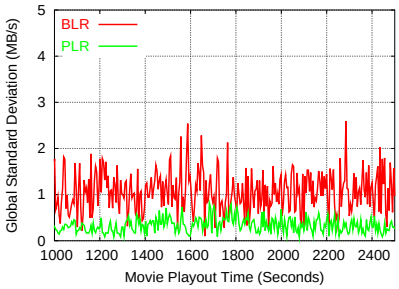

Fig. 7(c): $N_C = 50$ streams.

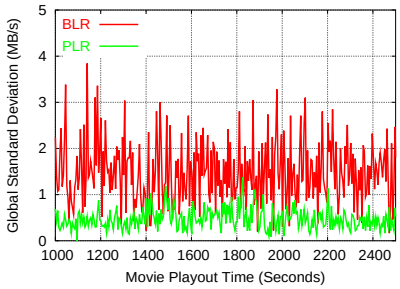

Fig. 7(d): $N_C = 100$ streams.

Fig. 7. Disk load imbalance across time for different number of concurrent DVD streams ("Saving Private Ryan"), where $N_C = 10$, 20, 50, and 100 respectively.

Comparison based on the Probability of a Disk I/O Request Missed Deadline p_{iodisk}: To evaluate the performance impact of the two schemes BLR and PLR at the system level, we compared the measured results based on the probability of a disk I/O request missing its deadline. Figure 8(b) shows the measured p_{iodisk} as a function of the number of concurrent streams N_C for both PLR and BLR. With PLR, the system always experiences fewer disk I/O requests that missed their deadlines. For example, with 205 concurrent streams, the system reported 0% of the total I/O requests that missed their deadlines with PLR, compared to 37.91% with the BLR scheme. This implies that with the PLR scheme, the system can support more concurrent streams than with the BLR scheme. Assuming that the end user can tolerate up to 1% of disk I/O request missed deadlines, then with PLR, the current experimental system setup could support 206 streams, but it can only support 199 streams with the BLR scheme, which is approximately a 3.5% improvement in terms of the number of supportable streams.

6 Conclusions

Load balancing is important to ensure overall good performance in a scalable multimedia storage system. This paper identifies and quantifies the performance trade-off of the packet level randomization (PLR) scheme over the traditional block level randomization (BLR) scheme. Both BLR and PLR ensures long term load balancing. But PLR achieves much better short term load balancing over

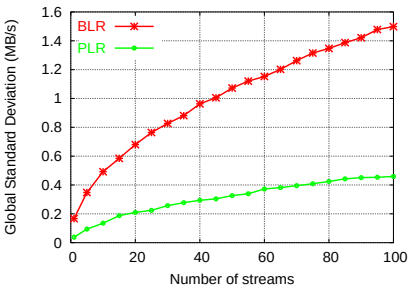

Fig. 8(a): The average disk load imbalance in terms of the global standard deviation with different number of streams.

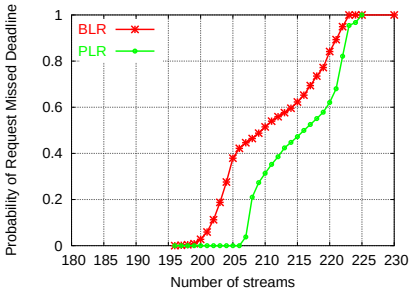

Fig. 8(b): Probability of a request missed deadline with different number of streams.

Fig. 8. Important experimental results.

BLR by utilizing more memory space. However, we believe the benefit of *PLR* outweighs its disadvantage since the cost of memory is continually decreasing. Therefore, *PLR* is a promising technique for high-performance media servers. Furthermore, we plan to implement the *PLR* approach into our streaming prototype, HYDRA, and evaluate its performance with real measurements.

References

1. S. Berson, S. Ghandeharizadeh, R. Muntz, and X. Ju. Staggered Striping in Multimedia Information Systems. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1994.
2. Antonio Corradi, Letizia Leonardi, and Franco Zambonelli. Diffusive load-balancing policies for dynamic applications. *IEEE Concurrency*, 7(1):22–31, January-March 1999.
3. A. Goel, C. Shahabi, S.-Y. D. Yao, and R. Zimmermann. SCADDAR: An Efficient Randomized Technique to Reorganize Continuous Media Blocks. In *Proceedings of the 18th International Conference on Data Engineering*, pages 473–482, February 2002.
4. V.G. Polimenis. The Design of a File System that Supports Multimedia. Technical Report TR-91-020, ICSI, 1991.
5. J. R. Santos and R. R. Muntz. Performance Analysis of the RIO Multimedia Storage System with Heterogeneous Disk Configurations. In *ACM Multimedia Conference*, Bristol, UK, 1998.
6. J. R. Santos, R. R. Muntz, and B. Ribeiro-Neto. Comparing Random Data Allocation and Data Striping in Multimedia Servers. In *Proceedings of the SIGMETRICS Conference*, Santa Clara, California, June 17-21 2000.
7. C. Shahabi, R. Zimmermann, K. Fu, and S.-Y. D. Yao. Yima: A Second Generation Continuous Media Server. *IEEE Computer*, 35(6):56–64, June 2002.
8. F.A. Tobagi, J. Pang, R. Baird, and M. Gang. Streaming RAID-A Disk Array Management System for Video Files. In *First ACM Conference on Multimedia*, August 1993.
9. Roger Zimmermann and Kun Fu. Comprehensive Statistical Admission Control for Streaming Media Servers. In *Proceedings of the 11th ACM International Multimedia Conference*, Berkeley, California, November 2-8, 2003.
10. Roger Zimmermann, Kun Fu, and Wei-Shinn Ku. Design of a large scale data stream recorder. In *The 5th International Conference on Enterprise Information Systems (ICEIS 2003), Angers - France*, April 23-26 2003.