Joint Conference of 1996 International Computer Symposium
December 19~21, Kaohsiung, Taiwan, R.O.C.

On Storage Performance for Multi-Stream Video-on-Demand Systems[1]

Chiung-Shien Wu, Gin-Kou Ma, Muh-Rong Yang, and Bao-Shuh P. Lin

Computer and Communication Research Labs.
ITRI, Hsinchu, Taiwan 310, R.O.C.
E-mail : cwu@e0sun3.ccl.itri.org.tw

ABSTRACT – *One of the most important criteria in evaluating a video-on-demand system is the maximum number of video streams that can be simultaneously supported. There are several bottlenecks which limit this capability, such as storage I/O, network bandwidth, and CPU processing overhead. In this paper, we focus on the performance modelling of disk drives in a multi-stream video-on-demand system. We investigate three types of storage subsystems, namely the* single-disk single-bus, *the* multi-disk single-bus *and the* multi-disk multi-bus. *To derive the maximal number of video streams, the data size per I/O request and the required buffer size for each video stream is formulated. The result shows that the number of video streams is limited due to the unavoidable mechanical delay of the disk drives. The performance improvement by using data stripping on an array of disk drives is also shown to be limited since unavoidable overhead still exists. To verify the correctness of the analysis, an inexpensive video-on-demand system based on personal computers and Ethernet is demonstrated. The experimental result is shown to be very close to the analytical result.*

1 Introduction

The design of a video server is emerging as a key technology in the trend toward integrated multimedia services such as video-on-demand, teleshopping, broadcasting and distance learning. One of the important requirements in designing a video server is the ability to support a large number of simultaneous video clients watching the movies stored in the server. To achieve this goal, parallel retrieving of the video storage has been introduced and the switch-based delivery networks were used, for example, the use of RAID (Redundant Array of Inexpensive Disks) systems and the ATM (Asynchronous Transfer Mode) networks. In [1] and [2], a PC-based video server has been designed to support 40 video streams. In [3], and [4], a high performance workstation with large I/O bandwidth is designed as a video server which supports up to 300 video streams. However, there is no direct evidence to show the maximum number of video streams that a single disk drive or a disk array can support. In [5], a simulation-based evaluation on the performance of disk drives was given and the results show that a larger request size would produce more number of video streams.

In this paper, we use an analytical approach to evaluate the performance of the disk drives which are connected to a server using the Small Computer Systems Interface (SCSI) bus. There are three connection types assumed between the server and the storage, namely the *single-disk single-bus*, the *multi-disk single-bus* and the *multi-disk multi-bus*. To derive the maximal number of video streams, the data size per I/O request and the buffer size for each video stream is formulated. The result shows that the number of video streams is limited due to the unavoidable mechanical delay of the disk drives. The performance improvement by using data stripping on an array of disk drives is also shown to be limited since unavoidable overhead still exists, i.e., the disk seeking delay and the SCSI bus contention. To verify the correctness of the analysis, an inexpensive video-on-demand system based on personal computers and Ethernet is demonstrated. The experimental result is shown to be very close to the analytical result.

This paper is organized as follows. A system description and assumptions on disk drives are given in Sections 2 and 3, respectively. The performance analysis on the disk access and the stream buffer size for video-on-demand systems is given in Section 4. A PC-based video-on-demand testbed is introduced in Section 5. Experimental result as well as analytical result for three types of storage-to-server connections are given in Section 6, followed by a short conclusion in Section 7.

2 System Description

An abstract model of a video server for video-on-demand systems is shown in Figure 1. There are n video streams in the system and each stream has a temporary buffer for storing the video data. The storage subsystem consists of the SCSI bus and the disk drives attached to it. A scheduler is created for accessing the video files in the storage subsystem. The scheduler has three functions as listed as follows.

1. Reads a block of video data from the disk drives via

[1]This work is currently supported by the *R.O.C. Ministry of Economic Affairs* under the project No. 37H3100 conducted by ITRI.

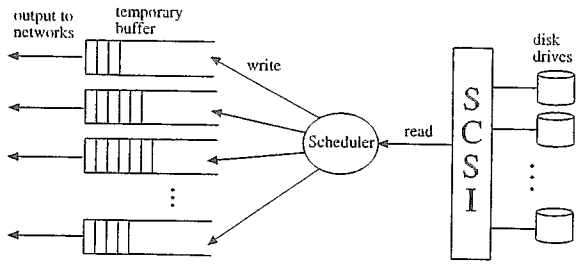

Figure 1: The abstract model of a video server.

SCSI bus,

2. Puts the block of video data into the temporary buffer, and

3. Repeat the above two functions for all the video streams in the system.

Each temporary buffer corresponds to a video stream which is transmitting the video data into the network at a sustainable bit rate, a bits per second. Let the block size that the scheduler reads each time from the disk drive be B bytes. Then, the scheduler is required to perform its functions fast enough such that the sustainable bit rate produced by each video stream can be satisfied. The scheduler performs its functions in a round-robin fashion among the video streams. Since the number of active video streams is n, then the total system requirement is therefore to maximize the value of n.

3 Disk Drive Assumption

The disk drives in our video-on-demand systems are assumed to be connected to the server via the SCSI bus. The host adapter on the bus would first issue a read command to the disk drive if there are requests for data transfer from the user. The disk drive receives the command and then disconnect with the host adapter in order to leave the bus free. Then the disk drive would interpret the command and move the read-write head to the right position, reconnect with the host adapter to gain the bus access permission, and transfer the data to the host adapter. By each command, a block of data can be transferred and the block size is usually independent with the requested size from the user. For a simple analysis, we assume that these two blocks are identical in their size. Each single disk drive transfers their data in a speed which is slower than the maximum bus transfer rate. Usually, this can be improved by a read-ahead operation which preloads the data into the temporary buffer in the disk controller, and if the next request matches with the preloaded data, then the data can be transferred using the maximum bus rate. However, the read-ahead operation is not effective because there are multiple video streams requesting for data transfer and the request pattern is very discontinueous.

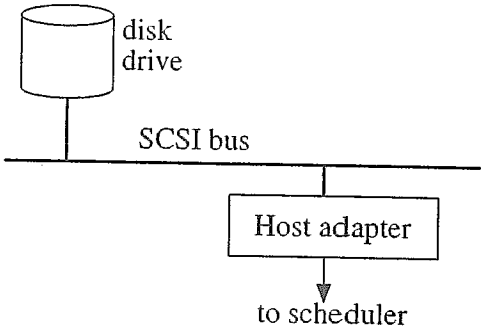

Figure 2: The single-disk single-bus case.

Each read operation unavoidably suffer from a constant delay which may include the time of issuing a command and the interpretation of the command (D1), the disk mechanical delay (D2), and the bus contention delay (D3). After then, the data can be transferred. D1 is a constant delay and it is typically around 1.0 ms. D3 grows as more disk drives are connected to the same SCSI bus. D2 includes several parts, namely the speedup period (S1), the seeking period (S2), the slowdown period (S3), the settle period (S4) and the rotation delay (R1) [6]. Typical values for the above time periods are S1 (< 1.0 ms), S2 (1 - 10 ms), S3 (< 1.0 ms), S4 (1 - 3 ms), and R1 (2 - 5 ms). Let d_m denote the overall waiting time before transferring data. Then, we have

$$d_m = D1 + D2 + D3 \tag{1}$$

where

$$D2 = S1 + S2 + S3 + S4 + R1 \tag{2}$$

In the multi-stream environment, D2 may vary from time to time since D2 is related to the movie location and there are many movies to be accessed. Assume that all the movies are selected randomly. Thus, we may think that D2 is randomly selected in between a minimum value and a maximum value.

There are three individual cases to be discussed, namely the *single-disk single-bus*, the *multi-disk single-bus* and the *multi-disk multi-bus*. We describe these three cases as follows.

3.1 Single-Disk Single-Bus

The case for single-disk single-bus is described in Figure 2, where the only disk drive communicates with the host adapter via a SCSI bus. Each read operation includes two waiting times, namely the initial wait d_m and the data transfer time. As pointed previously, the data is not transferred using the full bus bandwidth, but using a slower disk raw transfer speed, denoted as r_{raw}. And let r_{bus} denote the transfer rate using the full bus bandwidth. This case is the most convenient way to build up an inexpensive video-on-demand system. However, the performance and the storage capacity are limited.

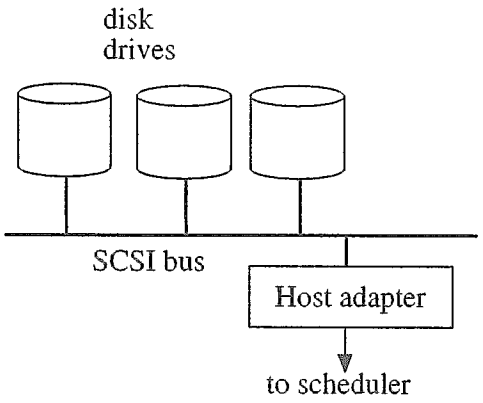

Figure 3: The multi-disk single-bus case.

3.2 Multi-Disk Single-Bus

In Figure 3, the diagram for multiple disk drives connected to one SCSI bus is shown. We assume that a data stripping method is performed on these disk drives, that is, the files are interleaved and stored in these disks. When a read request comes, the host adapter would generate a read command for each disk to send the required portion of the block. Since there are multiple disks, there are opportunities for the disks to transfer data without waiting. For example, one of the disk drives can keep on transferring data while the other drives can perform the command interpretation, seeking, and other necessary tasks. Therefore, delay D2 in this case is assumed to be equal to zero but delay D3 is greater than that for the case of the single disk due to bus contention. However, the data is still transferred in the disk raw transfer speed. This case is a fast and cheap way to extend the storage of the video-on-demand system.

3.3 Multi-Disk Multi-Bus

In this case, there is a disk array controller performing the stripping functions as if there is only one disk on the bus, as shown in Figure 4. Each disk can transmit their data concurrently to the disk controller since there is one dedicated bus for each disk drive. The disk controller has a very large memory space to store the data from every disk drive. Therefore, the disk controller can transmit the data using the full bus bandwidth. However, the disk array suffers from the initial waiting time as the case of single disk since each disk of the array is accessed independently. This case can result in a better performance but the cost is higher.

4 Performance Analysis

4.1 Data Block Size Per I/O

Let r_s denote the data transfer rate ($r_s = r_{bus}$ or r_{raw}).

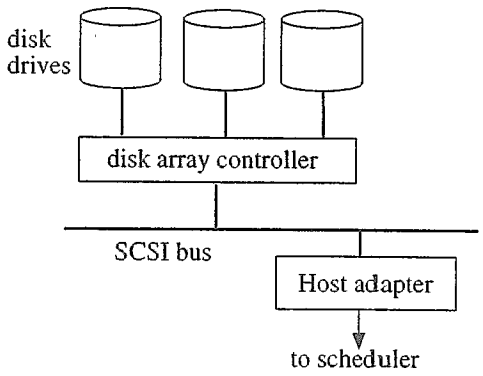

Figure 4: The multi-disk multi-bus case.

Assume that there are n video streams, the relation of the parameters in the previous assumption can be derived as follows. The latency for the scheduler to move a block of video data for one video stream includes the hard disk mechanical delay and the data transfer delay, that equals to

$$d_m + \frac{B}{r_s}.$$

Since there are n video streams, the time interval between moving two blocks of video files for a specific video stream is

$$n \cdot (d_m + \frac{B}{r_s}).$$

During the above time interval, there are B bytes of video data being transferred. Thus, we have the following equation.

$$\frac{8 \cdot B}{n \cdot (d_m + \frac{B}{r_s})} = a \tag{3}$$

Rewrite the above equation, we have

$$B = \frac{a \cdot n \cdot d_m}{8 - \frac{a \cdot n}{r_s}} \tag{4}$$

Defining the bandwidth utilization of the SCSI bus as μ, then we can calculate μ as follows.

$$\mu = \frac{n \cdot a}{8 \cdot r_{bus}} \tag{5}$$

By Equations (3) and (5), we have

$$\mu = \frac{\frac{8 \cdot B}{a \cdot (d_m + \frac{B}{r_s})} \cdot a}{8 \cdot r_{bus}} = \frac{B}{d_m \cdot r_{bus} + B \cdot \frac{r_{bus}}{r_s}} \tag{6}$$

4.2 Buffer Size for Each Stream

In Figure 5, we describe the model for our analysis on the buffer size for each video stream. The video stream is assumed to use a constant bit rate of a bits/sec for sending

out the video data. The length of the buffer, denoted as L, is assumed to be

$$L = l \cdot B \tag{7}$$

The video data are read into the buffer block by block, where the block size is B. Usually, the scheduler would fill up the buffer before sending the first video data into the network. To keep up with the quality of the constant bit rate service, the buffer should not be empty at any time when the movie is played. To obtain the worst-case analysis, we use a very strict assumption. Therefore, we have a property that the time for sending out all the data in the buffer should be greater than the time for the buffer to collect l blocks of data. Let inter-arrival time for two consecutive block be represented by a random variable, i.e., $x_1, x_2, x_3, \cdots, x_l$ for all the blocks in the buffer. Let P_c represent the probability of continuity for each video. Then, we have

$$P_c = Pr\{x_1 + x_2 + \cdots + x_l < \frac{l \cdot B \cdot 8}{a}\}$$

To minimize the buffer required by each stream, we need to find a minimum l such that probability P_c is maximized.

For each random variable x_i, i=*1 to* l, we could divide it into three parts, namely the dynamic mechanical delay, the constant mechanical delay and the data transfer delay. Then, for all i=*1 to* l, we have

$$x_i = y_i + k \cdot n + \frac{B \cdot n}{r_s} \tag{8}$$

where n is the number of streams, k is the constant mechanical delay, and $\frac{B \cdot n}{r_s}$ is the data transfer delay. For all i=*1 to* l, y_i is another random variable which determines the dynamic mechanical delay (DM) and is affected by the disk seeking, rotation, and other dynamic features. For simplicity, we assume that y_i is uniformly distributed between the range of 0 to DM_{max}. Then, we have

$$\begin{aligned} & Pr\{x_1 + x_2 + \cdots + x_l < \frac{l \cdot B \cdot 8}{a}\} \\ = \; & Pr\{y_1 + y_2 + \cdots + y_l < \frac{l \cdot B \cdot 8}{a} \\ & -l \cdot k \cdot n - \frac{l \cdot B \cdot n}{r_s}\} \qquad (9) \\ = \; & Pr\{y_1 + y_2 + \cdots + y_l < l \cdot \Delta\} \qquad (10) \end{aligned}$$

where

$$\Delta = \frac{B \cdot 8}{a} - k \cdot n - \frac{B \cdot n}{r_s} \tag{11}$$

Since we want to serve as many streams as possible, the value of n should be set to a maximum value n_0 such that $\Delta > 0$. That is,

$$n_0 = \max\{n | \frac{B \cdot 8}{a} - k \cdot n - \frac{B \cdot n}{r_s} > 0\}$$

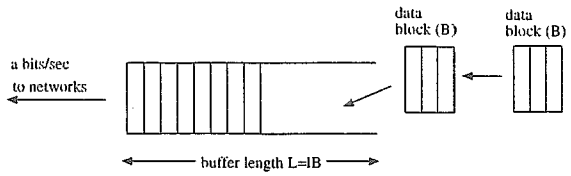

Figure 5: Buffer size model for video-on-demand systems.

Each random variable y_i is in the range $(0, DM_{max}]$. When $DM_{max} > l \cdot \Delta$, Equation (10) may be rewritten into

$$\begin{aligned} & Pr\{y_1 + y_2 + \cdots + y_l < l \cdot \Delta\} \\ = \; & \frac{(l \cdot \Delta)^l}{l!} \cdot \frac{1}{(DM_{max})^l} \qquad (12) \\ = \; & \frac{1}{l!} \cdot (\frac{l \cdot \Delta}{DM_{max}})^l \qquad (13) \end{aligned}$$

The above equation is explained in Figure 6(a), where l is assumed to be 2 for simplicity. The probability is equal to the fraction of the shaded area to the square area bounded by DM_{max}. When $\Delta < DM_{max} < l \cdot \Delta$, Equation (10) may be rewritten into

$$\begin{aligned} & Pr\{y_1 + y_2 + \cdots + y_l < l \cdot \Delta\} \\ = \; & (\frac{(l \cdot \Delta)^l}{l!} - \frac{(l \cdot \Delta - DM_{max})^l \cdot l}{l!}) \\ & \cdot \frac{1}{(DM_{max})^l} \qquad (14) \\ = \; & (\frac{(l \cdot \Delta)^l}{l!} - \frac{(l \cdot \Delta - DM_{max})^l}{(l-1)!}) \cdot \frac{1}{(DM_{max})^l} \qquad (15) \end{aligned}$$

The above equation is explained in Figure 6(b). When $DM_{max} < \Delta$, Equation (10) is always equal to 1, as shown in Figure 6(c).

In either case, we need to find an l such that Equation (10) approach 1.

5 PC-based Video-on-Demand Testbed

An experimental video-on-demand testbed is currently built up in CCL/ITRI, as shown in Figure 7. The platforms for both client and server are simply personal computers. We select an Acer Altos 9000 PC with Intel Pentium 100 CPU as the platform. For video server, we use Microsoft Windows NT as the operating system in order to serve the multiple video streams. The scheduler is implemented in the server and we select 5400 r.p.m disk drives for the storage subsystem. For the client, we use Microsoft Windows 3.1 with an MPEG-1 decoder attached. The clients are connected to the server via Ethernet. We successfully demonstrate 4 clients on the same Ethernet watching the movies from the server. The video files are transmitted using the TCP/IP protocol. At the client site, the shareware Trumpet 2.0B is selected as the

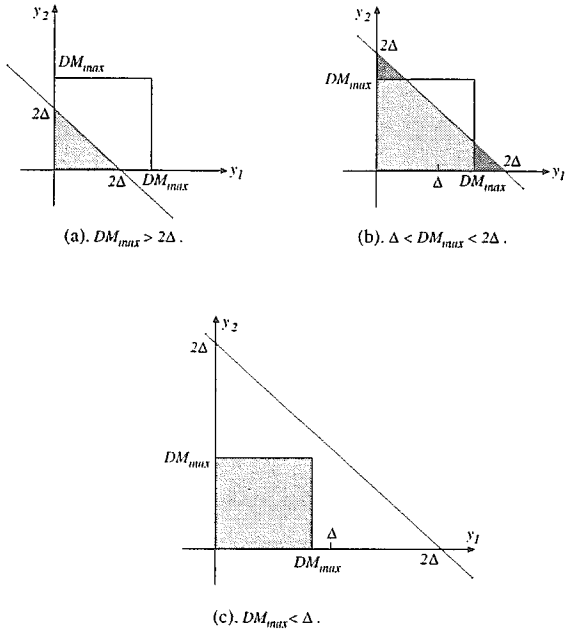

Figure 6: The relation of DM_{max} and $l \cdot \Delta$.

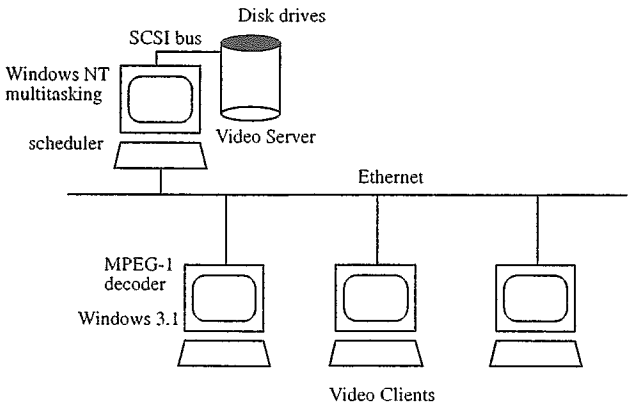

Figure 7: PC-based experimental video-on-demand testbed.

TCP/IP protocol stacks. All the components used are very cheap and available very easily.

We also observe the performance of the scheduler in response to the analytical result. We store 32 large movie files in the disk drives and each file is about 50 Mbytes in length. Each stream is generated by randomly selecting the movies from the disk drive.

6 Experimental Results

6.1 The Block Size per I/O

6.1.1 Single-Disk Single-Bus

For the single disk case, we assume the following values for the required parameters.

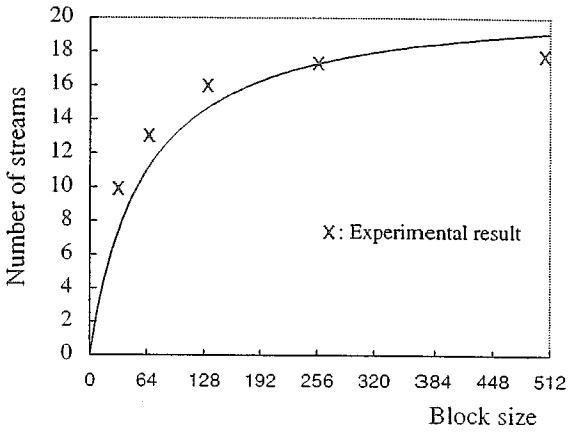

Figure 8: Performance of the single-disk single-bus subsystem.

1. $a = 1.5Mbps$ (for MPEG-1 streams),
2. $d_m = D1(1ms) + D3(2ms) + S1(0.5ms) + S2(6ms) + S3(0.5ms) + S4(2ms) + R1(3ms) = 15ms$.
3. $r_{raw} = 4$ M Bytes/sec.
4. $r_{bus} = 10$ M Bytes/sec.

We plot that curve of block size B with respect to n in Figure 8. In this case, a larger size of B, i.e., $512Kbytes$, could produce a larger number of video streams. The experimental result is also shown in Figure 8, where the maximum block size B is $512Kbytes$. When $B > 128KBytes$, n remains in a steady value of 18. The experimental result is very close to the analytical result and this implies the correctness of the proposed analysis.

6.1.2 Multi-Disk Single-Bus

In this case, we adopt the same assumption as the single disk case for the values of the required parameters except that of d_m, since we assume that D2 is zero and D3 must be larger due to the bus contention. Let D3 be twice as large as the case for single disk. Then, we have $d_m = 5ms$. In Figure 9, the curve for block size B and the number of streams n is plotted. The result is slightly better than that for single disk case. However, the difference is not much when the block size is greater than $256Kbytes$. The experimental result is also shown in Figure 9 indicated by a sequence of cross points. The behavior of the experimental result is very close to the result for the single disk case as well as the analytical result.

6.1.3 The Multi-Disk Multi-Bus Case

We adopt the same assumption as the single disk case for the values of the required parameters. However, in this case the disk array controller can transmit its data at the full speed

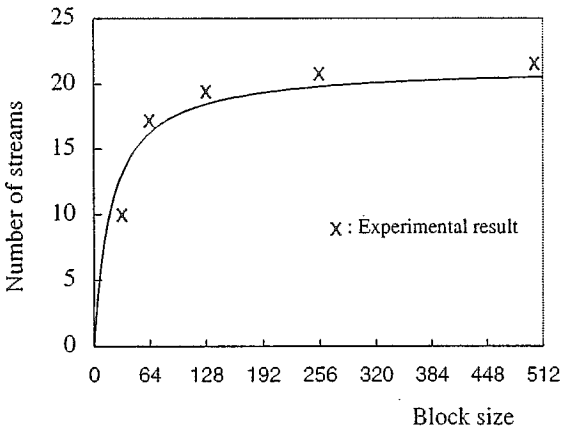

Figure 9: Performance of the multi-disk single-bus subsystem.

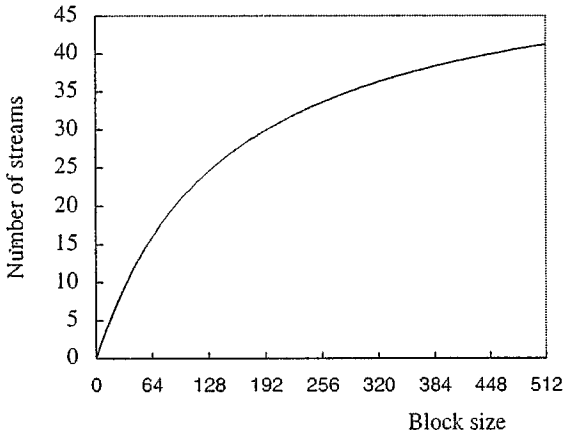

Figure 10: Performance of the multi-disk multi-bus subsystem.

of the bus. Thus, r_s is equal to r_{bus} in this case. In Figure 10, the curve for block size B and the number of streams n is plotted. The result is better than that for single and multiple disks case. When $B = 512KBytes$, the number of streams can be up to 40. For n to remain in a steady performance, the block size should be larger than that in the case for single or multiple disks case, i.e., $512KBytes$ or $1024KBytes$. We didn't show the experimental result for this case since we currently can not find a disk array of this type.

6.2 Bandwidth Utilization

The bandwidth utilization according to Equation (6) for the three cases is shown in Figure 11. For the first two cases (single and multiple disks), the utilization is not high because the disks are transferring their data in the raw disk speed. The case for multi-disk multi-bus gains a higher utilization because the maximum speed of the bus is used. Furthermore, the multi-disk multi-bus case should come with a larger block

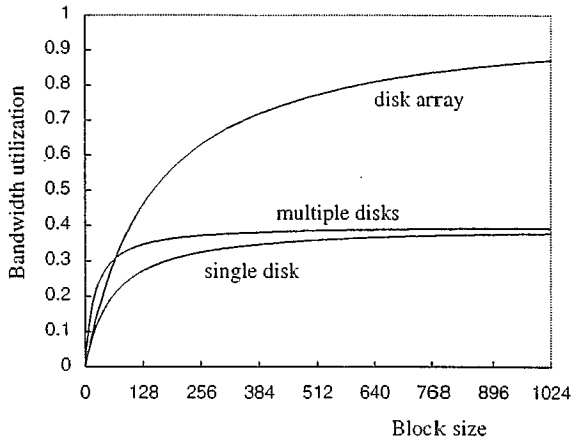

Figure 11: Bandwidth utilization of the three storage subsystem.

size, i.e., $1024KBytes$ or more, in order to gain a steady performance. For the other two cases, the performance is steady when the block size is around $128KBytes$.

6.3 Buffer Size for Each Stream

To find the minimum buffer length, we assume the following values for the required parameters.

1. $B = 32, 64K$bytes,
2. $n = 10$, (for $B = 32K$) and $n = 14$, (for $B = 64K$),
3. $a = 1.5Mbps$ (for MPEG-1 streams),
4. $r_s = 4$ M Bytes/sec.
5. $k = 8ms$,
6. $DM_{max} = 5ms$ to $35ms$,

In Figure 12, we plot the result of the buffer length by using $B = 32Kbytes$, $n = 10$. In this case, $\Delta = 0.010667$. When $DM_{max} < 25ms$, probability P_c approaches 1 when l is greater than 15. Therefore, in this case, the buffer size can be set to $15 \cdot 32K = 480K$bytes in order to guarantee a continous transmission under the restriction that the variation of the dynamic disk seeking time is smaller than $25ms$. The other case by using $B = 64Kbytes$ and $n = 14$ is similar to the first case, as shown in Figure 13. However, DM_{max} is limited to be within $10ms$ in order to have a maximum probability of continuity. To let $P_c = 1$, l should be greater than 5 when $DM_{max} = 10ms$. When $DM_{max} > 10ms$, there is no chance for P_c equal to 1. An important observation is that a smaller size of block B is better for guaranteeing the stream continuity. From the analysis in Section 4.1, we obtain that a larger size of request block is helpful in supporting more number of streams. Therefore, it is possible to find the best size of B to support a maximum number of streams and each

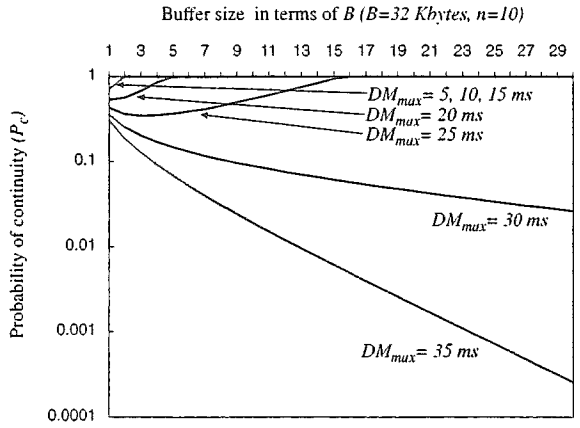

Figure 12: Buffer length v.s. Probability of continuity P_c (B=32 kbytes).

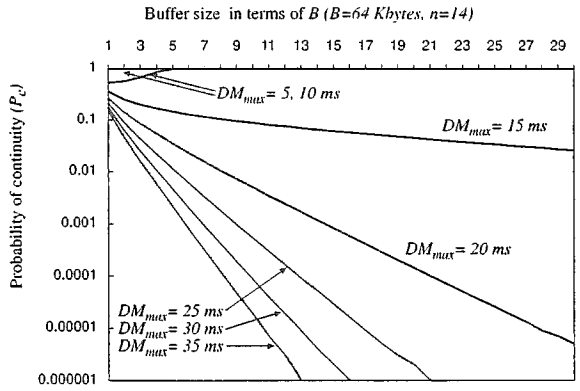

Figure 13: Buffer length v.s. Probability of continuity P_c (B=64 kbytes).

stream is transmitted continuously. For example, to select $B = 64K$bytes and $l = 5$ can have $n = 14$, while selecting $B = 32K$bytes and $l = 15$ can only have $n = 10$.

7 Conclusion

In this paper, a storage performance evaluation on the number of streams in a multi-stream video-on-demand system is provided. We assume three cases for the storage subsystem which are connected with the video-on-demand system. For extending the storage in a video-on-demand system, multiple disks can be connected on a SCSI bus to form an array. However, the improvement on performance is not significant due to some deterministic overhead of the SCSI bus. A more efficient method to extend the storage is to use a dedicated array controller on the bus, where data can be retrieved concurrently from the multiple disks. However, the required block size per I/O operation should be large, i.e., $1024KBytes$ or above, in order to obtain a steady performance. Furthermore, to guarantee the continuity of each stream, a buffer space is necessary for saving the video data before sending them into the networks. From our analysis, a smaller size of each I/O block is better for the continuity of each stream. By compromising these two factors, a best block size of each I/O operation to support the maximum number of streams in the VoD system can be obtained.

References

[1] F. A. Tobagi, and J. Pang, "StarWorks - A Video Application Server", *IEEE COMPCON Spring'93*, pp. 4-11.

[2] W. Tseng, and J. Huang, "A High Performance Video Server For Karaoke Systems", *IEEE Transations on Consumer Electronics*, Vol. 40, No. 3, Aug., 1994, pp. 329-336.

[3] J. Hsieh, M. Lin, J. C.L. Liu, and D. H.C. Du, "Performance of a Mass Storage System for Video-On-Demand", to appear in *Journal of Parallel and Distributed Computing*, 1995.

[4] J. Hsieh, M. Lin, J. C.L. Liu, D. H.C. Du, and T. M. Rewart, "Performance of a Mass Storage System for Video-On-Demand", *IEEE Infocom'95*, Boston, MA, 1995, pp. 771-778.

[5] A.L. Narasimha Reddy and James C. Wyllie, "I/O Issues in a Multimedia System", *IEEE Computer*, March 1994, pp. 69-74.

[6] Chris Ruemmler and John Wilkes, "An Introduction to Disk Drive Modeling", *IEEE Computer*, March 1994, pp. 17-28.