Impact of Buffer Sharing in Multiple Disk Video Server Architectures

Jamel Gafsi, Ernst Biersack
phone: +33 4 93 00 26 89, fax: +33 4 93 00 26 27, email: (gafsi,erbi)@eurecom.fr
Institut EURECOM
2229, route des Cretes B.P. 193, 06904 Sophia Antipolis Cedex, FRANCE

Abstract

A video server allows to deliver multiple video streams to different clients. To provide the required amount of storage and bandwidth, a video server must contain a large number of disks. Since the retrieval rate of the disk and the consumption rate of the clients differ, the data retrieved from disk need to be temporarily stored in main memory, which is an important cost factor in a video server. Video data must be retrieved from disks in such a way that neither buffer starvation nor overflow occurs for all concurrent video streams. In this paper we calculate the required buffer for the GSS scheduling algorithm for multiple streams retrieved from multiple disk server nodes. We prove analytically that shared buffer management reduces, in comparison with the dedicated buffer management, the required buffer by up to 50%.

1 Introduction

Large scale video server can contain hundreds of disks and support thousands of concurrent clients. Because of the synchronous nature of multimedia streams and the asynchronous nature of data retrieval, main memory is needed where video information is stored before transmission onto the network. Since main memory is very expensive, a cost effective video server architecture must minimize the total amount of main memory. In this paper, we calculate the buffer requirement for constant bit rate video streams for the case of single or multiple streams and single or multiple disks. The calculations are analytical and assume the worst case.

2 Video Server Architecture and Environment Parameters

An important issue designing a large scale video server is its scalability. Such a video sever comprises many server nodes (server array). Each node contains a set of disks (disk array). To achieve perfect load balancing, every video is stored over all server nodes. To retrieve video data from the video server, all server nodes send substreams of the required video stream in a fixed order to the client. For more details about the server array design see [BB96].
To satisfy the increasing demand for bandwidth and admit more concurrent clients, new server nodes have to be added into the video server. Thus, the total number of disks become higher. Further, when we distribute a video over all available disks of the server, the number of seek events of video data inside the disks increases as the total number of disks increases. Consequently, the seek times, which are added to the latency, increase too. Additionally we need very large buffer space to support all substreams retrieved from all disks. To overcome these problems, we propose to divide the whole video server into G independent groups, called **retrieval groups**. Figure 1 shows an example of the proposed video server architecture that contains N nodes with respectively a set of disks. Disks of a retrieval group g ($g \in [1..G]$) belong to different nodes. In this way, the load balancing

in terms of storage volume and required bandwidth is conserved over all nodes of the server array. During a time interval called **service round**, each client is served from a single retrieval group. During the next service round, the next retrieval group has to serve this client. We assume round robin to order the retrieval groups. It should be mentioned that the amount of buffer required for one video stream (one client) does not depend on the total number of disks maintaining in the server, but only on the retrieval group size (number of disks inside a retrieval group).

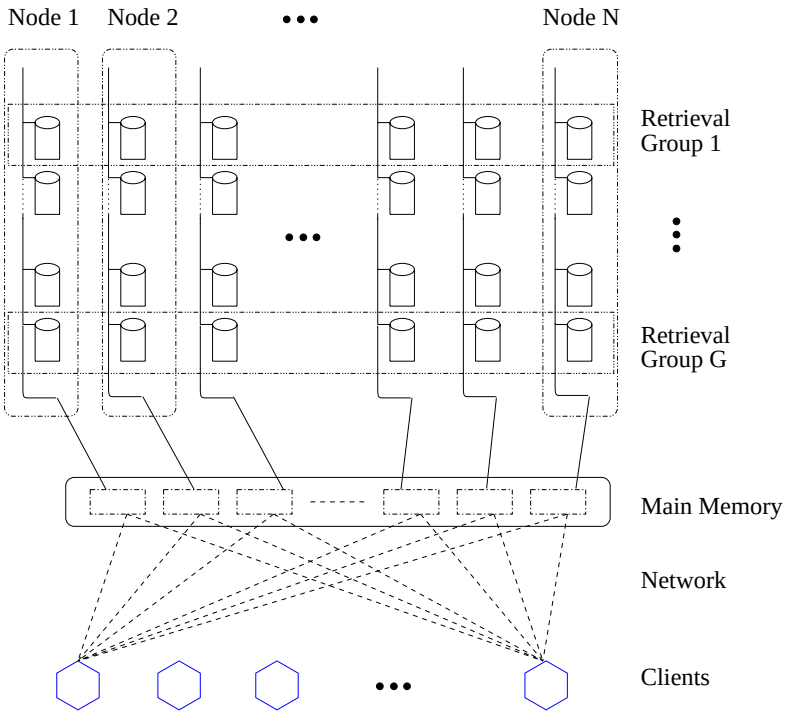


Figure 1: Video Server Architecture.

We introduce now all parameters needed in this paper to calculate the total buffer by the server array that supports many concurrent clients.

- D: The total number of disks in the whole server array.
- d: Denotes a single disk, where $d \in [1..D]$
- N: The total number of server nodes in the server array.
- n: Denotes the server node n and $n \in [1..N]$
- D_n: Number of disks that belong to the server node n. We assume that our video server contains only homogeneous server nodes and the whole number of disks is a multiple of the number of nodes. Thus:
  $\forall n \in [1..N] : D_n = D/N$
- S_s: Denotes the s-th video stream. the maximum value of s depends on the system throughput (the number of admitted streams or clients).
- b_{dr}: The size of the disk retrieval block. A disk retrieval block is the amount of data retrieved for one video (client) from a single disk during a service round. Its value is a multiple of the physical disk block and depends on the striping method used (Video-Narrow/Wide-Striping and Segment-Narrow/Wide-Striping).

- b_{ru}: The size of the retrieval unit. The retrieval unit is the whole amount of data retrieved for one video (client) during one service round. It is a multiple of the disk retrieval block and can be read either from all disks or only from a group of disks of the server. This value influences directly the amount of buffer needed for one client. When it increases, the required buffer size increases too. Thus, optimizing the buffer requirement consists on minimizing b_{ru}. b_{ru} is a multiple of b_{dr}. The later discussion will more focus this issue.

- G: Number of Retrieval Groups in the server array: Each retrieval unit is read from a set of disks during a service round. This set of disks is called the retrieval group.

- g: Denotes the Retrieval Group number: $g \in [1..G]$

- D_g: Number of disks that belong to the Retrieval Group g. It also depends on the striping policy and indicates where the Retrieval Unit is to be read from (during one service round). Like for nodes, we assume that we only deal with homogeneous Retrieval Groups and D is a multiple of G. Thus the Retrieval Group size is the same for all Retrieval Groups: $\forall g \in [1..G] : D_g = \frac{D}{G}$. We distinguish the following cases:

  - $D_g = 1$: The whole Retrieval Unit is stored on a single disk. This is the case of the Independent Retrieval.
  - $D_g = D$: The Retrieval Unit is distributed over all disks of the server: the Dependant Retrieval Technique.
  - $1 < D_g < D$: The Retrieval Unit is distributed over a subset of disks of the server (Dependant Retrieval).

- Q_d: Is the maximal number of clients that can be simultaneously served by the disk d.

- Q_g: Is the maximal number of clients that can be simultaneously served by the Retrieval Group g.

- Q: Is the whole number of clients that can be simultaneously served by the video server. The variable s which denotes the Video Stream S_s has the characteristic: $s \in [1..Q]$ When the server has only one group (g = 1), then we have: $Q_g = Q$

- τ: Is the service round size. In this paper, we assume that the service round is: $\tau = \frac{b_{ru}}{r_p}$, r_p is the playback rate of every video stream.

  During τ, up to Q_d disk retrieval blocks belonging to Q_d Video Streams (clients) are read from a single disk and put in the server buffer. For the CBR coding algorithm, every service round τ can be partitioned into Q_d equal *sub-service-rounds* Δ as: $\tau = Q_d * \Delta$.

3 Buffer Requirement

Because of the asynchronous nature of data retrieval from the disk, the video server needs to store the retrieved video information in a buffer before sending it to the sender through the network. Designing a cost effective video server consists in admitting the maximal number of concurrent video streams for a given amount of resources. Optimizing the required buffer is a potential factor to reduce server costs. This section gives detailed buffer calculations. We are the first who exactly calculate the buffer requirement for a multiple disk server array, where each disk supports multiple streams and compare the *dedicated* buffer management and the *shared* buffer management strategies. We assume round based scheduling algorithms and prove analytically that the buffer requirement strongly depends on the scheduling policy (Round Robin, SCAN or GSS). Section 3.1 introduces

the exclusive and shared buffer management strategies. In section 3.2 we classify round based scheduling algorithms into order- (OCS), non-order- (NOCS) and semi-order-conserving (SOSC) algorithms. Since the paper space is limited, we only analyze the buffer requirement for SOCS algorithms in section 3.3.

3.1 Exclusive and Shared Buffer Management

In the following, we list the functions needed to compute the buffer requirement regarding multiple streams and multiple disks. We make the difference between the two different buffer management strategies: the **exclusive** and the **shared** buffer management. The exclusive buffer management does not take into account the dependency of multiple video streams or multiple disks and assumes that there is a dedicated buffer for every disk and every stream. The shared buffer management assigns to all video streams a single buffer. We use the suffix $_{excl}$ for the exclusive buffer management and the suffix $_{shrd}$ for the shared buffer management. Furthermore, we introduce the corresponding formulas for the different functions.

- $B_{excl}(s,d)$: exclusive buffer needed to support the video stream s served by the disk d.
- $B_{shrd}(s,d)$: shared buffer needed to support the video stream s served by the disk d. Because we only consider a single video stream, there can be no buffer sharing. Thus:

$$B_{shrd}(s,d) = B_{excl}(s,d) \tag{1}$$

- $B_{excl}(d)$: exclusive buffer needed to support the Q_d video streams served by the disk d:

$$B_{excl}(d) = \sum_{s=1}^{Q_d} B_{excl}(s,d) = Q_d \cdot B_{excl}(s,d) \tag{2}$$

- $B_{shrd}(d)$: shared buffer needed to support the Q_d video streams served by the disk d. We consider the worst case buffer situation:

$$B_{shrd}(d) = max(\sum_{s=1}^{Q_d} B_{shrd}(s,d)) \tag{3}$$

- $B_{excl}(s,D_g)$: exclusive buffer expected to support the video stream s served by the server. (we take the general case, where a video stream only needs D_g of the D disks of the server:

$$B_{excl}(s,D_g) = D_g \cdot B_{excl}(s,d) \tag{4}$$

- $B_{shrd}(s,D_g)$: shared buffer needed to support the video stream s served by the server. (we take the general case, where a video stream only needs D_g of the D disks of the server. There is no buffer sharing possible, when we assume a single stream:

$$B_{shrd}(s,D_g) = B_{excl}(s,D_g) = D_g \cdot B_{shrd}(s,d) \tag{5}$$

- $B_{excl}(D_g)$: exclusive buffer needed to support D_g disks. D_g can take values between 1 and D. In this case, there are Q_g video streams that can be supported. When $D_g = D$ then $Q_g = Q$:

$$B_{excl}(D_g) = (\sum_{d=1}^{D_g}(\sum_{s=1}^{Q_g} B_{excl}(s,d))) = (\sum_{s=1}^{Q_g}(\sum_{d=1}^{D_g} B_{excl}(s,d))) = \sum_{s=1}^{Q_g} B_{excl}(s,D_g) \tag{6}$$

- $B_{shrd}(D_g)$: shared buffer needed to support D_g disks. D_g can take values between 1 and D. In this case, there are Q_g video streams that can be supported. When $D_g = D$ then $Q_g = Q$.

$$B_{shrd}(D_g) = max(\sum_{d=1}^{D_g}(\sum_{s=1}^{Q_g} B_{shrd}(s,d))) = max(\sum_{s=1}^{Q_g}(\sum_{d=1}^{D_g} B_{shrd}(s,d))) \quad (7)$$

To calculate the total buffer requirement for the whole video server, we use $D = D_g \cdot G$. Thus the amount of buffer needed to support the D disks of the server is a multiple of the buffer requirement for a retrieval group (D_g disks). Let $B_{excl}(D)$ and $B_{shrd}(D)$ denote the buffer values of the whole video server respectively for the exclusive and the shared buffer model. It is important to know that the amount of buffer needed for the whole video server is the sum of the amounts of buffer needed to support every retrieval group independently from the others.

$$B_{excl}(D) = \sum_{g=1}^{G} B_{excl}(D_g) = G \cdot B_{excl}(D_g) \quad (8)$$

$$B_{shrd}(D) = \sum_{g=1}^{G} B_{shrd}(D_g) = G \cdot B_{shrd}(D_g) \quad (9)$$

3.2 Order and Non-order Conserving Scheduling

The required amount of buffer depends on the number of clients that can be simultaneously served. It also depends on the scheduling algorithm that determines *how* data is retrieved from the disks. We classify the existing round-based scheduling algorithms depending on whether the *order* in which the data is read for the different streams is maintained from round to round. We distinguish the following classes:

- Order Conserving Scheduling algorithms (*OCS*): With OCS, the retrieval of video streams happens in a round robin fashion respecting a *fixed order* for all service rounds. An example of this class is the *Round Robin* scheduling algorithm [NY94].
- Non-Order Conserving Scheduling algorithms (*NOCS*): With NOCS, concurrent video streams are retrieved in different orders during different service rounds. This allows to minimize the seek times that takes the physical disk head and therefore to increase the maximal number of admitted streams. *SCAN* is a NOCS algorithm [Mou96].
- Semi-Order Conserving Scheduling (*SOCS*): NOCS algorithms minimize the latency overhead. OCS optimize the buffer because of the fixed order of data retrieval. A combination of NOCS and OCS results in the SOCS class. In [YCK93] the GSS algorithm was proposed as a typical example of a SOCS algorithm.

3.3 Buffer Requirement for SOCS Algorithms

We consider GSS as example of a SOCS algorithm. With GSS, the order of block retrieval from disks is not conserved between single video streams, but between sets of them. The Q_d blocks are grouped into k sets and the order, in which the sets are set to the buffer is constant for all service rounds. The order of video stream belonging to the same set can change inside this set during different service rounds. A service round is therefore partitioned in k equal round-groups. We assume that Q_d is a multiple of k. Let have $Q_d = m \cdot k$ where m is an integer (compare with [TPBG93]).

Dedicated SOCS-buffer requirement for the video stream s served from the disk d Let us compute the maximal amount of buffer needed to serve one video stream retrieved from one disk (figure 2(a)). We analyze two consecutive service rounds in terms of the arrival and the consumption of multiple video streams to and from the buffer. Assume that t_{min} is the minimal time between two arrivals to the buffer serving the same client. Thus we have: $t_{min} = (\frac{(k-1)}{k}) \cdot \tau + (\frac{1}{Q_d}) \cdot \tau = (1 + \frac{1}{Q_d} - \frac{1}{k}) \cdot \tau$
We compute the required buffer by computing the amount of data contained in the buffer at the worst case:
After t_{min}, the buffer content equals $b_{dr} - (1 + \frac{1}{Q_d} - \frac{1}{k}) \cdot b_{dr} = (\frac{1}{k} - \frac{1}{Q_d}) \cdot b_{dr}$.(Up to t_{min}, $(1 + \frac{1}{Q_d} - \frac{1}{k}) \cdot b_{dr}$ was consumed from the buffer).
After $t_{min} + \frac{\tau}{Q_d}$: the buffer content equals $b_{dr} + (\frac{1}{k} - \frac{1}{Q_d}) \cdot b_{dr} - \frac{1}{Q_d} \cdot b_{dr}$. We finally derive the maximal buffer for a single video stream served from a single disk as follows:

$$B_{excl}(s,d) = b_{dr} \cdot (1 + \frac{1}{k} - \frac{2}{Q_d}) = B_{shrd}(s,d) \tag{10}$$

Dedicated SOCS-buffer requirement for the Q_d video streams served from the disk d

$$B_{excl}(d) = \sum_{s=1}^{Q_d} B_{excl}(s,d) = Q_d \cdot b_{dr} \cdot (1 + \frac{1}{k} - \frac{2}{Q_d}) \tag{11}$$

Dedicated SOCS-buffer requirement for the video stream s served from D_g disks

$$B_{excl}(s,D_g) = D_g \cdot B_{excl}(s,d) = D_g \cdot b_{dr} \cdot (1 + \frac{1}{k} - \frac{2}{Q_d}) \tag{12}$$

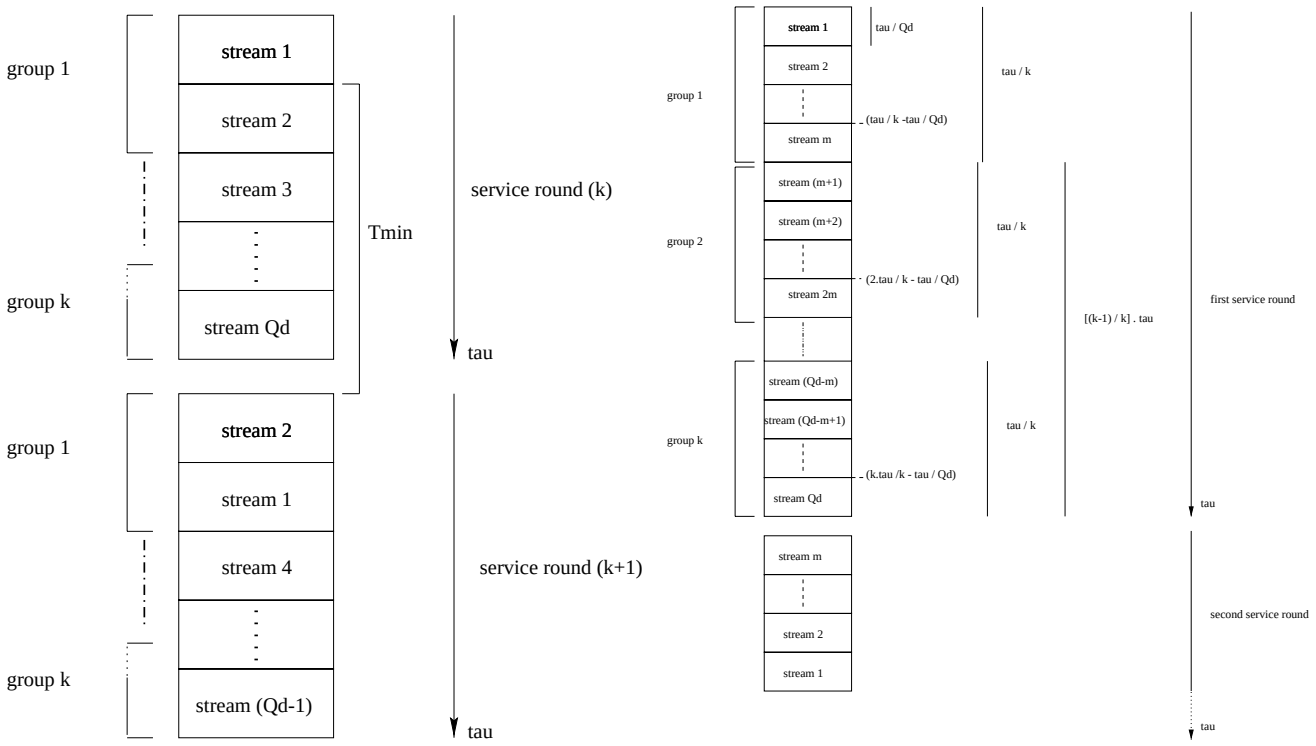
Dedicated SOCS-buffer requirement for Q_g video streams served from D_g disks

$$B_{excl}(D_g) = D_g \cdot \sum_{s=1}^{Q_g} B_{max}(s,d) = D_g^2 \cdot Q_d \cdot b_{dr} \cdot (1 + \frac{1}{k} - \frac{2}{Q_d}) \tag{13}$$

Dedicated SOCS-buffer requirement for all Q video streams served from all D disks

$$B_{excl}(D) = G \cdot B_{excl}(D_g) = D \cdot D_g \cdot Q_d \cdot b_{dr} \cdot (1 + \frac{1}{k} - \frac{2}{Q_d}) \tag{14}$$

Shared SOCS-buffer requirement for the Q_d video streams served from the disk d : To compute the amount of buffer needed, we must take into account that buffer starvation must be avoided. With the SOCS algorithm, i.e. GSS, the retrieval order only changes inside a group. In the following, we calculate the worst case amount of buffer needed while using the shared buffer management (figure 2(b)). Inside a group of retrieved streams, the order of streams is not conserved during two consecutive service rounds. Let us consider the first service round. Because we do not know the order of retrieval of streams during the next service round, we assume for each of they the worst case (retrieval of the stream as late as possible during the next service round). To be sure that there is no buffer starvation, the consumption time must be delayed. The additional delay for a stream depends on its retrieval position inside a group. Let denote T_i the earliest time, at which the consumption of all streams of a group can begin.
For the first group: $T_1 = (\frac{\tau}{k}) - (\frac{\tau}{Q_d})$, for the second group: $T_2 = (\frac{2 \cdot \tau}{k}) - (\frac{\tau}{Q_d})$ and for the k^{th} group: $T_k = (\frac{k \cdot \tau}{k}) - (\frac{\tau}{Q_d})$



(a) A worst case example of an SOCS buffer occupancy scenario

(b) The Retrieval of Video Streams from a Single Disk for GSS during a service round

Figure 2: Data Retrieval for GSS.

Now, lets compute the amount of data contained in the buffer at the time τ (after a service round): This is the difference between the amount of data written into the buffer from the disks (B_{write}) and the amount of data read from the buffer and sent to the client (B_{read}).
It is obvious that $B_{write} = Q_d \cdot b_{dr}$
The calculation of the amount of data consumed B_{read} up-to τ is more complicated. To derive this, we calculate first for every group the amount of data consume:

- For group 1:
  - The consumption duration of the first m streams is $\tau - (\frac{m-1}{Q_d}) \cdot \tau = \tau \cdot (1 - \frac{(m-1)}{Q_d})$
  - The amount of data consumed is $m \cdot (1 - \frac{(m-1)}{Q_d}) \cdot b_{dr}$
- For group 2:
  - The consumption duration of the second m streams is $\tau - (\frac{2 \cdot m-1}{Q_d}) \cdot \tau = \tau \cdot (1 - \frac{(2 \cdot m-1)}{Q_d})$
  - The amount of data consumed is $m \cdot (1 - \frac{(2 \cdot m-1)}{Q_d}) \cdot b_{dr}$
- For group k:
  - The consumption duration of the k^{th} group is $\tau - (\frac{2 \cdot m-1}{Q_d}) \cdot \tau = \tau \cdot (1 - \frac{(k \cdot m-1)}{Q_d}) = \tau \cdot (1 - \frac{(Q_d-1)}{Q_d})$

– The amount of data consumed is $m \cdot (1 - \frac{(k \cdot m - 1)}{Q_d}) \cdot b_{dr} = m \cdot (1 - \frac{(Q_d - 1)}{Q_d}) \cdot b_{dr}$

This allows as to calculate B_{read}: $B_{read} = \sum_{j=1}^{k} m \cdot b_{dr} \cdot (1 - \frac{(j \cdot m - 1)}{Q_d})$
Consequently we derive the amount of data contained in the buffer at time τ. We use the equation: $Q_d = m \cdot k$

$$\begin{aligned}
B_{shrd}(d) &= B_{write} - B_{read} = Q_d \cdot b_{dr} - \sum_{j=1}^{k} m \cdot b_{dr} \cdot (1 - \frac{(j \cdot m - 1)}{Q_d}) \\
&= Q_d \cdot b_{dr} - m \cdot b_{dr} \cdot \sum_{j=1}^{k} \frac{(Q_d + 1) - j \cdot m}{Q_d} \\
&= Q_d \cdot b_{dr} - m \cdot b_{dr} \cdot (\frac{k \cdot (Q_d + 1)}{Q_d} - (\frac{m}{Q_d} \cdot \frac{k \cdot (k+1)}{2})) \\
&= Q_d \cdot b_{dr} - \frac{Q_d \cdot bdr}{k} \cdot ((k \cdot \frac{(Q_d + 1)}{Q_d}) - \frac{(k+1)}{2}) \\
&= Q_d \cdot b_{dr} - (b_{dr} \cdot (Q_d + 1)) + (\frac{Q_d \cdot b_{dr} \cdot (k+1)}{2 \cdot k}) \\
&= (\frac{b_{dr} \cdot Q_d}{2}) + (\frac{b_{dr} \cdot Q_d}{2 \cdot k}) - b_{dr} = Q_d \cdot b_{dr} \cdot (\frac{1}{2} + \frac{1}{2 \cdot k} - \frac{1}{Q_d}) \\
&= \frac{B_{excl}(d)}{2}
\end{aligned} \tag{15}$$

Shared SOCS-buffer requirement for Q_g video streams served from D_g disks

$$\begin{aligned}
B_{shrd}(D_g) &= max(\sum_{d=1}^{D_g} (\sum_{s=1}^{Q_g} B_{shrd}(s,d))) = max(\sum_{d=1}^{D_g} (D_g \cdot B_{shrd}(d))) \\
&= D_g \cdot (D_g \cdot B_{shrd}(d)) = D_g^2 \cdot Q_d \cdot b_{dr} \cdot (\frac{1}{2} + \frac{1}{2 \cdot k} - \frac{1}{Q_d}) \\
&= \frac{B_{excl}(D_g)}{2}
\end{aligned} \tag{16}$$

Shared SOCS-buffer requirement for all Q video streams served from all D disks

$$\begin{aligned}
B_{shrd}(D) &= G \cdot B_{shrd}(D_g) = D \cdot D_g \cdot Q_d \cdot b_{dr} \cdot (\frac{1}{2} + \frac{1}{2 \cdot k} - \frac{1}{Q_d}) \\
&= \frac{B_{excl}(D)}{2}
\end{aligned} \tag{17}$$

4 Conclusion

In this work:

- We calculated the buffer requirement for multiple disk server arrays to support multiple streams.
- The buffer calculations assume that buffer starvation never occurs. We compared the exclusive and shared buffer management strategies and proved analytically that buffer sharing reduces the buffer requirement by 50 % for the GSS algorithm.

We considered also the Round Robin and SCAN algorithms and proved the benefit of buffer sharing compared to dedicated buffer management.

References

[BB96] Chr. Bernhardt and E. W. Biersack. The server array: A scalable video server architecture. In W. Effelsberg, O. Spaniol, A. Danthine, and D. Ferrari, editors, *High-Speed Networking for Multimedia Applications*. Kluwer Publishers, Amsterdam, The Netherlands, March 1996.

[Mou96] Antoine Mourad. Issues in the design of a storage server for video-on-demand. *Multimedia Systems*, 4(2):70–86, 1996.

[NY94] R. T. Ng and J. Yang. Maximizing buffer and disk utilization for news-on-demand. In *Proc. 20th VLDB*, pages 451–462, Santiago, Chile, 1994.

[TPBG93] Fouad A. Tobagi, Joseph Pang, Randall Baird, and Mark Gang. Streaming raid – a disk array management system for video files. In *First International Conference on Multimedia*, Anaheim, California, August 1993.

[YCK93] Philip S. Yu, Mong-Song Chen, and Dilip D. Kandlur. Grouped sweeping scheduling for dasd-based multimedia storage management. *ACM Multimedia Systems*, 1(3):99–108, 1993.