Mitigating Impact of Starting New Session in Zoned Disk

Youjip Won
ECE Division, Hanyang University
Seoul, Korea zip: 133-791
yjwon@ece.hanyang.ac.kr

Kyeongsun Cho
ECE Division, Hanyang University
Seoul, Korea zip: 133-791
goodsun@ece.hanyang.ac.kr

Seung-Min Park
Computer SW Tech. Lab., ETRI
Daejeon, KOREA zip: 305-350
minpark@etri.re.kr

Abstract
Cycle based disk scheduling approach is widely used to satisfy the timing constraints of the multimedia data retrieval. When new service request arrives at the server, the length of cycle needs to be extended to accommodate the new service session. Under legacy cycle based disk scheduling paradigm, the amount of blocks fetched for a session in each cycle becomes insufficient when the cycle is extended to accommodate a new stream and subsequently some of the ongoing sessions suffer from jitter. In this article, we present a technique called "prebuffering" which makes the streaming server resilient to the cycle extension. Our model is based upon the zone based disk.

Keywords
Disk Scheduling, Jitter, Multimedia, Streaming, Zoned Disk

1. Introduction
As a means to supply the data blocks to individual sessions guaranteeing a certain amount of disk bandwidth, cycle based disk scheduling techniques have been proposed[1,2,3,4,7]. The amount of data blocks retrieved for individual session in a cycle should be sufficient for the respective end user's playback for the length of cycle. When the new service request arrives, the length of the cycle needs to be properly extended to accommodate the new streaming session. However, extension of a cycle causes jitter to some of the users because the amount of data blocks which have been read from the disk in each cycle is not sufficient to survive the newly extended cycle. This phenomenon is actually found in a number of commercially available streaming servers. In this article, we propose a technique, "prebuffering", which enables to absorb this jittery situation by preloading a certain amount of data blocks prior to starting service. The important issue here is to obtain the appropriate prebuffer size as well as the prebuffering strategy to minimize the start-up latency. We propose a method to obtain the amount of data blocks to be loaded to avoid startup jitter and two different prebuffering strategies. Our model presented in this article is based on multi-zone disk platform.

2. Multimedia Disk Scheduling in Zoned Disk
The objective of zoning is to exploit the storage capacity of the cylinder with larger perimeter using the constant linear bit density. While zoning technique exploits storage capacity, its varying transfer rate adds another dimension of complexity in scheduling the data block retrieval operation. [6] presents the method of placing the data blocks in zoned disk based on the popularity of the video file. [9] investigates the various performance aspects of zoned disk for multimedia streaming operation. [8] investigate the scheduling issues in streaming as well as I/O request for text data. [5] proposes to place the multimedia data blocks to each zone in round-robin fashion and proposes SCAN based disk scheduling algorithm for multimedia data retrieval. To provide continuous flow of data blocks to each service session, following two constraints need to be satisfied. The amount of data blocks retrieved in a cycle should be greater than the amount of data blocks played in each cycle. It takes less than the cycle to retrieve the data blocks for all sessions for single cycle's playback. Two conditions of continuity guarantee can be formally described as in Eq. 1 and Eq. 2.

$$T \times r_{display} \le n_i \times b_i \tag{1}$$

$$T \ge \sum_{i=1}^{S} \sum_{j=1}^{Z} \frac{n_i \times b_i}{Z \times r_j} + \sum T_{seek} + \sum T_{latency} + T_{fullseek} \tag{2}$$

Solving Eq. 1 and Eq. 2, the length of cycle T can be obtained as in Eq. 3. Details can be found in [8]. ε, $r_{display}$, r_i, S, and Z in. Eq. 3 corresponds to the total disk head movement overhead, playback rate, maximum transfer rate of i^{th} zone, number of streams and the number of zones in the disk, respectively.

$$T \ge \frac{\varepsilon}{1 - \left(\frac{r_{display} \times S}{Z} \times \sum_{j=1}^{Z} \frac{1}{r_j} \right)} \tag{3}$$

3. Starting New Session in Zoned Disk

3.1 Cycle Extension

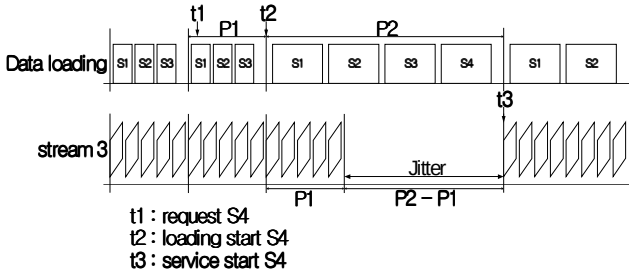

Figure 1 Extension of Cycle and Jitter

When the disk subsystem starts a new streaming session, aggregate playback bandwidth increases. Thus, from Eq. 3, it is mandatory to appropriately extend the length of cycle as well as the amount of data blocks read in a cycle. Fig. 1 illustrates the occurrence of jitter in ongoing stream when cycle length is extended due to the start of new

session. Top half of Fig. 1 illustrates loading of data blocks from the disk. Initially, disk subsystem is servicing three streams, s_1, s_2 and s_3 and the respective cycle length is denoted by P_1. The new request, s_4, arrives at t_1 and the server starts to fetch the data blocks for s_4 in the next cycle. From the third cycle, the cycle length is extended to accommodate new streaming session. Assuming double buffering scheme is used to fetch the data blocks, data blocks loaded in the *extended* cycle are available for playback only after t_3. The bottom half of Fig. 1 illustrates playback sequence of s_3. The amount of data blocks in memory fetched in the second cycle is only for the P_1's playback duration. Since, the blocks fetched in the third cycle will be available only after t_3, it is inevitable that s_3 is exposed temporal lack of data blocks due to the delay in data block retrieval.

3.2 Data Placement in Zoned Disk

In this work, we assume that the data blocks are placed using the placement strategy proposed in [5]. Data blocks in a file are placed to each zone in round robin fashion. Since disk head always sweeps entire disk platter, it is better to make the number of data blocks retrieved for a session in a cycle needs to be the integer multiples of the number of zones. This entails extra buffer space requirement. Fig. 2 compares the number of data blocks retrieved in a cycle when the playback rate is 1.5 Mbits/sec. As can be seen in Fig. 2, loading the integer multiples of the data blocks does not entail significant buffer overhead.

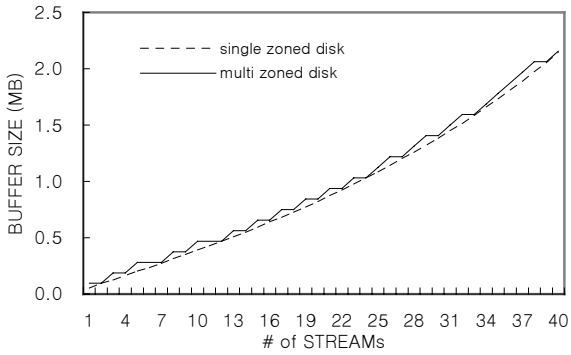

Figure 2 Buffer Size : Multi vs. Single Zoned Disk

4. Prebuffering

We call the operation of loading the data blocks for the possible usage in the future as *prebuffering*. Fig. 3 illustrates how preloading can resolve the insufficiency of data blocks when extending a cycle. X and y axis denotes the time and the amount of data blocks in memory for a single session, respectively. At the end of each cycle, the client consumes all data blocks loaded for the respective cycle. In Fig. 3, quality of ongoing session is not affected by cycle extension from c_n to c_{n+1} since it has sufficient amount of *surplus* data blocks which was loaded prior to service start. The important issue is to determine the optimal amount of preloaded data blocks, *m*, and to devise a method to minimize the startup latency, *L* involved in prebuffering.

Cycle extension keeps occurring until the number of ongoing streams(or the disk utilization) reaches the upper bound. To make a streaming session resilient to jitter caused by cycle extension, the amount of data blocks should be sufficient for playback in maximum length cycle. Let *max* and T_{max} be the maximum number of streams and cycle length with *max* number of streams, respectively. The underlying assumption is that every stream has the same playback bandwidth. It is not difficult to generalize this constraint to heterogeneous playback bandwidth situation. The server can start dispatching data blocks once the data blocks for T_{max} duration's playback are loaded to memory. Let B_i be the amount of data blocks required for T_i's playback. The total amount of buffer when there are *i* sessions is sum of B_{max} and B_i and is represented as in Eq. 4.

$$Buffer_i = B_{\max} + B_i = (T_{\max} + T_i) \times r_{display} \quad (4)$$

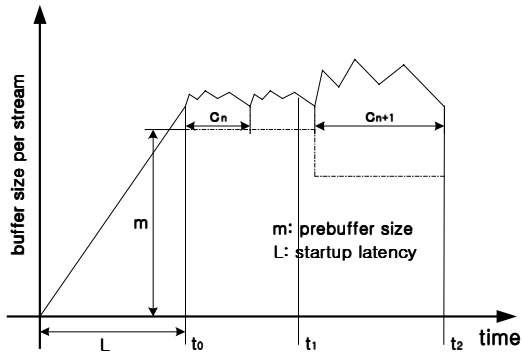

Figure 3 Preloading of Data Blocks

5. Pre-buffering Strategy

5.1 Prebuffering Latency

With B_{max} amount of data blocks in memory, i.e. the amount of data blocks for maximum length cycle, a stream is robust against the temporal insufficiency of data blocks in cycle extension. However, the streaming client experiences longer service startup latency because client can consume the data blocks only after B_{max} amount of data blocks becomes available in memory. Let us assume that with the arrival of new stream, *s*, the cycle length is extended to T_i. Each session loads B_i amount of data blocks in each cycle. Since the *s* can consume the data blocks only after B_{max} amount of data blocks are available in memory, start up latency, τ for *s* can be formulated as follows.

$$\tau = \left\lceil \frac{B_{\max}}{B_i} \right\rceil * T_i \quad (5)$$

5.2 Incremental Pre-buffering

To improve the startup latency, we propose *Incremental Pre-buffering* scheme. In incremental pre-buffering, the server establishes the cycle longer such that surplus fraction of which is used for preload the data blocks[4,10]. The streaming client can start playback immediately after the first cycle completes and is supplied more blocks than it consumes in each cycle. The streaming client can save data blocks which can be potentially used to survive the cycle extension.

Let *T* be the length of cycle. We extend the cycle by extension factor, α, such that a certain fraction of cycle can be used for preloading the data blocks. The continuity requirement of Eq. 1 is re-stated as in Eq. 6.

$$\alpha T \times r_{display} \le n_i \times b_i \quad (6)$$

Eq. 6 states that the amount of data blocks read in each cycle should be sufficient for αT_i's playback. However, the Eq. 2, i.e. total time to retrieve all the data blocks, should still be satisfied. With Eq. 6 and Eq. 2, we can obtain the cycle length and the respective amount of

data blocks to be loaded in each cycle when a certain fraction of cycle is used to prebuffer the data blocks as in Eq. 7. The actual cycle length T_i will corresponds to αT.

$$n \geq \frac{O(n)r}{\frac{1}{\alpha} - \left(\frac{\sum_{i=1}^{n} r_i}{Z} \times \sum_{j=1}^{z} \frac{1}{B_j} \right)} \times \frac{1}{b_i} \quad (7)$$

Details of derivation step can be found in [10]. *(α-1)T* time in each cycle can be used to *preload* the data blocks for individual session. This fraction of time will be used to incrementally *preload* the data blocks for *n* sessions. Since retrieving additional data blocks from the same file does not entail any additional disk head movement overhead, surplus time can be dedicated to transferring the data blocks. In multi-zoned disk, amount of preloaded data blocks needs to be the integer multiples of the number of zones. Let *n*, *m*, B_i and *z* be the number of sessions, the number of blocks preloaded for a single stream in a zone, maximum transfer rate of zone *i* and the number of zones, respectively. Then, Eq. 8 should be satisfied.

$$0 < \frac{n\left(\sum_{i=1}^{z} \frac{mb}{B_i} \right)}{(\alpha - 1)T} \leq 1 \quad (8)$$

The amount of preloaded buffer for individual session for each cycle can be formulated as in Eq. 9. *m* is largest integer satisfying Eq. 8.

$$b_{prebuffer} = mzb \quad (9)$$

6. Simulation Study

We perform simulation to examine the buffer overhead of proposed prebuffering strategy. The disk is modeled after IBM Deskstar 34 GXP with storage capacity of 27.3 GB. Disk transfer rate ranges from 13.8 MB/s to 22.9 MB/s. There are 17494 cylinders and 12 zones. The disk rotates at 7200 RPM. Fig. 4 and Fig. 5 illustrate the total buffer size for 1.5 Mbits/sec and 19.2 Mbits/sec streams, respectively. The disk used in our simulation can theoretically support upto 7 and 95 number of streams for 19.2 Mbits/s and 1.5 Mbits/s playback rate, respectively. Fig. 4 illustrate the per stream buffer when the call admission control module sets the disk service limit to 70, 80, and 90 streams, respectively. The buffer size in these figures is the sum of prebuffer and the data blocks retrieved in the respective cycle. The prebuffer size varies widely depending on the max number of streams. In Fig. 4, prebuffer size is 24 MByte when the upper bound is set to 95. However, prebuffer size is 4 Mbyte when the upper bound of the concurrent number is stream is set to 70.

7. Summary

In this article we present the novel method of avoiding temporal insufficiency of data blocks which occurs due to cycle extension. In cycle based disk scheduling for multimedia data retrieval, it is inevitable that the ongoing session gets exposed to jitter when the server starts the service of new session. We propose a technique called *prebuffering* which is to load the data blocks to overcome the temporal insufficiency of data blocks with cycle extension. Prebuffering necessary amount of data blocks prior to starting service can cause non-negligible amount of service startup latency. To resolve this problem, we propose a technique called *incremental prebuffering* where a fraction of cycle is set aside to be used to load the data blocks for future use. We develop an elaborate model to compute the length of the cycle which incorporates the time to preload the data blocks.

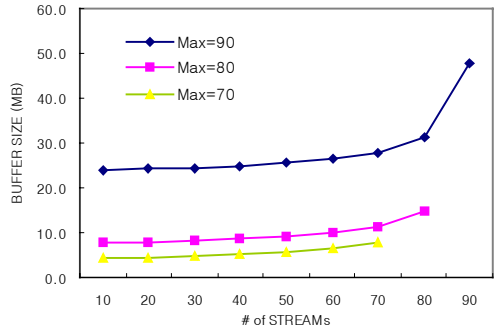

Figure 4 Per Stream Buffer Size with 1.5 Mbits/sec Stream

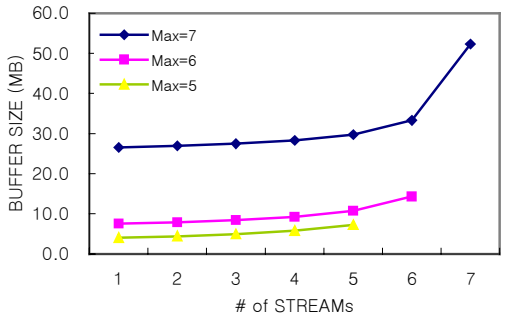

Figure 5 Per Stream Buffer Size with 19.2 Mbits/sec Stream

8. References

[1] Kenchammana-Hosekote, D.R. and Srivastava, J., Scheduling Continuous Media on a Video-On-Demand Server. In Proc. of International Conference on Multi-media Computing and Systems, Boston, MA, IEEE, 1994.

[2] Chen, M.S., Kandlur, D.D., and Yu, P.S., Optimization of the Grouped Sweeping Scheduling (GSS) with Heterogeneous Multimedia Streams. In Proc. of ACM Multimedia 1993. August 1993. Anaheim, California.

[3] Rangan, P., Vin, H. and Ramanathan, S., Designing an On-Demand Multimedia Service. IEEE Communication Magazine. 1992. 30(7): p.56-65.

[4] Won, Y. and Ryu, Y.S., Handling Sporadic Tasks in Multimedia File System. In Proc. of ACM Multimedia Conference. Oct. 2000. Los Angeles, CA, USA.

[5] Ghandeharizadeh, S., Kim, S.H., and Shahabi, C., Continuous Display of Video Objects Using Multi-Zone Disks. Second International Baltic Workshop on DB and IS, 1996.

[6] Tewari, R., et al., Placement of Multimedia Blocks on Zoned Disks. In IS&T/SPIE Conference on Multimedia Computing and Networking (MMCN 1996). Jan. 1996. San Jose, California.

[7] Won, Y., Minimizing Intermedia Interference in Integrated File System, Submitted for Publication.

[8] Guido Nerjes, Peter Muth, and Gerhard Weikum. “Stochastic service guarantees for continuous data on multi-zone disks”, In *Proceedings of the 16 th Symposium on Principles o Database Systems*, Tucson, Arizona, 1997.

[9] P.K.C. Tse and C.H.C. Leung. “Improving multimedia systems performance with constant-density recording disks”. Multimedia Systems, 8(1):47-56, January 2000

[10] A. Neogi, A. Raniwala, T. Chiueh, “Phoenix: A Low-Power Fault-Tolerant RealTime Network-Attached Storage Device”, In Proc. of ACM Multimedia Conference, Oct. 1999, Orlando, FL, USA