A Buffer Management Strategy for a Model of Digital Continuous Media Display

Kingsley C. Nwosu,
IBM Corporation,
Large Scale Computing Division,
MS/992, Neighborhood Rd.,
Kingston, NY 12401.
email:nwosuck@donald.aix.kingston.ibm.com

Abstract

The advent of multimedia information processing has brought a lot of information management issues into attention in the past few years. Prominent among them include the efficient storage and retrieval of multimedia objects. When handling digital continuous media, these issues become more critical. An efficient storage and retrieval process becomes useful and meaningful when combined with efficient buffer management. The buffer acquisition and retention mechanisms go a long way to achieving the display requirements. In this paper, we present a technique for continuous media displays that strive to maintain some degree of data availability through efficient buffer management. Media display factors are developed for data acquisition and buffer retention to satisfy media display requirements.

Keywords: *buffer contention, data consumption rate, efficient storage, multimedia, object decomposition, object modeling.*

1 Introduction

Over the years, different areas of the computing technology have developed each in its own respect and most often ignoring other concurrent advancements in related areas. Most of these advances, in any given area, have been primarily to solve specific problems related to a given application area.

The rapid advances in the technology of display devices, computers, networks, storage devices, and software engineering have pushed the emerging multimedia applications into becoming one of the most important and promising research areas. The speed and bandwidth of computer networks have increased substantially that in no distant future we will have gigabit per second networks. The speed of the CPU have increased drastically as a result of advances in VLSI technology that today we have tens of hundreds of MIPS processors and we expect even faster ones in the near future. The capacity of mass storage devices has increased enormously and at the same time their physical sizes have shrunk. The presence of powerful personal computers and work-stations with very high resolution monitors and cheap prices have become pervasive. The price of memory has also decreased considerably. The availability of multicomputers with high processing and storage capabilities has changed our vision of computers. Today, we have multicomputers with several hundreds of processors and enormously large disk space and the future looks brighter for multicomputers with thousands of processors and larger disk space. In spite of the heterogeneity of computing systems, it has become increasingly easy to interconnect different systems across the networks. There have been considerable improvements in digital compression algorithms and the development of specialized processors for digitalization and processing of video and related media information.

Since most of the technological advances progressed, more or less, in parallel, it has become obvious that the integration and exploitation of these advances will bring the computing technology and information processing to a different dimension. This dimension is the capability to process those media information that we, over the years, thought too large to store, process, and transport and too diverse to integrate. Of course, I am talking about *multimedia.*

Multimedia information processing encompasses the integrated generation, representation, processing, storage, and dissemination of independent machine processable information expressed in multifarious time dependent and independent media. A unique feature of multimedia is the highly diversified media types and characteristics. Multimedia objects come in many different innate sizes, with differing processing and display requirements. The storage, transmission, and display efficiencies of multimedia objects are greatly affected by their sizes.

One of the system's resources that plays a substantial role for efficient digital information processing are the *buffers* between the main memory and secondary storage devices. Buffers are used to temporary store those data that we expect to use in the near future and to mitigate the speed disparity between the main memory and secondary storage devices. An efficient buffer management strategy becomes a *sine qua non* for multimedia information processing especially for the displays of digital continuous media. For real-time information retrieval and presentation, it is imperative that data, for a given medium, be retrievable and available at some given rate.

In this paper, we present a buffer management strategy for digital continuous media display. We use the multimedia object model that we proposed in our earlier works. The model presents a classification of different multimedia objects based on their I/O requirements for efficient storage and retrieval. The buffer management technique discussed here relies on some of the characteristics and properties of the multimedia objects described by the model.

2 Related Issues

If accesses to secondary storage devices do not incur considerable performance degradation in a system as a result of the speed disparity between main memory and secondary storage devices, then the role of buffers and caches will redundant. Taking cognizance of the problems with I/O's to secondary storage devices, a number of efficient data storage techniques [1]-[12] have been proposed and utilized. These techniques involve either data compression, data stripping, data clustering, or a combination of them. Due to the predictable and mostly invariable sizes of conventional data, buffer management, in conjunction with these techniques, has been very instrumental in maximizing a system's throughput. The importance of buffering cannot be over-emphasized as is evident from the performance enhancements in networks, file systems, and data management strategies as a result of buffer utilization. These approaches have proven helpful for conventional data processing until the advent of multimedia. Multimedia data, by their diversified nature, properties, and requirements, pose additional and different problems for system performance. The buffer management techniques that have been proposed and utilized are inadequate and inefficient because they are not targeted towards continuous media objects with emphasis on their peculiar characteristics.

3 The Multimedia Object Model

The buffer management strategy described here relies on the properties, characteristics, and storage techniques for the multimedia object model presented in [13][14]. These earlier works present a classification of multimedia objects with respect to their I/O requirements and the decomposition and storage of the objects to meet their retrieval requirements.

The multimedia object model is built around a *composite object*, a number of *complex objects*, and several *Data Elements* (DEs). A composite object represents the results of a multimedia session and forms the root of its components. A complex object is any internal node of a composite object tree while the DEs are the leaf nodes. The DEs are the storable entities of a composite object, i.e., they represent the data for the multimedia objects, e.g., text, sound, graphics, video, audio, etc. For example, in Figure 1, $o8$ and $o9$ form the complex object $o6$ while $o6$ and $o7$ form the complex object $o3$. The model described is strictly a data storage model and not a multimedia database model. A database model captures the functional, operational, and manipulative characteristics and requirements of some set of data while a data storage model defines the storage and retrieval mechanisms to satisfy data availabilities and maximize system throughput. Every database system implements its own storage mechanism. Our storage model is designed so that it can

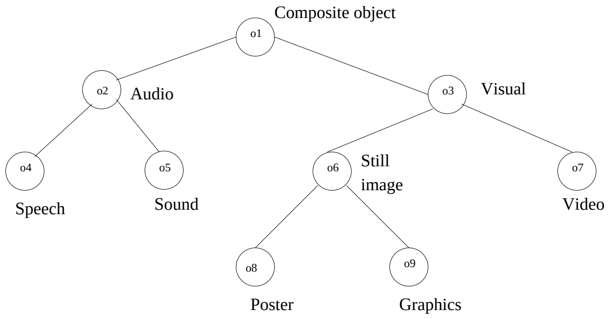

Figure 1: An example of a composite multimedia object tree.

Figure 2: Examples of class-one, class-two, and class-three DEs.

the aim of ensuring parallel accesses to media data by accessing different AUs to meet the data availability rates. Each DE, after decomposition, comprises a number of AUs which are allocated individually on separate storage devices. For example, using a composite object with 4 DEs, $DE1, DE2, DE3$, and $DE4$, where $DE1$ is a class-one DE, $DE4$ is a class-three DE, and $DE2$ and $DE3$ are class-two DEs; and assuming that the sizes of $DE1, DE2, DE3$, and $DE4$ are 320KB, 50KB, 100KB, and 500KB, respectively, therefore, in a homogeneous storage device configuration (where the bandwidths of the storage devices are the same), the DEs are decomposable into 7 SEs, $\{se1, \ldots, se7\}$, 1 SE, $\{se1\}$, 2 SEs, $\{se1, se2\}$, and 10 SEs, $\{se1, \ldots, se10\}$, respectively, and the AUs are constructed as shown in Figure 3. We assume, also, that the bandwidth of the storage devices is 50KB per unit time. The numbers below the SE numbers in the boxes represent the *physical addresses* of the SEs within its AU. We assume that each AU's physical address starts from zero. For example, $se3$ in DE4 spans the physical addresses from 100 to 149. In a homogeneous storage device configuration, the size of all the SEs of all the AUs of a multimedia object is the same, thereby, making the offsets between different SEs in different AUs uniform. This uniformity does not exist for heterogeneous storage device configurations. For heterogeneous storage device configuration, a number of different sizes and configurations of AUs are possible because of the differences in bandwidths. For example, assuming that we have storage devices whose bandwidths are either 50KB or 100KB per unit time, therefore, using the composite object in Figure 3, we can construct 5 SEs for $DE1$, 7 SEs for $DE4$, 1 SE for $DE2$ and 1 SE for $DE3$ as shown in Figure 4. These are one possible generation of AUs; there are other ones. If none of the storage sets of a DE element is valid or acceptable, then we cannot allocate the DE to meet its requirements.

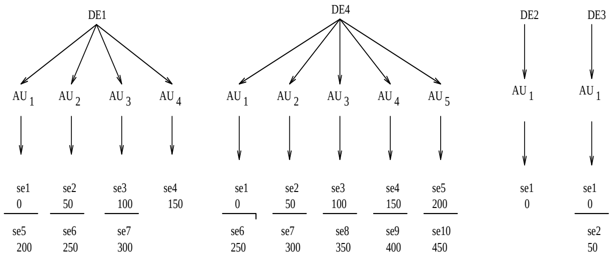

Figure 3: Homogeneous devices: sample construction of AUs.

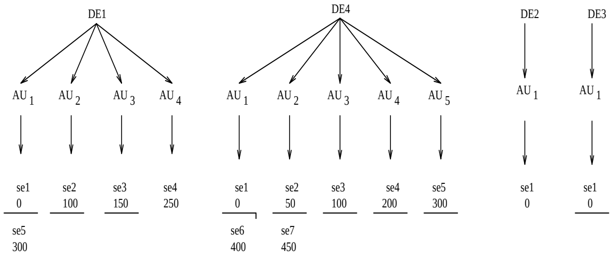

Figure 4: Heterogeneous devices: sample construction of AUs.

6 Object Allocation Requirements

Within a computing environment, certain allocation strategies must be utilized to allocate a DE or groups of DEs belonging to a composite object to the storage devices. In [13][14] we present the *intra DE, intra complex object*, and *inter composite object* allocation policies. These policies specify how the AUs of a DE, complex object, or composite object are allocated with respect to other AUs of same or different DE, complex object or composite object. The intra DE allocation policy requires that every AU of a DE be allocated to a different storage device. The intra complex object allocation policy determines the allocation of DEs within a complex object. Specifically, it requires that the AUs of the DEs of a complex object be stored in different storage devices. The inter complex object allocation policy stipulates the allocation requirements for the AUs of DEs that belong to different complex objects. The inter composite object allocation policy stipulates the allocation process when two composite objects share a DE. When an object is shared, the shared object is logically thought as belonging to each of the composite objects and the aforementioned allocation policies are enforced in allocating the DEs of those composite objects.

7 Problem Formulation and Analysis

The allocation process described above efficiently decomposes the objects of a composite multimedia object and stores them in storage devices such that a retrieval request will achieve a bandwidth which satisfies each DE's retrieval requirement. Therefore, assuming that each I/O request for a DE maintains its retrievability request, our problem for effective buffer management include:

- What are the temporal implications of a read-ahead for a multimedia object?
- What are the effects of an object's inherent properties on the quantitative values of data read-ahead?
- How are the quantitative conditions of a system buffers effectively and efficiently managed for continuous display of an object?
- With respect to the temporal and quantitative issues of data availability, how do we develop an efficient and reliable buffer management strategy that precludes buffer over-flow, satisfies display requirements, and maximizes system performance?

In conventional environments devoid of continuous media, the primary motivation for data read-ahead is for anticipation of future use. However, for a continuous media display, there is no anticipation of usage but a certitude of usage. We are certain that within some specified time frame, some data that we do not currently have in memory will be required. Furthermore, as a consequence of display mechanism, with the exception of program dependent interruptions, data for display must be available within some time interval. Delays introduced by secondary storage device I/O latencies are intolerable. The amount of data required for continuous display of an object must be readily available at the time of need.

7.1 Pertinent Display Factors

From the model used, it becomes obvious that the expected retrieval rate of an object represents the amount of data that must be available per unit time for the object's display requirements. However, at the initiation of a display session, we do not, in most cases, expect some of the required data to be available. Furthermore, we must decide the minimum amount of data that must be available during each display session. This amount of data is a function of the amount of data that is required per unit time of display. We refer to that amount of data as the *minimum utilization value* (muv). The amount of time required for the muv of an object to be available is called the *startup time* ($stime$). At any given time, an object being displayed must maintain its muv. The amount of data that is consumed by a display of an object per unit time is called its *data consumption rate* (dcr). During the display of an object, there must be a minimum number of free buffers available to cover the dcr. The minimum number of buffers required to maintain an object's dcr is called its *buffer consumption rate* (bcr). We denote the size of a buffer as buf^{size}, the maximum fraction of the system's buffers that a particular process can utilize as r, and the number of buffers in the system as buf^{num}.

7.2 Functional Dependent Derivations

Since we are only concerned about displaying objects, the objects to be dealt with, with respect to buffer management, are those associated with expected retrieval rates. Consequently, we are dealing with class-one and class-three DEs. Obviously, the amount of data that a target object utilizes per unit time is the same as its expected retrieval rate, as elucidated in the references. Consequently,

$$dcr = err.$$

The bcr identifies the number of buffers that must be filled per unit data retrieval. It depends on the amount of data that must be retrieved per unit time. Specifically,

$$bcr = \lceil \frac{dcr}{buf^{size}} \rceil$$

The amount of data that must be available for display, in main memory and buffer, is expressed as a function of the data consumption rate. We express that value as

$$muv = dcr \times n$$
$$\text{where } 1 \leq n \leq \tfrac{buf^{num} \times r}{bcr}.$$

The value, muv, is expressed such that the number of buffers it utilizes is within the allowed range.

The amount of time that must elapse from the time of a display request and time of satisfying the availability of muv is the $stime$. Therefore, given an muv, we express the startup time as

$$stime = \frac{muv}{dcr}.$$

For example, given that a system has 1000 buffers, each buffer is of size $100KB$, a process can use a maximum of 0.02 of those buffers, and that we want to display an object with an expected retrieval rate of $500KB$, and the minimum amount of data that must be available, at any given time from the secondary storage devices is $1500KB$, therefore,

$$\begin{gathered} buf^{num} = 1000, \\ r = 0.02, \\ dcr = err = 500, \\ bcr = 5, \\ stime = 3, \\ muv = 1500 \text{ where } n = 3. \end{gathered}$$

In the example above, 3 time units must elapse from the time of request initiation and actual display of data. Each I/O request to the secondary storage devices must deliver 500KB per unit time, and there must be at least 5 free buffers for each I/O request to the devices.

8 The Buffer Management Process

We assume that a display request provides the object name, *object*, offset within the object, $offset$, and an optional duration (time), *duration* or size of display, *size*. Either *duration* or *size* must be specified in conjunction with other parameters. In a display request, we need to know either the total amount of data to display or total time for the display. Given either the *size* or *duration* we can determine the other. Specifically,

$$size = err_{object} \times duration$$

and

$$duration = \lceil \frac{size}{err_{object}} \rceil$$

Actually, for the purposes of data retrieval, *size* is more important.

The first thing to be done is to determine the degree of buffer management necessitated by a display based on the time for the display. If the total time for the display is less than the time unit of *dcr*, then we will incur minimal buffer management. The *stime* becomes less than a unit time and we will only be concerned about the availability of at most *bcr* free buffers for the whole display session.

When the time of display is larger than the time unit of *dcr*, we initially have to satisfy the data availability for the *muv*. We collect the number of free buffers for *muv* and retrieve an amount of data equal to *muv* or *size*, whichever is smaller. For a given display session, two processes (lightweight processes) are utilized - one for sending data to the display device, and the other for collecting free buffers and issuing I/O requests to the secondary storage devices, when necessary. Data requests to secondary storage devices are made synchronously with data display. Therefore, once a number of buffers necessary for *stime* and its initial synchronous I/O request have been obtained, these buffers are reserved for the display session. At each unit time cycle, the buffers released from the latest display are used for the next satisfaction of the data availability. By reserving those buffers, we are guaranteed that we can not run into buffer starvation as a result of unavailability of *bcr* free buffers. Consequently, all things being equal, the continuity of data availability and flow of display are constantly satisfied.

8.1 Multi-user and Network Effects

The availability of the required buffers to satisfy *stime* and subsequent continuous display of data, as described above, applies to the best case situation. However, since must operating environments are multi-user or distributed systems, the inter process buffer sharing and network delays affect buffer availabilities. In the best case scenario, buffers become available on request and so the *stime* and display continuity are maintained and achieved as needed. In other words, there may not be buffer contentions or network delays to exacerbate display quality and efficiency. However, in multi-user system environments where any number of users and processes may be active concurrently, buffer contention between processes arise. Furthermore, in a distributed environment, where data must be carried over networks, network delays become an issue. The degree of contention, processes' buffer utilizations, and network issues affect display quality and efficiency of continuous display. Therefore, with respect to a system environment, it may be necessary to accentuate display buffer acquisition characteristics to account for potential delays.

8.1.1 Accentuating Buffer Acquisition Factors

When buffer contention is anticipated or network delays are imminent, which may lead to data availability delays, the data acquisition factors should be adjusted to mitigate the adverse effects. Obviously, it more efficient to make the adjustments at the display initiation than during the inter data retrieval process. We can use the real startup time (*rstime*) vis-á-vis the expected startup time (*stime*) to predict the degree of system buffer contentions or network delays. The ratio between the *rstime* and *stime* can be used to accentuate the *muv*. The new *muv* computed as a result of adverse buffer contention or network effects is called the *accentuated muv* (*amuv*) and dervied as

$$amuv = \left[\frac{rstime}{stime} \right] \times dcr$$

. The objective of the *amuv* is to guarantee that after its initial satisfaction, subsequent consumption of data for display, and acquisition of buffers and data retrieval will maintain *muv*. Therefore, the delays incurred as a result of buffer contentions do not affect display continuities.

9 Simulation Model

In order to analyze the buffer strategies proposed, we conducted simulated some continuous media displays with buffer utilizations in a multi-user environment. A number of processes were generated where each process periodically acquires a number of buffers, and release the buffers after some randomly determined possession time. There are also a number of processes that are involved with handling large continuous media objects for displaying. Each of the continuous media objects specifies its *bcr*. For a consistent analysis, the processes being analyzed have the same *size* or *duration* and a uniform *muv* is used at each experiment. Our objective is to determine the rate of buffer availability for the processes at each time cycle. At each unit time, we determined the number of a process's buffers that were filled with data and the *muv*. We then compare the real number of buffers and expected number of buffers.

9.1 Simulation Result

Figure 5 shows the results of the real and expected number of buffers for a process without our buffering strategy. The buffer consumption rate for the process is 2, i.e., the display per unit time consumes the contents of 2 buffers. We assumed that its data retrieval rate is equivalent to the contents of two buffers. Therefore, its *stime* is 1. From the figure, we observe that at time $t = 1$, the initial 2 buffers became available, but were subsequently consumed. Between $t = 1$ and $t = 2$, the process has to obtain the next 2 buffer worth of data. During that interval, no display was possible. At $t = 2$, the buffers became available and were subsequently consumed, etc.

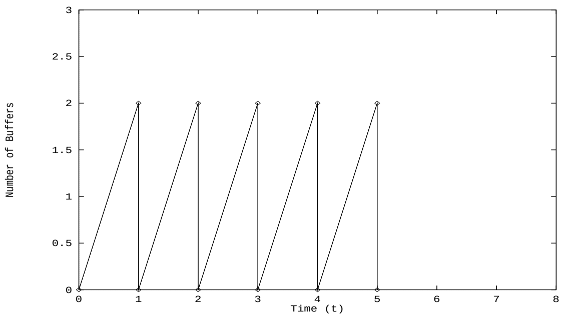

Figure 5: Buffer utilization and availability without proposed strategy.

In Figure 6, we show the same process under the proposed buffer management strategy. It utilizes an *stime* $= 2$. At the inception of the display, the process has 4 buffer worth of data. We also observe that at $t = 2$, the process consumes its *bcr* but can still concurrently display data between $t = 2$ and $t = 3$ while more data were being retrieved. In a buffer contention or distributed environment with adverse effects, the graph looks a little different from Figure 6, however, there were always enough data for concurrency.

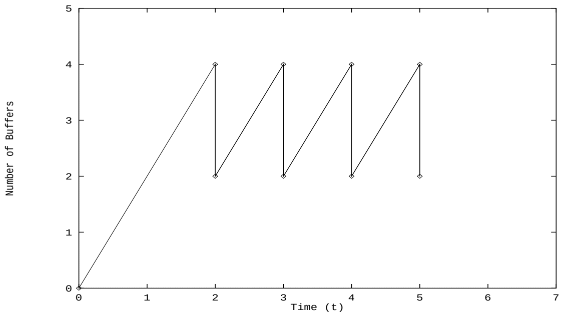

Figure 6: Buffer utilization and availability with proposed strategy.

10 Conclusions

In this paper, we have presented an efficient buffer management for the display of a model of continuous media. We highlighted the decomposition and storage of the media objects. The necessary and sufficient buffer management factors and their applications were defined and analyzed. We have shown that in multiuser system environments were buffer contention or network delays may be prevalent, the utilization of a buffer management that does not take cognizance of some of the unique properties of continuous media objects may adversely affect display quality and efficiency. We defined such buffer management factors as the minimum utilization value, buffer consumption rate, data consumption rate, startup time, etc. The buffer management strategy presented here, with its simplicity, can easily be incorporated into existing environments.

In designing the buffer management strategy described here, we assume that the data contained in the buffers have been de-compressed when necessary. It is necessary that no additional over-head be incurred for transferring data from the buffers to the display devices. Therefore, no delays should be introduced by data de-compression. Data in the buffers awaiting display must be in immediate usable form.

References

[1] **C. G. Bell**, The mini and macro industries, *IEEE Computer*, Vol. 17, No. 10, 1984, pp. 14-30.

[2] **W. Myers**, The Competitiveness of U.S.A. Disk Industry, *IEEE Computer*, Vol. 19, No. 11, 1986, pp. 85-90.

[3] **M. Y. Kim**, Synchronized Disk Interleaving, *IEEE Trans. on Computers*, Vol. C-35, Vol. 11, November 1986, pp. 978-988.

[4] **M. Livny, S. Khoshafian, H. Boral**, Multi-Disk Management Algorithms, *Proc. 1987 ACM SIGMETRICS Conf. on Measurement and Modeling of Comp. Syst.*, pp. 69-77.

[5] **P. Chen, D. Patterson**, Maximizing Performance in a Striped Disk Array, *Proc. 1990 ACM SIGARCH 17th Intern. Symp. on Comp. Arch.*, Seattle, WA, May 1990, pp. 322-331.

[6] **G. Held**, Data compression: techniques and applications, hardware and software considerations, published by *Wiley, NY, NY*,1987.

[7] **James Storer**, Data compression: methods and theory, published by *Computer Science Press*, Rockville, MD,1988.

[8] **D. J. LeGall**, MPEG: a video compression standard for multimedia applications, *CACM Vol. 34, No. 4*, April 1991.

[9] **J. Zupan**, Clustering of Large Data Sets, *Technometrics*, Vol. 29, Nov., 1987, pp. 497.

[10] **A.D. Bell, F.J McErlean, P.M. Stewart**, Clustering Related Rules in Databases, *The Computer Journal*, Vol. 31, June 1988, pp. 253-257.

[11] **J.S. Deogun, V.V. Raghava, T.K.W. Tsou**, Organization of Clustered Files for Consecutive Retrieval, *ACM Trans. on Database Systems*, Vol. 9, December 1984, pp. 646-671.

[12] **D. P. Anderson, Y. Osawa**, A File for Continuous Media, *ACM Trans. on Computers*, Vol. 10, No. 4, November 1992, pp. 311-337.

[13] **C. Y. R. Chen, Kingsley C. Nwosu, P. Bruce Berra**, Modeling and Storage Allocation Strategies for Homogeneous Parallel Access Storage Devices in Real Time Multimedia Information Processing, *Proc. IEEE 5th International Conference on Computing and Information* (ICCI), Sudbury, Ontario, Canada, May 27-29, 1993.

[14] **C. Y. R. Chen, Kingsley C. Nwosu, P. Bruce Berra**, Multimedia Object Modeling and Storage Allocation Strategies for Heterogeneous Parallel Access Storage Devices in Real Time Multimedia Computing Systems, *Proc. IEEE Computer Society's 17th Annual International Computer Software and Applications Conference (COMPSAC)*, Phoenix, Arizona, November 1-5, 1993, pp. 216-223.