5-1997

Predictable File Access Latency for Multimedia

Dan Revel
Oregon Graduate Institute of Science & Technology

Crispin Cowan
Oregon Graduate Institute of Science & Technology

Dylan McNamee
Oregon Graduate Institute of Science & Technology

Calton Pu
Oregon Graduate Institute of Science & Technology

Jonathan Walpole
Oregon Graduate Institute of Science & Technology

Citation Details

Revel, D., Cowan, C., McNamee, D., Pu, C., & Walpole, J. (1997, March). Predictable file access latency for multimedia. In Proc. 5th International Workshop on Quality of Service (IWQOS'97), Columbia University, New York, USA (pp. 401-404).

Predictable File Access Latency for Multimedia

D. Revel, C. Cowan, D. McNamee, C. Pu, and J. Walpole
Department of Computer Science and Engineering
Oregon Graduate Institute of Science & Technology
20000 N.W. Walker Rd., P.O. Box 91000
Portland, OR 97291-1000
(503) 690-1121
(503) 6901553 fax
{revel,crispin,dylan,calton,walpole}@cse.ogi.edu

Abstract

Multimedia applications are sensitive to I/O latency and jitter when accessing data in secondary storage. Transparent adaptive prefetching (TAP) uses software feedback to provide multimedia applications with file system quality of service (QoS) guarantees. We are investigating how QoS requirements can be communicated and how they can be met by adaptive resource management. A preliminary test of adaptive prefetching is presented.

Keywords

QoS, adaptive, multimedia, prefetching

1 INTRODUCTION

Multimedia applications need to handle continuous media data in real-time. Because of its high volume this data must be streamed into memory from local or remote file systems on secondary storage devices. Real-time constraints make good multimedia performance dependent on prefetching data so that it will be available in memory in a timely and predictable manner.

Effective prefetching decisions depend on two inputs: 1) application behavior and quality of service requirements, and 2) system resource availability and performance characteristics. For distributed multimedia applications both of these inputs may vary dynamically.

On one hand, complex multimedia content and interactive user controls can cause fluctuations in the demands placed on the system by a particular application. On the other hand, distributed multimedia applications operate in shared environments where they may compete with other application for system resources, such as network and disk bandwidth, memory, and processor time.

This paper describes *transparent adaptive prefetching* (TAP). TAP is one aspect of our on-going research into 1) how to specify QoS requirements and communicate them among system components and layers, and 2) how to manage resources such that QoS requirements are met.

This research is partially supported by DARPA grant N00014-94-1-0845, DARPA contract F19628-95-C-0193, NSF grant CCR-9224375, and grants from Hewlett-Packard, Intel and Tektronix.

In the next section we describe the limitations of current multimedia prefetching. Section 3 describes our design of an adaptive prefetching service that provides constrained latency file system access. Section 4 presents the results of a preliminary test of adaptive prefetching. Section 5 summarizes our results and discusses our current work.

2 MOTIVATION

Prefetching hides storage access latency by fetching data into memory before it is demanded by an application. The timely delivery of data is critical to multimedia presentations. Data which arrives too late will either cause a gap or a delay in the presentation. Data which arrives too early will displace other data from the file system buffer cache. The ideal is to have prefetched data streaming into memory so that it is available *just in time* as it is needed by the application (Maier 1993).

Many file systems recognize when a file is being read sequentially and do heuristic prefetching (McKusick 1984). Prefetch depth, how far in advance data is requested, can be adjusted to match the rate at which data is being read. This approach takes advantage of an access pattern that can be easily inferred to provide the latency hiding benefits of prefetching. The problem with relying on heuristic prefetching for multimedia is that it is *reactive*. There is inevitably a delay between the time when an application starts accessing data (or changes the rate of access) and when the system adjusts to the new behavior. Further, when data is accessed in a non-sequential pattern, for example an MTV-style series of short video clips drawn from different source files, then no prefetching happens at all.

Faced with inadequate system support for prefetching multimedia applications must address the associated problems of latency, synchronization, and resource allocation on an *ad hoc* basis. By handling prefetching explicitly an application can take advantage of its specific knowledge of what data is likely to be needed in the future and when it needs to be available.

We see two problems with direct application management of prefetching.

First, application management eliminates the device independence provided by utilizing operating system abstractions of resources. Instead applications are left to manually control the timing of prefetch requests. As a result, developers must tune current high performance multimedia applications for specific storage devices (Aref 1997). Without device-specific information applications may use excessive amount of memory due to overly aggressive prefetching, or they may suffer from poor performance due to under-prefetching and dynamic system behavior. Another possibility is that applications may become overly complex trying to track and adapt to current system load.

Second, application management can produce poor resource allocation and scheduling decisions in a shared environment. Applications do not have the system level information or control needed to make or enforce useful decisions. For example, an application may try to buffer prefetched frames in virtual memory expecting them to remain available for low-latency access only to have them paged out by the virtual memory system. One alternative would be to allow applications to pin virtual memory pages so they could not be paged out. In this case, however, resource sharing would be defeated.

3 TRANSPARENT ADAPTIVE PREFETCHING

The goal of transparent adaptive prefetching is to provide multimedia applications with low-latency access to data on secondary storage devices. To achieve adaptive prefetching we use a software feedback mechanism.

Our prefetcher monitors the time it takes to service file system I/O requests and the number of prefetch requests completed but not yet read by the client application. These two pieces of information are used to dynamically adjust the prefetch depth. Between adjustments a constant prefetch depth is maintained by issuing prefetch requests at the same rate that data is read by the client.

We select a prefetch depth using a hybrid algorithm to avoid late arrival of prefetched data An eager criteria decides to increase the prefetch depth, and a more cautious measure is used to decrease the prefetch depth.

The prefetcher maintains a small amount of work-ahead. This is data that has been fetched, but has not yet been read by the client. This work-ahead provides a cushion of ready data which is consumed when prefetch requests are delayed by a burst of competing I/O requests. When the level of work-ahead drops below a safety threshold we increase the prefetch depth.

A ratio between the current prefetch depth and the worst-case latency observed over the past 32 file system I/O's is used to decide when to decrease the prefetch depth. Lulls in activity between I/O bursts could be misinterpreted as a cue to reduce the prefetch depth. Using the worst-case latency introduces a damping function that keeps the prefetcher from reducing the prefetch depth prematurely.

4 EXPERIMENTAL RESULTS

To demonstrate adaptive prefetching through software feedback we conducted an experiment. Our experiment consisted reading one thousand 4 kB 'frames' of data from a file at a rate of 50 frames-per-second. Linux's built-in file read-ahead mechanism was replaced with our adaptive prefetcher for that file. Non-blocking reads were used to simulate a multimedia application that would discard late data rather than wait for it to arrive. After allowing the reading process to run for five seconds a competing process was started. This process read five hundred 4 kB 'frames' at a rate of 50 frames-per-second, but it used Linux's built-in read-ahead instead of our prefetcher.

Our experiment ran on a Toshiba laptop with a 75 MHz Intel Pentium processor, 40 MB of main memory, a Toshiba MK1301MAV 1.3 GB IDE disk drive with a 128 kB cache, and version of the Linux version 2.0.18 operating system.

With the exception of the first four frames of each run, which were missed due to start-up latency, 99% of the non-blocking read calls made by the prefetching application returned with data.

Figure 1 is a sample execution of our experiment demonstrating the behavior of our prefetching algorithm. Prefetch depth is the controlled variable which is adjusted dynamically. Work-ahead and worst latency are the input variables for our algorithm as described earlier. Examination of the graphs shows that our hybrid algorithm was able to track a change in I/O workload and to adapt the prefetch depth accordingly.

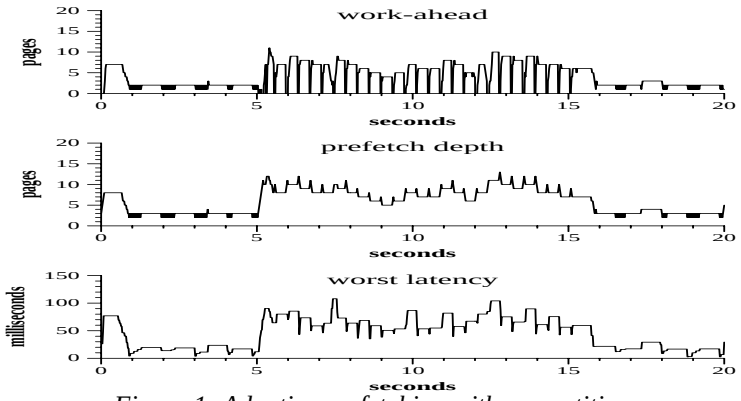

Figure 1: Adaptive prefetching with competition

5 SUMMARY AND CURRENT WORK

Our initial results show that adaptive prefetching can be used to provide multimedia applications with predictable low-latency access to data on secondary storage.

We are currently working on a QoS interface for TAP. This interface will build on Patterson's transparent informed prefetching (Patterson 1995) allowing applications to express their needs in a vocabulary that is meaningful to them. We will then modify OGI's multimedia player (Koster 1996) to use TAP.

6 REFERENCES

Aref, W.G., Kamel, I., Niranjan, T.N. and Ghandeharizadeh, S. (1997). Disk Sceduling for Displaying and Recording Video in Non-Linear News Editing Systems. *Proceedings of Multimedia Computing and Networking 1997*. SPIE Proceedings Vol. 3020, San Jose, February 1997.

Koster, R. (1996). Design of a Mulitmedia Player with Advanced QoS Control. Master's thesis, Oregon Graduate Institute of Science and Technology, Portland, Oregon, 1996.

Maier, D., Walpole, J. and Staehli, R. (1993) Storage System Architectures for Continuous Media Data. In *FODO '93 Proceedings*, *LNCS*, v. 730, Springer-Verlag, pp. 1-18, 1993.

McKusick, M.K., Joy, W.N., Leffler, S.J., and Fabry, R.S. (1984) A Fast File System for UNIX. *ACM Transactions on Computer Systems*, 2(3):181-197, August 1984.

Patterson, R.H., Gibson, G.A., Ginting. E., Stodolsky., D., and Zelenka, J. (1995). Informed Prefetching and Caching. *Proceedings of the Fifteeth ACM Symposium on Operating System Principles*, pages 79-95, December 1995.