IEEE TENCON '93 / Beijing

Design of a File Server Operating System *

Ho-Te Yeh, Lih-Fang Lin, and Yen-Jen Oyang

Department of Computer Science
and Information Engineering
National Taiwan University
Taipei, Taiwan, China

Abstract

This paper describes the design of a file server operating system called Cypress. The design of Cypress is aimed at improving the file server response time through exploiting latest hardware advances. Cypress is distinctive in its caching and management strategies that take advantage of the large-capacity disk caches, tens to hundreds megabytes, and the powerful processors in contemporary file servers. If compared with general-purpose operating systems that are installed to perform file server operations, Cypress enjoys another advantage of featuring more predictable response time for real-time applications. Based on our measurement on our first version of implementation, the average hit response times of Cypress when running on a Intel 80486 CPU based PC with 32-megabyte memory are 1.38 ms and 1.39 ms for read and write requests, respectively.

1 Introduction

File servers play an important role in networked and distributed computing systems. As of today, most file servers operate under a general-purpose operating system such as UNIX. However, a specially-designed operating system for file servers will become the choice of the future since only such a system can be dedicatedly optimized to meet the new challenges in the design of a file server. The new challenges develop as the result of the increasing gap between the CPU and disk speeds [1] and the emerging of new applications such as multimedia and real-time systems. In this paper, we will discuss the design of the Cypress file server operating system that addresses the new challenges with several distinctive features.

One of the main design goals of Cypress is to cope with the increasing gap between CPU speed and disk access time [1]. A recent trend in computer system design in response to this development is to incorporate a large disk cache. In a distributed system, a large disk cache at the file server site can significantly reduce the average disk access latency [2]. A crucial issue emerging along with the incorporation of large disk caches is how to manage the disk cache to achieve a high hit ratio and fast response time. In the design of Cypress, we built the framework by assuming that the hardware installation comprises a very large memory, tens of megabytes to hundreds of megabytes, as the disk cache and a powerful processor for performing the management tasks. The assumption of large-size disk caches allows us to employ a caching strategy that is appropriate only for large-size disk caches. The assumption of the presence of a powerful CPU allows us to increase the intelligence of the management software.

When running on a file server, Cypress enjoys a major advantage over general-purpose operating systems in meeting the challenges of real-time systems. One crucial issue in real-time system design is to provide a predictable response time. In this regard, Cypress has its advantage since it is dedicatedly-designed for file server operations and thus is much smaller in size if compared with a general-purpose operating system. Due to its small size, Cypress can operate without virtual memory support and features more predictably response time.

In the rest part of this paper, section 2 elaborates the major features of Cypress. Section 3 discusses current

*This research was sponsored by National Science Council, under grant NSC 82-0408-E-002-091

status of implementation and future work. Section 4 summarizes the discussion of this paper.

2 Major Features of Cypress

In this section, the major features of Cypress are elaborated. As mentioned earlier, Cypress is distinctive in its caching and management strategies that take advantage of the large-capacity disk caches and powerful processors in contemporary file servers. Hence, the following discussion will focus on Cypress's disk cache management strategies.

1. In Cypress, disk data are cached by the file rather than by the fixed-size block. That is, when a file is cached, its inode and data blocks are all loaded into the disk cache at once. This strategy is aimed at improving the hit ratio of the disk cache. As reported by a research team at U.C. Berkeley [3], most file accesses are sequential and accesses to large files contribute to a significant portion of cache misses. Therefore, if we cache disk data by the file rather than by the fixed-size block, then we can improve cache hit ratio substantially. For example, a sequential head-to-end access to a file of 100 KBytes will incur 25 cache misses if files are cached by the fixed-size block of 4 KBytes. Apparently, we can increase the size of disk cache blocks to reduce cache misses incurred by sequential access to a large file. However, this may cause some sort of waste or overhead because most files are of size less than 10 kilobytes [3]. On the other hand, if the entire file is cached when it is opened, then only one cache miss will occur. This strategy of caching by the file works well with typical files since most of them are less than a few kilobytes [3] and the disk cache is assumed to be of size tens to hundreds megabytes. Nevertheless, special treatment is needed for caching unusually large files, which are commonly found in applications such as multimedia. Caching an extra large file entirely may cause quite a few files that are currently in use being expelled from the disk cache due to the amount of space the extra large file needs. In Cypress, this problem is resolved by breaking down extra large files into smaller pieces called fragments and treating fragments as if they were files in respect to disk cache management. In Cypress, all fragments are of the same size. The optimal choice of fragment size is a function of many factors that can differ from one system to another. For example, the size of the disk cache and the distribution of file size in a particular installation are some important factors.

2. In Cypress, the disk cache contents are periodically, e.g. every 30 seconds, saved into a dedicated backup area on the disk. The allocation of a dedicated backup area was intended for localizing most disk accesses and thus achieving more effective utilization of disk bandwidth. In Cypress, with the assumption of large disk cache capacity, it can be expected that most disk access requests are serviced without actually going to the disk. Therefore, backup of disk cache contents for crash recovery would become the dominant cause of disk operations. By assigning a dedicated backup area, we then can reduce disk head movement and minimize average disk access time. Actually, it is further recommended that a dedicated disk be used for backup. The reason is to make the background writeback, which will be described shortly, function well.

3. In Cypress, we adopted a practice called background writeback to improve the response time of a cache miss. When a cache miss occurs, the system must allocate a space in the disk cache to load the accessed data in. The operation of space allocation normally involves expelling of least recently used data from the cache. If the data being expelled are dirty, then one or more disk writes are needed. In order to avoid the disk writes incurred by replacement of cached dirty data and thus improve the response time upon a cache miss, the background writeback mechanism is adopted. The background writeback works by writing the dirty data that are to be replaced upon next cache miss, say determined by a LRU algorithm, into the file system ahead of time. This operation is referred to as a background one because it takes place only when the disk is not used for servicing explicit I/O requests and has a lower priority. Here, we must clarify the difference between the backup operation and the background writeback operation. They are two separate processes. The backup operation works indiscriminately and periodically to save the disk cache contents into the dedicated backup area for crash recovery while the background writeback process involves only the least recently used data and it steals time from regular operations to write back the selected dirty data onto the file system. Figure 1 illustrates the operations of these two sep-

arate processes. As mentioned earlier, it is recommended that a dedicated disk be used for backup operation. The idea is to make the backup operation and the background writeback operation access different disks. Otherwise, if the disk cache is large, then the backup operation may use up most disk bandwidth and leave no time for the background writeback operation.

4. Cypress is multiple-threaded. The scheduling of I/O requests is based on a prioritized first-in first-served scheme. However, if a request leads to a disk I/O, then the CPU would switch to service the next request with the highest priority in the queue and put the request currently under service into the sleeping state. Once the disk I/O is completed, the suspended request will gain the control of the CPU immediately as long as no requests with higher priority are currently being serviced or are ready for service.

3 Current Status of Implementation and Future Work

As of today, we have implemented the first version of the Cypress file server operating system on an Intel 80486 based PC with a 32-Megabyte memory. The PC serves as the file server in a networked workstation and PC system. Table 1 gives the hit response times of Cypress on such a machine. These times come from a measurement on the Cypress kernel operation. The network latency is not included. As far as the cache miss response times are concerned, they are dominated by the disk access times and therefore are not given here. Currently, we are trying to measure the hit ratio of the disk cache with various sizes and to determine the optimal fragment size to divide a large file in a typical system. These works should be done in the near future.

4 Conclusion

In this paper, we discussed the design of the Cypress file server operating system. The design of Cypress is aimed at improving the file server response time through exploiting latest hardware advances. Cypress is distinctive in its caching and management strategies that take advantage of the large-capacity disk caches, tens to hundreds megabytes, and the powerful processors in contemporary file servers. If compared with general-purpose operating systems that are installed to run file servers, Cypress enjoys another advantage of providing more predictable response time for real-time applications.

Response time to a read request of 4-KByte block	Response time to a write request of 4-KByte block
1.38 ms	1.39 ms

Table 1

References

[1] R. H. Katz, G. A. Gibson, and D. A. Patterson, "Disk System Architectures for High Performance Computing", Proc. of the IEEE, Vol. 77, No. 12, Dec. 1989.

[2] M. N. Nelson, B. B. Welch, and J. K. Ousterhout, "Caching in the Sprite Network File System", ACM Trans. on Computer Systems, Vol. 6, No. 1, Feb. 1988.

[3] M. G. Baker, J. H. Hartman, M. D. Kupfer, K. W. Shirriff, and J. K. Ousterhout, "Measurement of a Distributed File System", Proc. of the 13th ACM Symposium on Operating System Principles, 1991.

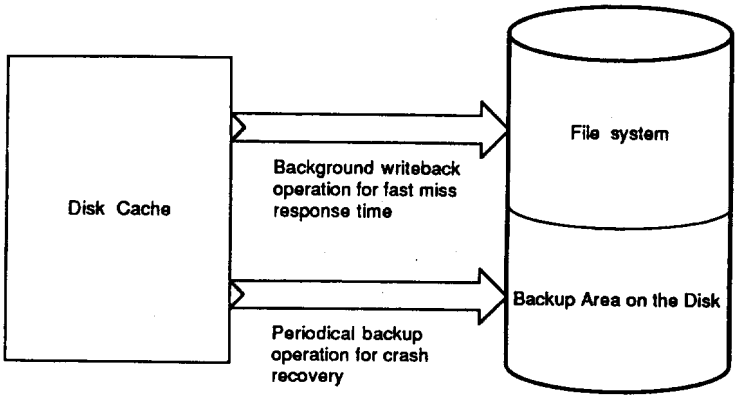
Disk Cache
Background writeback operation for fast miss response time
Periodical backup operation for crash recovery
File system
Backup Area on the Disk

Figure 1