Apple II
Technical Notes

Developer Technical Support

GS/OS
#3: Pointers on Caching

Written by: Matt Deatherage November 1988

This Technical Note discusses effective use of the GS/OS cache.

Introduction

GS/OS is the first Apple II operating system to offer a sophisticated caching mechanism. However, using the cache and using it **wisely** are two different things. This Note presents some concepts which should lead to higher performance for your application if it uses the cache.

What's Cached Automatically?

All blocks on a GS/OS readable disk could be classified into one of two categories. "Application blocks" are all blocks on the disk contained in any file (except a directory file), while "system blocks" are other blocks on the disk. System blocks belong to the file system and include directory blocks, bitmap blocks, and other housekeeping blocks specific to the file system.

GS/OS always maintains at least a 16K cache, even if the user has set the disk cache size to 0K with the Disk Cache new desk accessory. When the system (usually an FST) goes to read a system block, the block is identified as a candidate for caching and is cached if possible. Applications define blocks as candidates for caching by using the `cachePriority` field of many class 1 GS/OS calls. Note that class 0 calls do not have this field, thus applications using exclusively class 0 calls will not be able to cache any application blocks.

Although this difference may seem like a limitation, it in fact improves performance. On the Macintosh, most applications that work with files (like database managers) leave the file with which they are working open while they need it; the file is only closed when the window containing it is closed. Apple II programs historically are quite different—they usually read an entire file at the beginning, modify it in memory, and write it when the save function is selected. A moment's thought will show that if GS/OS arbitrarily cached most or all application blocks, system blocks that would be used again (such as directory blocks) will be kicked out to make room for them. We will see that this is probably a bad thing to do.

How to Cache Effectively

The first tendency of many programmers is to attempt to completely cache any given file, but this usually leads to a degradation in performance, not an improvement. In small caches such strategies can slow the system to a crawl, and large caches offer no significant improvement. Remember that until the cache memory is needed, it is available to the system. The cache size for GS/OS as set by the user is the maximum to be allotted, not the minimum.

Suppose you are attempting to cache a 40K file (80 512-byte blocks). If the cache is set to less than 40K, the entire cache will be written through, kicking out all system blocks currently cached. A cache of this size slows system performance for little gain, since the entire file could not be cached anyway. Even if the cache is large enough to hold the entire file, you are needlessly taking twice the amount of memory with the same file (by reading it into memory you have obtained from the Memory Manager and by asking GS/OS to keep a copy in the cache).

It is evident that the system makes the best use of the cache automatically, freeing your application from the duty of caching system blocks, but there are certain instances where caching application data can improve system performance.

An application which does not limit document size to available memory will often only keep a portion of the document in memory at any given time. Suppose that the beginning of such an application's document file contains a header which to various parts of the document file. (These parts could be chapters for a word processor, report formats for a database manager, or individual pictures for an animation program.) This document header is probably not very long, but the application will likely need to read it quite often to quickly access various portions of the document file.

This header is a prime candidate for caching since it is a part of the file which will definitely be read many times during the life of the application. Contrast this with arbitrarily caching the entire file, which needlessly wastes both cache space and available memory to keep a duplicate copy of something that may or may not be read from disk again.

Although caching provides enormous benefits to GS/OS, indiscriminate use of the cache will waste memory and degrade overall system performance. Be prudent and limit your use of the cache to those portions of your document files which will be read from disk many times.

Further Reference

- *GS/OS Reference*, Volume 1