Mini-Micro Systems

A CAHNERS PUBLICATION

FEBRUARY 1983

VCRs for disk backup

Floppy market overview

5¼-inch Winchester stores 140 Mb

DISK CONTROLLERS

Floppy controller speeds access with cache

STEPHEN GOLDMAN, Distributed Processing Technology

On-board file-management and paging systems keep floppy data in controller RAM

Floppy-disk drives have remained a popular means of data storage despite challenges in the OEM market by Winchester drives and bubble memories. Floppy disks are inexpensive, removable and easily handled, and their storage capacity has been increased through double-density, double-sided and, more recently, 96-tpi drives. Little has been done, however, to improve access times for data stored on this media. Cache-memory and I/O paging have been used to decrease access times for rigid disks, but the added cost of the extra hardware required to implement such schemes has priced most of them out of the range of floppy-disk systems.

A floppy-disk controller introduced by Distributed Processing Technology, Maitland, Fla., uses on-board file-management and paging systems to greatly reduce floppy-disk access times.

Inside the PM-3001

The PM-3001 is a single-board floppy-disk controller with an integrated ROM-based file-management system and capacity for as much as 32K bytes of on-board paging (cache) RAM. With bus adapter card, the PM-3001 controls as many as four 8-in. drives and three 5¼-in. drives. Disk access time is significantly decreased by using the on-board paging RAM, allowing floppy drives to handle many applications in place of higher cost rigid-disk drives. Two DMA channels operate simultaneously to transfer data between disk and the paging RAM and between paging RAM and the host computer memory (Fig. 1). All DMA, host-communications, disk-control, paging and file-management functions are controlled by multitasked firmware running on an on-board 8085 CPU.

The heart of the PM-3001 is the ROM-based paging file-management system. PFMS runs entirely on board the PM-3001, transparent to the host computer system. Because file management normally consumes both processor time and memory space, off-loading this function to the PM-3001 processor can increase total system performance.

All PFMS files appear to the user as 2M bytes of byte-addressable virtual memory space. PFMS allocates and deallocates actual disk space during and between disk accesses. Because files are treated as virtual memory, a user program need not consider record sizes and file-space allocation.

The PM-3001 uses a disk paging scheme to speed data access. Paging schemes use RAM as a fast buffer for a relatively slow I/O device. The paging buffer is often a reserved part of the computer's main memory. The paging area is segmented into blocks or pages that usually correspond to the size of the minimum data record on the I/O device. In traditional disk-paging schemes, each page holds one sector of disk data. When a sector is read from the disk, the data are placed in a page, which then acts as a fast buffer between the disk and the CPU. The next time the CPU requests data, it can be quickly read from the paging area, averting a relatively slow disk access.

Most paging methods are based on the least recently used algorithm (Fig. 2). Each page is assigned an age. When the CPU accesses the data in a page, the page's age is set to zero. All ages are incremented at intervals. When new data must be read into the paging area and all pages are being used, the oldest (least recently used) page is emptied and reused for the new data. Another way to implement an LRU scheme is to link all pages into a list. When a page is accessed, it is removed from the list and relinked as the first page in the list. The last page in the list is then always the least recently used page.

The PM-3001 PFMS links its pages into a tree structure. Each level of each branch of the tree consists of a linked list of pages. By organizing pages into a tree structure, the number of pages to be searched for a data item is reduced, allowing search-and-replacement algorithms to be implemented in firmware. Thus, the PM-3001 can be priced in the same range as conventional controllers.

PFMS file structure

The PM-3001 hierarchical paging scheme reflects the hierarchical structure of PFMS disk files. PFMS files are byte-addressable. Data are accessed by transferring 1 byte to 64K-byte segments between a PFMS file and the host computer's memory. Data can be read from or written to any location within a file, regardless of the history of that file. File addresses range from byte location 0 to byte location 2,097,151, providing each file with 2M bytes of memory space. Although every disk file appears to the user to be in 2M bytes of RAM, only the areas of a file that contain non-zero data require allocated disk space. This is accomplished by structuring each file as a three-leveled tree (Fig. 3).

The root level of the file, consisting of a single disk sector, is the file-header block. Each file header contains statistical information about the file plus 64 sector pointers. Each pointer points to an intermediate level sector or can contain a null value indicating that an intermediate level sectors does not exist for that pointer position. The intermediate level sectors, called bin blocks, each contain 128 sector pointers. Each pointer on this level can point to one of 128 top-level sectors or contain a null value indicating that a top-level sector does not exist for that position in the file. The 8192 top-level sectors contain the actual file data. All other sectors are file overhead and are thus transparent to users.

When a file is created, only the file header block is allocated to the file. All pointers to the next level are set to null. As non-zero data are written into the file, bin blocks and data blocks are allocated so that a three-leveled tree structure emerges. Branches are added to the tree in positions determined by the file

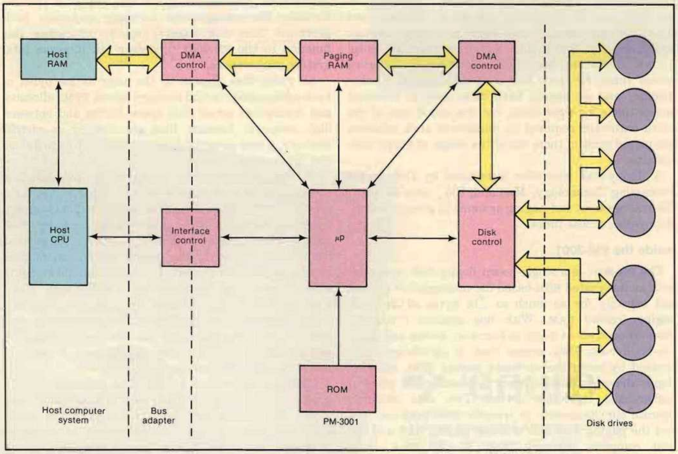

Fig. 1. PM-3001 internal architecture *includes an on-board microprocessor that controls all DMA, host-interface, disk-control and paging functions. Two DMA channels operate simultaneously to transfer disk data between floppy drives and paging RAM, and between paging RAM and host memory. The microprocessor runs multitasked, ROM-based firmware, allowing paging algorithms, disk accesses and host-DMA activity to be overlapped.*

addresses of the data written to the file.

If, in the course of writing to a file, a data sector is found to contain all zeros, that sector is automatically deallocated from the file and returned to the pool of free blocks. The pointer to that sector is set to null. Likewise, if a bin block is found to contain all null pointers, it too is deallocated. Branches grow to accommodate new data and are "pruned" from the tree as the data become zero-filled. Sectors that are deallocated from a file can then be reused by the PFMS operating system to create or expand files.

PFMS page structure

The PFMS paging area is also structured as a tree. In this case, pages are the nodes of the tree, each page buffering one disk sector. The branches of this tree differ in some respects from those of disk files. Each page contains a page header, which, among other things, acts as a link to the other pages in the structure (Fig. 4).

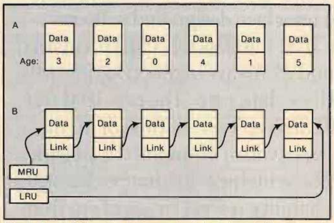

Fig. 2. Two LRU paging schemes *include: (a) assigning an age to each page and setting the age of the most recently used page to zero and (b) a linked list of pages that tracks recency of use by placing the most recently used page at the head of the chain. The LRU page is then always at the end of the chain.*

At the lowest level of the tree is a single page called the root page. The root page acts as a general-purpose buffer for operating overhead and links into the second level of the page tree. At the second level are pages containing disk-header information. Each node at this level represents control data for a disk drive. All pages at this level are chained to form a linked list of nodes. The ends of the chain are linked back to the root page. Whenever data on a disk drive are accesssed, the second level of the page tree is scanned to find the page containing disk-header information for that drive.

If the disk-header page is not in the paging area, it is read into a free page. That page is then linked into the second level as the beginning position in the chain. If the disk-header page is in the paging area, the page is simply relinked into the beginning position in the chain. The ordering of pages in the chain is thus determined by how recently each was used—the first page being the most recently used and the last page being least recently used. Pages can thus be searched, starting with the most recently used page, and the most frequently accessed pages take the least amount of time to find.

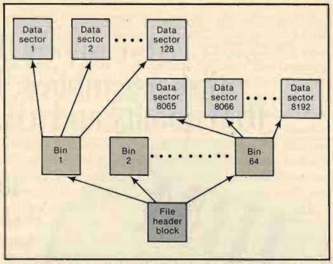

Fig. 3. PFMS hierarchical file structure *includes a file-header sector containing statistical information about the file, and linkage pointers for as many as 64 bin sectors. Each bin sector contains pointers for as many as 128 data sectors. All or none of the data and bin sectors can be allocated for any file.*

The root page acts as a general-purpose buffer for operating overhead, and links to the second level of the page tree.

In the next level of the page tree are pages containing file-header blocks for paging-resident disk files. Each disk-header page in the previous level acts as the root node for one chain consisting of file-header pages for that disk. As before, each chain is connected at its ends to the previous level, and the ordering of pages in the chain is determined by recency of use.

The next two levels of the page tree are composed of bin pages and data pages, respectively. These pages are also chained into linked lists that are connected at the end points back to a node page on a previous level, to which the pages logically belong. Bin pages belonging to a file are grouped and linked into that file's header page. Data pages are chained and linked back to bin pages. The resultant structure is a five-level tree, each page acting as the root node for a sub-tree composed of pages on higher levels. When a page is relinked into the most recently used position in its chain, all pages belonging to its sub-tree get moved along with it.

In a more traditional scheme, paging and file

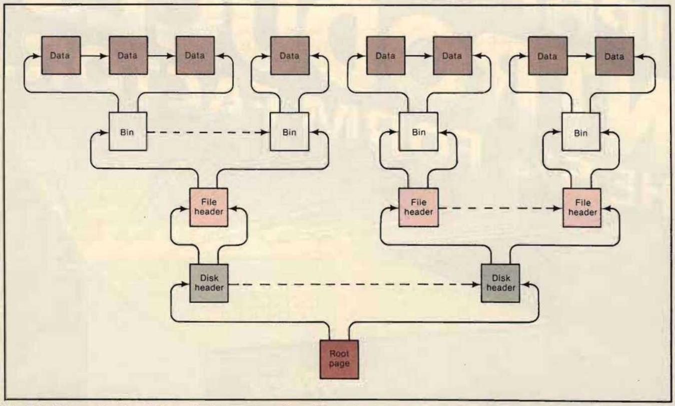

Fig. 4. PFMS page structure *is hierarchical. A page's position within any level of a branch is determined by recency of use. The most recently used page is always at the beginning of each chain. Whenever a page gets relinked, all pages belonging to its sub-tree are moved with it.*

management functions are separate. All file and disk-management overhead sectors must be buffered in main memory, or sector requests must be issued by the file-management software when an overhead sector must be accessed. If the overhead sector has not been accessed for a while or if a long block of data has been read from the disk, the overhead sector may have been paged out, necessitating an additional disk access.

The PM-3001 always pages out the highest level page of the least recently used branch in the page tree; for a complete five-level tree, this is the LRU data page of the LRU bin of the LRU file of the LRU disk. Consequently, pages used to buffer overhead information for other resident pages always remain in the paging area until after the other pages have been paged out.

Disk controllers normally perform best when sectors are read sequentially from a disk. Because of hardware speed limitations, however, disk sectors are usually interleaved so that two full rotations of the disk are required to read all the data on one disk track. This yields a best-case time of approximately 12.8 msec. per 256-byte sector when reading sequential disk sectors from an 8-in. floppy. The PM-3001 can access page resident data at a typical rate of 2.2 msec. per 256-byte sector. This includes all file overhead manipulation, and the file data do not have to be in contiguous disk sectors.

Worst-case disk accesses using a conventional controller take one disk revolution plus the time it takes to read one disk sector. For an 8-in. floppy, this yields a worst-case access time of 172 msec. The worst-case time for accessing page-resident data using the PM-3001 —when the data are located in the least recently used page—is 19 msec., including all file overhead accesses. Data that are not page resident are accessed in times comparable to those of other controllers.

Using the PM-3001 typically reduces access times by a factor of at least six when the file data is page resident. The improvement is greater for 5¼-in. floppy-disk drives. Host memory requirements for file-management software and file overhead buffering are reduced, and more host processor time can be allocated to other resources. □

Stephen Goldman is general manager at Distributed Processing Technology, Maitland, Fla.