CTSS BULLETIN 83

Subject: Timing of Disk File I/O

Date: July 9, 1965

Attachments

AG.2.11 Write-around to new file system
AG.2.02 Buffered Disk Input (V,B,D,READ)
AG.2.03 Buffered Disk Output (V,B,D, WRITE)
AG.2.10 Buffered Input and Output (BFREAD,BFWRIT)
AG.2.06 Library Service Routines

Discussion

The new file system described in CC-241-1 (and -2) which will be installed as MAC2A1 and CTR2A1 uses the disk file storage in ways significantly different from the old file system. These differences can affect the timing of programs in several ways.

First: Two 432 word records can be read in the same time (35 ms) it used to take to read one 465 word record. This can be used to advantage by programs that read data in large blocks (.GE. 864).

Second: Calls to the file system to read or write data merely initiate the I/O; control is returned to the user who may compute while waiting for the completion of the I/O (multiprogramming).

Third: As a price for the generality of a multiprogrammable file system, the overhead for each call to the file system is double the overhead of the old system.

The following table of times required to read a ten-record file from the disk summarizes the nature of interactions between these three effects (1 record = 465 or 432 words for old and new systems, respectively).

Amount of data read per call	Time to read 10 records		time available for computation with new system
	old system	new system	
1 word	9.3 sec	17.3 sec	.47 sec
14 words	1.2	1.7	.47
1 record	.34	.51	.47
2 records	.34	.34	.32
10 records	.34	.17	.16

Time available for computation with old system = 0.0

It should be noted that entries to the supervisor and blocking and unblocking of core-B buffers by core-A routines has always been expensive, partly because of the hardware. Now that the overhead has doubled, it is strongly recommended that all blocking and unblocking be done in core-B and the file system be called to move only large blocks of data (minimum and multiples of 432 words). The library subroutines have been rewritten to provide core-B buffering and most of the public commands have been rewritten to take advantage of the multiprogramming ability as well as the core-B buffering.

Usage

There are several ways a user may approach the problem of timing and the details of the available tools can be found in the attached write-ups.

First: The user need do little or nothing to existing programs. Write-arounds have been provided so that calls to the old supervisor entries will be simulated as closely as possible by core-A programs which in turn will call the new file system (also core-A). The overhead for this sort of operation

may be considered by most users to be excessive. Note also in the write-up of the write-arounds that there are some conditions which can not be simulated.

Second: The user may use the library routines which have been available for some time and which have been rewritten for the new system. These routines are the packages of V,B,D, READ and WRITE. They have been rewritten to accept more buffers if space permits and to do all the blocking and unblocking in core-B rather than core-A. Notice the restriction that, for a given file, calls to the supervisor entries and calls to the library subroutines may <u>not</u> be intermixed. Also note that supplying one buffer to the write routines is the slowest way of writing. Double buffering is strongly recommended.

The library routines for pseudo-tapes have also been rewritten so that programs which use the MAD or FORTRAN I/O statements or programs which already use the V,B,D READ and WRITE subroutines will need only to be reloaded with the new version of the library.

Third: New core-B buffering routines have been added to the library, namely the BFREAD/BFWRIT package. This is a fast core-B blocking and unblocking routine which is smaller and less general than the V,B,D READ-WRITE package. This routine is recommended for programs which are pressed for space and don't need the generality of V,B,D READ etc.

Fourth: The user may call the new file system directly, e.g. RDFILE, WRFILE etc. These entries are available as TIA's from the library or the user may supply his own TIA's. These calls are equivalent to but more expensive than .READK and .WRITE. They should be used only when a user wants to do his own blocking and unblocking. One should avoid using them for reading and writing small blocks. The size of physical records in the new system is 432 words so that these routines can be used efficiently for moving blocks of multiples of 432.