To: Distribution

From: James A. Bush

Date: March 17, 1982

Subject: The Multics Tape Problem

INTRODUCTION AND PURPOSE

Anyone who has ever processed a tape on Multics knows that our tape software is not exactly "Multicious" in nature. There have been several attempts in the past to rectify this situation, by designing improvements to the tape software, mostly in the area of replacing the common tape module, tdcm_. The most notable of these designs was tape_ioi_ (documented in MTB 301, published in 1976 and updated in MTB 383, published in 1978). Due to manpower and budgetary constraints, these improvements have never been implemented. Even if an implementation had been completed for tape_ioi_, only one of the generic problems with our tape software is addressed by its current design.

There are at least two generic problems with our tape software. They are performance, which was addressed by tape_ioi_, and the user interface. This past year, a tape continuum meeting was set up with the express purpose of discussing various aspects of the Multics tape problem. A brain storming session was held on December 10, 1981 to discuss the various problems brought up in the tape continuum meeting and add any others that seemed pertinent. The purpose of this MTB then is to detail the problems brought up in the tape continuum meeting and brain storming session and offer a planned solution for improvement of our tape software.

THE PROBLEMS

Below are the problems brought up at the brain storming sessions (plus some others I have thought of since):

o Multics tape processing speed is to slow

On a recent benchmark (which we lost by the way), the effective rate of a 200 IPS MTU610 tape drive was measured to be 38 IPS when writing data at a density of 6250 BPI. The reasons for this speed discrepancy are many, but it is mostly due to the large amounts of overhead incurred in a users process when writing or reading data to or from tapes. In order to write the contents of a paged segment onto tape, a users process must:

- Reference the desired segment, which if not already known to the process will cause a segment fault to occur.

- If any compaction or formatting of the data is to be done, parts of the segment must be copied into a buffer segment, which would cause a page fault to occur on the desired segment and possibly the buffer segment as well.

- Call the appropriate tape module and the tape module copies the data into one of the available buffers in the "tseg" (a buffer segment shared by all tape modules and lower level tape interface module, tdcm_), after adding any formatting or control information to the data. This could also cause a page fault, since the tseg is paged.

- When enough data is accumulated in a tseg buffer to satisfy the desired physical record size, or when several of the tseg buffers are filled, if the tape module is writing more than one physical record per I/O, the tape module must call tdcm_ to initiate a write to empty the filled buffer(s).

- The tdcm_ module will now copy (again) the buffer(s) from the tseg into the ioi_workspace which is paged and potentially unwired (a feature of the io buffer manager, iobm, keeps the workspace wired for some period of time after an I/O has completed), which could cause a page fault to occur on the ioi_workspace.

- The tdcm_ module now calls ioi_ to issue the physical write request. (This will wire the workspace if it was previously unwired.)

- ioi_ sets up the appropriate tape channel mailboxes and calls the io_manager which finally issues the I/O channel program which will terminate and stop the tape after the I/O is complete.

- After the I/O is initiated, control is returned to tdcm_ which will either (1) go blocked if in sync mode or (2) return to his caller if in async mode, allowing the caller to fill another buffer or set of buffers.

- Assuming async mode, when the next set of buffers is filled, the tape module calls tdcm_ again which must now go blocked and await the completion of the previous I/O.

- When ioi_interrupt gets the terminate interrupt from the previous I/O, an ips_ wakeup signal is generated, which is caught by tdcm_ (if the users process is still eligible) because he is sitting blocked waiting for it.

- The tdcm_ module now makes a cursory check of the terminate status and if it is judged to be ok, calls ioi_ to issue the write for the next set of buffers.

- Since the tape motion had stopped with the last terminate, several milli-seconds must now be expended by the tape drive to get up to speed before any data is actually written on the tape.

- This process continues in more or less the same fashion until the desired data is written on the tape.

One can see from this scenario that there are really two problems which effect tape performance:

(1). Data is copied too many times, not only incurring the overhead of the actual data copy, but potentially causing page faults and the associated overhead in processing them.

(2). Tape motion stops when each I/O is complete, which not only incurrs some finite amount of overhead by the tape drive to come up to speed when another I/O is issued, but if the users process had become non-eligible due to going blocked and waiting for the tape I/O to complete, tape motion will not be initiated again until the users process is put back into execution by the traffic controller.

The performance problems associated with reading data from a tape into a segment are pretty much the same as that for a write, but in the reverse direction. Therefore I will not detail the read segment scenario.

o The tape I/O modules are basically unmaintainable

The iox_ compatible io modules, tape_ansi_, tape_ibm_, tape_mult_, ntape_, and tape_nstd_ were written by Multicians that have long since departed the Multics development group, and were written in less than a structured format. This makes them difficult for a new person to understand and next to impossible to correct the many bugs that exist within them. This is also true of the ios_ compatible module, nstd_ (still used by the GCOS simulator and tape_nstd_), and the tape device control module, tdcm_.

o Some tape modules are missing

There are currently no tape modules that support some of the common tape formats such as GCOS standard, GCOS UFAS, CP5/6, and GCOS 64. Although some of these are similar to ANSI standard, there are enough differences so that tape_ansi_ will not process them adequately.

o Little support for stranger tapes

Currently, the only stranger tape processing capability we have, is the interactive command, read_tape_and_query (other than individuals private tools). The read_tape_and_query command does a fairly good job of allowing a stranger tape to be inspected by reading records and dumping their contents, and has a limited repertoire of canned tape formats which it can process, once the format is determined. But in many cases this is not enough. Many times, a stranger tape will be encoded in some non-standard format, (e.g. character data encoded in an "extended" BCD or ASCII character set, some of the characters of which have no equivalent in the Multics Ascii character set), or character and binary or hexadecimal data concatenated in the same record.

o Tape error recovery is inadequate

All of the current tape modules do their own error recovery, instead of having consistent error recovery procedures centralized in one place which would logically be tdcm_ in todays tape software. The tape_mult_ module even implements its own unique (in the industry) write error recovery by simply re-writing the record in error without backspacing and erasing over the bad spot on the tape. With todays high density data encoding techniques used on our tape subsystems, this type of error recovery is ill advised at best.

Our current tape subsystems have many hardware features to aid in the essential task of error recovery. The tape modules currently have no interface to use these features and therefore must rely on the traditional backspace/retry type of error recovery, which is not always adequate to recover marginal data written at high densities.

o Large number of outstanding trouble reports on the tape software

There are approximately 100 open TRs that currently have no resolution. The reason for this (besides the obvious, buggy software), is that no one has been assigned to maintain the tape software for sometime. Bugs have only been fixed recently, because some individual developer became interested in a particular bug and took it upon him/herself to fix it.

o Most tape modules exhibit a poor user interface

The user interface to most of the tape modules is in general inconsistent and restrictive. Some tape modules use a "-ring" attach description argument to specify that the tape is to be mounted with a write ring, while others use "-write" for the same purpose. The tape_ansi_ and tape_ibm_ modules in particular are to restrictive in their enforcement of their respective standards. The ANSI tape standard specifies that the maximum block size supported is 8192 bytes. The tape_ansi_ module supports this rigidly, even when a user has a tape that otherwise meets ANSI standards, but has a block size that is greater than 8192 bytes.

There is also no concept implemented which allows default values to be inserted for attach description arguments that are omitted. This is particularly true of the tape_ansi_ and tape_ibm_ I/O modules, making their use fairly frustrating when little is known about the format of a particular tape volume. This is especially true when a person is not that familiar with what exactly has to be specified in a tape_ansi_ or tape_ibm_ attach description (after a user once reads the voluminous writeups on these modules, he/she is usually confused as to what exactly is required), since the error diagnostics from these modules leaves much to be desired.

o RCP does not pass on tape drive and volume info

In the course of tape volume authentication, RCP learns many things about a particular tape volume such as: tape volume recording density and format type (i.e. IBM, ANSI, GCOS, Multics standard, or Unlabeled). RCP also has speed and density capability information available on the selected tape drive. Unfortunately, there is currently no way for RCP to pass this valuable information on to the IO module that has requested the tape attachment. This forces each tape module to repeat the procedure of validating the label to see if the requested tape volume is correct as far as format type and tape reel number.

o General lack of tape utilities

There are very few tape utilities available on Multics. There are tape utilities meant for specific tasks, such as copy_mst and copy_dump_tape, which are used to copy Multics system and release tapes for shipment to the field, but there are no general logical or physical tape copy routines available. There are also some utilities for reading data from a tape into the Multics file system, and writing data from the Multics file system to a tape (e.g. tape_in/tape_out, copy_file, and tape_archive). But there is no simple command which will write or read a tape, dumping information to or reading information from a tape, in ANSI standard tape interchange format.

o The iox_ IO system is not particularly suited for tape IO

The current Multics IO system, iox_ is byte oriented and works with a file as a single entity. Data on a tape on the other hand, may be nine bit byte, eight bit byte, six bit character or 36 bit word oriented and may be contained on several different tape files, each of which may have its own unique format. In iox_ terminology, a file is "attached to an IO module" and then "opened" for reading or writing in one of several different modes. When doing tape IO, this involves not only attaching an IO module, but also the physical mounting and positioning of the tape volume on an assigned tape drive (this assignment is implicit by default), with the desired tape file name or number being specified in the "attach description". The tape is then "opened" for reading or writing and IO operations are begun on the desired tape file. IO continues until an "End of File" condition is reached, at which time a user would "close" the file. If the user wanted to process the next sequential tape file, one might logically think the only thing that would be required would be to "open" the next file, but due to the fact that the file name/number is part of the "attach description", the user must first "detach" the IO module and re-attach the same IO module to process the next file. Fortunately, all tape IO modules that support multiple file formats, also support an attach description argument known as "-retain". The -retain argument allows the user to detach an IO switch and reattach the same IO switch, without requiring that the tape volume be demounted and remounted and repositioned to the next sequential file. However, I maintain that the "-retain" attach description argument is only a "kludge" to get around this weakness in the iox_ IO system.

Because of the reasons mentioned above, iox_ does not support physical file (and physical record within a file) positioning. It would be much more convenient and less costly if, after determining that the current file was not what the user wanted, he could cause a forward space file command to be executed by the tape software and read from the next file. Currently the only kind of positioning that is supported is "type 3" or relative character positioning. And this is implemented by simply reading data forward until the desired position is reached. This particular function is one of the biggest performance problems with the volume_retriever today.

o Tape labels do not meet FIPS standard

  Tape labels generated by tape_ansi_ are not in compliance with latest FIPS/ANSI specification.

o New software subsystems will put new requirements on the tape software

The New Data Manager (NDM), currently being designed by MSD, and the "CRAY Connection" subsystem, being designed by HISUK, both have unique tape processing requirements. The NDM will use tape output for "after image" database journalization, which will require almost real time tape response and may require multi-process access to the journal tape for Transaction Processing applications. The Cray attached array processing subsystem, will require lower ring tape attachment for doing "backup" dumping of the Cray operating system, through a high speed data link, which may require almost real time tape processing speeds.

THE SOLUTIONS

I believe the ultimate solution to the tape problems stated above, is a complete overhaul of the tape software. Most of the performance problems could be taken care of by finally implementing most of tape_ioi_ as it is currently documented in MTB 383. As far as solving the problems associated with the user interface, I feel a completely new approach should be undertaken.

This new approach will be the design and implementation of a new tape module that I have named "mtape_". Although the technical details of the design of mtape_ have not been completed and will be the subject of a future MTB, a thumbnail sketch of mtape_ might be helpful to the reader at this time.

The basic idea behind the design of mtape_ is the premise that only one tape module is required to meet the needs of all Multics tape processing, if that tape module is designed with flexibility and extensibility in mind. When a tape volume is opened for reading, mtape_ will obtain information as to its format type from RCP (i.e. ANSI, IBM, GCOS, Multics standard, UNLABELED, etc.). This information will be used as a key to call the appropriate tape label processing routine. A default mechanism will assign reasonable default values for any information that could not be obtained from the label record or explicitly from the users invocation of mtape_ (e.g. if RCP indicated a GCOS tape, the default values would be: Format = VB, Block = 1284 bytes, reading mode = binary).

Using mtape_ for tape output, the user may specify as much or as little as he wishes, pertaining to the output tape format. Again the default mechanism will fill in the blanks. If the user specified nothing at all about the output tape format, the default mechanism would set up the format as an ANSI standard interchange tape (i.e. Format = DB, Density = 800 BPI, Track = 9, Block = 2048 bytes, recording mode = nine).

SHORT TERM SOLUTIONS

Since tape_ioi_ and mtape_ have not yet been implemented and by current estimates will take 24 man months to complete, what, if anything, can be done in the interim to improve tape performance and the user interface? Several inexpensive short term "fixes" present themselves, for implementation within the MR10.0 time frame:

o Perform metering on the tape software

  By using the metering tools available on Multics, such as trace, profile and cumulative_page_trace, we can find out where the performance bottlenecks are. If some of these bottlenecks turn out to be inefficient coding techniques, the code can be tightened up in these areas.

o Quick fixes to the traffic controller

  A process using tapes could be given retained eligibility after that process goes blocked and a short, high priority time slice upon receipt of a tape interrupt.

o Pick up support for tape_gcos_

  The Air Force Data Service Center (AFDSC) has written an iox_ compatible GCOS tape module which they have offered to HIS in return for continued support and bug fixes for this module. We could install this module for use by other Multics customers such as Ford and Bell Canada, who have heavy GCOS to Multics (and visa-versa) program transport requirements.

o Relax tape_ansi_ standards for reading

  We should remove the 8192 byte block size limit within tape_ansi_. Several sites have already done this in local modifications to tape_ansi_. If MSD made this mod to tape_ansi_, it would relieve sites from making this mod when new releases are sent out.

o Re-write tape_mult_ error recovery.

  We should re-write tape_mult_write_ error recovery to perform error recovery by backspace/erase/re-write instead of re-writing the error record as is done today. This would greatly improve reliability of system tape applications, such as the volume_dumper and the hierarchical dumper systems. Note that tape_mult_read_ error recovery was re-written for MR9.1 to do backspace/re-read error recovery.

o Install new version of tape_in/tape_out

  We should complete and install a new version of tape_in/tape_out that was left about 90% complete by a recently departed member of the Multics development staff. This would fix the many outstanding bugs and user complaints of this tape processing utility.