Appendix B

Performance Notes for itsibmtape

The **itsibmtape** command is designed to maximize throughput between an IBM 3480−compatible tape system and the SDA. This appendix explains how to make best use of **itsibmtape** performance optimization features.

The principal reason performance tuning may be beneficial is that tape systems of the 3480 type employ streaming tape access methods. This means that the tape system is capable of reading and writing data as fast as the tape passes beneath the tape heads. However, if the system runs out of data to write, it will not stop tape motion immediately, and will be forced to rewind the tape past the point that it stopped I/O and bring the tape up to speed again before resuming data transfer. If the tape unit is "starved" for data when writing a tape, or if the user's peripheral is not able to receive data as fast as the tape unit provides it, a greater amount of time will be spent repositioning the tape.

To solve this problem, the **itsibmtape** command employs a system known as *double buffering*. When the user specifies a buffer size with the **bs** option, two of these buffers are actually allocated in the ITS buffer memory. When reading from tape, data is read into one of the two buffers. When that buffer has been filled, the tape unit starts depositing data into the second buffer, and the first buffer is used to write to the SDA. As the tape unit completes transfers, these buffers are repeatedly switched, ensuring that the tape is kept in more or less continuous motion. A similar process is employed for writing, with the SDA filling one buffer with data while the tape unit is removing data from the other.

An ITS board makes 4 Mbytes of an 8-Mbyte buffer memory available for use by **itsibmtape**. Since **itsibmtape** double-buffers its data, the **bs** option should be set as close to 2 Mbytes as possible (1/2 the available buffer space). The buffer size must also be a constant multiple of the physical record size.

Copyright © 1994 Thinking Machines Corporation

Performance of the tape system may also be improved by selecting a data format that is optimum for the tape. Not all data formats are equal. In general, the various fixed-record formats (**f**, **fb**) can be processed much faster than the variable formats (**v**, **vb**, **vbs**, and **u**). Because the UNIX file system does not support structured files, additional processing is performed on variable-format files as they are read from the tape system.

Variable-format files contain information in the data stream indicating the start and end of records. Most application programs will want to remove or skip this data when processing the file, so the **itsibmtape** command creates an index into the data set, called a *control file*, as it is read. This control file is created with the same name as the data set with an extension of **.ctl**. The process of creating this file can be time-consuming, as it requires scanning the entire input buffer before it is written to disk. Double buffering provides extra time for **itsibmtape** to scan the buffer, but extremely short variable-length records can still cause considerable performance degradation. If a variable-length data format is being used to store primarily fixed-length data, it would be advantageous to switch to a fixed or unblocked format if at all possible.

Short blocks in the data stream are another potential cause of performance degradation when reading from a tape system. **itsibmtape** attempts to use *fixed-block mode* when reading data from the tape, assuming that this will achieve the maximum transfer rate possible. If, during the read operation, the tape system encounters a block that is shorter than the given block size, it will be unable to continue the read and will return a count of the number of blocks actually read to **itsibmtape**. **itsibmtape** must then take the drive out of fixed-block mode, back the tape up one record, reread the short record, return the drive back to fixed-block mode, and resume the read operation. If this occurs frequently, it can impose a considerable performance penalty. Since the block size for fixed-block mode is specified by the block size in the **rformat** option (or in the tape label) it is important for this value to indicate accurately how data are blocked on the tape.

It is important to note that even "fixed" format tapes are not exempt from having short blocks written on them. The "fixed" term refers to the length of the user records in the data, not the length of the blocks. It is therefore possible for a tape to have an **rformat** of **fb:8000:80** (record length of 80 bytes, block size of 8000 bytes, 100 records per tape block) where every other tape block contains 99 records. A data layout of this nature can considerably degrade performance. Unfortunately, such considerations are largely governed by the system that produced the tapes being read.

Version 7.3, April 1994
Copyright © 1994 Thinking Machines Corporation