Digital Technical Journal
Volume 6, Number 2
DLT2000 paper

Analysis of Data Compression in the DLT2000 Tape Drive

by

David C. Cressman

ABSTRACT

The DLT2000 magnetic tape drive is a state-of-the-art storage product with a 1.25M-byte-per-second data throughput rate and a 10G-byte capacity, without data compression. To increase data capacity and throughput rates, the DLT2000 implements a variant of the Lempel-Ziv (LZ) data compression algorithm. An LZ method was chosen over other methods, specifically over the Improved Data Recording Capability (IDRC) algorithm, after performance studies showed that the LZ implementation has superior data throughput rates for typical data, as well as superior capacity. This paper outlines the two designs, presents the methodology and the results of the performance testing, and analyzes why the LZ implementation is faster, when the IDRC hardware implementation had twice the bandwidth and was expected to have faster throughput rates.

OVERVIEW

Data compression, a method of reducing data size by coding to take advantage of data redundancy, is now featured in most tape drive products. Two compression techniques in widespread use are (1) an arithmetic coding algorithm called Improved Data Recording Capability (IDRC) and (2) variants of the general Lempel-Ziv (LZ) compression algorithm. Current tape products that implement these algorithms are IBM's fast (a maximum throughput rate of approximately 3M bytes per second [M bytes/s]) and relatively expensive (originally about $60K) family of half-inch, 36-track tape products, which have employed the IDRC algorithm for about five years. More recently, the 8-millimeter (mm) helical scan tape products began incorporating IDRC data compression. Also, some 4-mm helical scan digital audiotape (DAT) products now use a variant of the LZ algorithm, as do some quarter-inch cartridge (QIC) tape products.

In developing a complex product like an industry-leading tape drive, it is difficult to determine at the beginning of the

project the design that will have the best performance characteristics and meet time/cost goals. When Digital included data compression in the plans for its DLT2000 tape product, the choice was not clear regarding which compression technology would best enhance the tape drive's data transfer rate and capacity. Keeping within cost constraints and incurring an acceptable level of risk to the development schedule were important factors as well. The options were greatly limited, however, because the schedule was too short for the engineering team to implement a compression method on a silicon chip designed specifically for the DLT2000 tape drive; therefore, the team needed to find a compression chip that was available already or would be soon.

Another important consideration was that the compression method used on the DLT2000 tape drive would likely be used on future digital linear tape (DLT) products. For media interchangeability, such products would have to be able to write and read media compatible with the DLT2000 tape drive. New products that used different compression methods would require extra hardware to handle both types of data compression. Since extra hardware adds significant cost and complexity to products, the use of different compression methods is undesirable. Also, to meet future data throughput needs, the compression method used on the DLT2000 tape drive had to support the significantly higher data transfer speeds planned. If the compression chip used initially was too slow for future products, it had to be at least possible to develop an implementation of the same compression algorithm that would be fast enough for future DLT products.

To gain more expertise in applying data compression technology to tape drives, the tape development group investigated several designs using various data compression chips. Eventually, we created about 20 DLT2000 engineering prototype units, each of which used one of the two most common data compression methods: IDRC and an LZ variant. The specific Lempel-Ziv variant used was designated Digital Lempel-Ziv 1 (DLZ1).[1,2] We tested the performance of the prototype units and studied the results to check for consistency with our expectations. Such analysis was important since tape drive performance with data compression was a new area for the engineering team, and the interplay of higher tape transfer rates, new gate arrays, compression chip, memory buffers, new firmware, and host tape applications is complex.

Figure 1 shows the basic design of the data path on the DLT2000 tape drive's electronics module. (Microprocessors, most gate arrays, firmware read-only memories [ROMs], and other electronic components are not shown.) Note that the data cache size is effectively increased because it contains compressed data. The data processing throughput of the compression chip, however, can potentially be a bottleneck between the cache and the small computer systems interface (SCSI) bus. The IDRC compression chip can process data at throughput rates of up to 5M bytes/s, whereas the DLZ1 chip can process data at rates of up to about 2.5M bytes/s when compressing data and up to about 3M bytes/s when

decompressing data. In each design, the memory and data paths outside the compression chip were designed to be adequate for the compression chip used.

[Figure 1 (Tape Drive Data Path) is not available in ASCII format.]

One major goal of this study was to quantify the performance of each implementation to determine if the lower throughput of the DLZ1 chip was a practical disadvantage in the DLT2000 product. The IDRC version of the DLT2000 product, with its maximum throughput rate of 5M bytes/s, would seem to have a clear throughput advantage, but the typical compression ratio and the data rate to the tape media are significant factors in the overall throughput of the tape drive.

The development group expected the IDRC and DLZ1 chips to have approximately the same compression ratio (i.e., the result of dividing the number of units of data input by the number of units of data output). The DLZ1 ratio would possibly be slightly higher. The group based their expectation on comparisons of results from several studies.[2,3,4] These studies reported compression ratios for various types of data on implementations that used either the IDRC algorithm or an LZ algorithm but not both.

Compressing data within the tape drive has a multiplying effect on the drive's throughput rate, as seen by a host computer. If the uncompressed data throughput rate to the tape media is 1.25M bytes/s and the data compression ratio is 2.0:1 (or 2.0), the expected average data transfer rate is 1.25 x 2.0 = 2.5M bytes/s. Since the development group thought that the typical compression ratio of each implementation was 2.0:1, and because the DLZ1 chip would tend to become a bottleneck as data rates approached the chip's maximum throughput rate, the group expected the IDRC prototype to be at least as fast as the DLZ1 prototype for a given data set.

Testing showed, however, that the DLZ1 DLT2000 prototype consistently, and significantly, surpassed the IDRC prototype in both metrics! To ensure the correctness of the IDRC implementation used on the prototype DLT2000 and thus confirm the unexpected result, the group verified the IDRC compression efficiency results by testing two other tape products that use the IDRC algorithm. Given identical data sets, the benchmark test results were consistent with those of the IDRC DLT2000 prototype.

The marked difference between the DLZ1 and IDRC prototypes can be mainly attributed to the differences in the compression efficiencies of the two algorithms. Relatively low compression ratios on the IDRC unit limit its throughput capabilities. The author believes that the discrepancy between the results of the DLT2000 prototype testing and the results of the earlier studies can be explained by two factors: variations in the data sets used

and differences in media format.

First, the compression efficiency for different samples of data, even if of the same type, e.g., PostScript data, can vary widely. The data sets tested on the DLT2000 prototypes were not identical to those tested in the earlier studies.

Second, some tape drive implementations combine IDRC data compression with a feature IBM calls autoblocking (also known as superblocking). This coupling occurs when the tape drive has a media format that contains interrecord gaps (IRGs) whose number is inversely proportional to the tape block (record) size used (sometimes linear). Autoblocking minimizes the number of IRGs by automatically using a large, fixed on-tape block size (e.g., 64K bytes). The autoblocking feature packs multiple compressed blocks from the host into the larger blocks on the media.[4] Reducing the number of IRGs on such tape formats is important because IRGs are wasted space. If block sizes are small, the number of IRGs will be large and the tape capacity significantly reduced. Tape products that combine autoblocking with IDRC compression derive an increased capacity from both techniques.

These two factors, however, were not relevant to the test results of our study, i.e., the favorable DLZ1 findings. We performed the DLT2000 prototype testing with tape drives that were virtually identical except for the compression technology used. Also, the data samples, tools, and test environments were the same.

From the test results and analysis we concluded that, when compared with the IDRC implementation, the DLZ1 implementation combines consistently superior cartridge capacity (25G bytes at a compression ratio of 2.5:1) and superior data throughput for most types of real data. The testing did not reveal any real data types that compressed better with the IDRC technique than with the DLZ1 technique. In addition, the DLZ1 technique is supported by the strong prospect of future DLZ1 compression chips that will greatly increase the maximum data throughput rates. This addresses the concern that the DLZ1 technique should support a growth path in data throughput rate for future members of the DLT product family.

The remainder of this paper outlines the operation of the IDRC and DLZ1 compression techniques, discusses what testing was done and how, presents the test data, and gives an analysis of the results.

DESCRIPTION OF THE IDRC AND DLZ1 COMPRESSION ALGORITHMS

This section provides some historical/industrial background on the IDRC and DLZ1 algorithms and some cursory information on how they work. An in-depth technical presentation of these (or other) compression techniques is beyond the scope of this paper. For more details on their operation and mathematics, please refer to

the references.

The IDRC Compression Algorithm

IBM developed the IDRC algorithm and employs this technique on some members of the Model 3480 and Model 3490 tape subsystems. EXABYTE Corporation is currently licensing the IDRC algorithm from IBM.[4]

The IDRC algorithm is a lossless, adaptive arithmetic compression technique. Arithmetic compression encodes data by creating an output string that represents a sequence of fractional numbers between 0 and 1. Each fraction is the result of the product of the probabilities of the preceding input symbols.[4,5,6,7]

The IDRC technique has two modes: byte oriented and binary (bit) oriented. On input, bytes are compared with the last byte processed. If three or more consecutive bytes are found to be equal, processing occurs on a byte-by-byte basis. Otherwise, the data is compressed bit by bit.[6]

Parallel recording implementations for which the number of IRGs is a capacity issue (for example, the IBM Model 3490 product) usually combine IDRC compression with autoblocking. Since autoblocking reduces the number of IRGs (assuming that a smaller block size is commonly used), the effective increase in tape capacity due to autoblocking surpasses the increase that compression alone would yield.

In some tape implementations, though, data is packed into fixed-size blocks on the media whether or not compression is used. If done efficiently, this packing makes tape capacity on such products independent of block size.

The DLZ1 Compression Algorithm

A number of variations of the Lempel-Ziv algorithm (also referred to as the Ziv-Lempel algorithm) have been implemented and are in wide use in the industry today. Some examples are the common PC compression software tools PKARC, PKZIP, and ZOO; the compression method built into the MS-DOS Version 6.0 system; and Hewlett-Packard's HP 7980XC tape drive. IBM recently announced that it has developed a high-speed (40M bytes/s) compression chip that uses the LZ algorithm. In addition, STAC Electronics' data compression products and the QIC-122 data compression standard use derivatives of the LZ algorithm.[4,5]

Lempel-Ziv methods generally replace redundant strings in the input data with shorter symbols. The methods are lossless and adapt to the input data. Implementations typically simplify the general algorithm in one or more ways for practical reasons, such as speed and memory requirements for string storage.[1,3,4,5,8]

The LZ variant used in the DLZ1 implementation maps variable-length strings in the input to variable-length output symbols. During compression, the algorithm builds a dictionary of strings, which is accessed by means of a hash table. Compression occurs when input data matches a string in the table and is replaced with the corresponding dictionary symbol. The dictionary itself is not output to the tape media but is rebuilt during decompression.[1]

When the dictionary fills up with strings, the algorithm cannot adapt to new patterns in the data. For this reason, the dictionary needs to be reset periodically. The DLT2000 DLZ1 algorithm resets the dictionary on each logical block boundary. Thus, the compression efficiency can vary according to the block size, as well as with the actual data. With small blocks, the dictionary is typically still adapting to the input data when the block ends and the dictionary is reset. This tends to keep the compression algorithm from reaching full efficiency. For example, with an LZ variant similar to the DLZ1, the LZW algorithm presented in Welch's "A Technique for High-Performance Data Compression," compression efficiency increases rapidly as the block size used goes from 1 byte to about 8K bytes.[3] The efficiency peaks at about 12K bytes, and larger block sizes show good but gradually decreasing compression efficiencies. The initial input block range that exhibits rapid improvement in compression efficiency (1 byte to 8K bytes, in this case) is referred to as the "adaptation zone."

TEST PROCEDURES

The development group carried out three main sets of tests.

1. Tests that measured the compression efficiency on an OpenVMS system and on an ULTRIX system, which is based on the UNIX system

2. Tests that measured the compression efficiency and the data throughput in a high-throughput test system environment

3. Benchmark tests that measured the IDRC compression ratios on two other tape products

The DLT2000 firmware measured the compression ratios precisely by comparing the block size (in bytes) before and after compression, during write command processing. In the benchmark tests, compression ratios were calculated from total tape capacities with and without compression enabled. We repeated the DLT2000 tests with minor variations in test parameters; the results suggested an uncertainty of approximately +/-1 percent in the measurements.

Test configurations were identical in system type, test software, and operating system versions. We often used the same test bed and varied only the tape unit under test, i.e., the DLZ1 or the IDRC. The hardware and firmware on the different DLT2000 prototypes were identical to ensure that factors such as diagnostic code overhead and clock speed did not skew test results between the DLZ1 and the IDRC units, or between test runs. We also varied some parameters and repeated tests to ensure that the measured performance characteristics were consistent with and reflective of the final product.

Operating System--based Tests

Since the system configurations used could not supply data fast enough for conclusions to be made regarding the DLT2000 tape drive's maximum throughput rates, compression efficiency was the focus of the operating system testing. Test parameters were still chosen to minimize throughput bottlenecks in the host system. For each test, the data was set up on a single disk on each of two systems -- an OpenVMS system and a UNIX system.

OpenVMS Tests. The OpenVMS system used in the tests was a clustered MicroVAX 3400 machine with a KZQSA adapter for the SCSI bus. The MicroVAX 3400 system was running the OpenVMS Version 5.5-2 operating system and used the standard backup utility (BACKUP) to write data to the DLT2000 tape drive. Although compression efficiency was the focus of the operating system testing, we selected the following BACKUP options to maximize system throughput as much as possible:

- /NOCRC. This option disables a cyclic redundancy check (CRC) calculated and stored in the tape block by BACKUP for extra data integrity protection. Since the CRC calculations are CPU intensive, they were disabled to minimize system bottlenecks.

- /BLOCK_SIZE=65024. A block size of 65,024 minimizes host and SCSI bus overhead to a reasonable degree.

- /GROUP_SIZE=0. This option disables the creation of (and the writing to tape of) an exclusive OR (XOR) block calculated by BACKUP. By default, BACKUP would create one XOR block for every 10 data blocks. We disabled XOR blocks because their presence would probably decrease the compression ratio and system throughput.

We tested the following types of data on the OpenVMS system.

- Bin -- the BACKUP of a set of binary files, mainly executable files

- Sys -- the image BACKUP of the system disk

- C -- the BACKUP of the DLT2000 product's firmware source library, primarily C code and include files

UNIX Tests. The UNIX configuration used for testing was a DECsystem 5500 system running the ULTRIX Version 4.2c operating system. The SCSI common access model (CAM) software driver was used, running on this machine's native SCSI port. The standard ULTRIX tar and dd utilities were used to copy the following data to the tape:

- Text -- ASCII text files of product documentation manuals
- PS -- PostScript versions of the manuals
- tar -- tar backup of the system disk
- HarGra -- the chart and art files shipped with the standard Harvard Graphics software package
- ValLog -- the files containing the gate array design database, which was built using Valid Logic tools

Throughput Tests

The throughput tests were performed on PC-based Adaptec SDS-3 SCSI development/test systems. The development team chose this test environment to do repeatable, high-performance testing because it is relatively unconstrained by disk, file system, CPU, or application software bottlenecks for the performance range of the DLT2000 tape drive.

We tested the following data types on the SDS-3 system:

- Binary -- an OpenVMS VAX object file
- Source -- C source code
- VAXcam -- a VAXcamera image file in PostScript format
- HarGra -- a collection of chart and art files shipped with the standard Harvard Graphics software package
- Paint -- a complicated Paintbrush file, in bitmap format
- Ones -- an all ones (hex FF) pattern
- Repeat -- a string of 24 unique characters, repeated as needed

SCSI bus protocol overhead can be somewhat high on an SDS-3 system, and compression ratio and throughput rate can vary

depending on the tape block size. Consequently, all measurements were taken using 64K-byte tape blocks. This block size minimizes per-command overhead on the SCSI bus, as well as in the host. With high enough compression ratios, however, this overhead was still a limiting factor for 64K-byte blocks on the IDRC testing, as will be shown later in the SDS-3 Test Results section.

Another factor in SCSI bus performance is whether synchronous or asynchronous data transfer mode is used. Asynchronous transfer mode requires a full handshake to transfer each data byte, which can seriously decrease the bandwidth of the SCSI bus in many configurations. Synchronous transfer mode (period/offset = 200/7) was enabled, which tends to minimize the effect of cable length on performance.

For a given data type, the same amount of data, i.e., from 50M bytes to 300M bytes, was transferred to both versions of the tape product. We often performed several test runs using different amounts of data to check the consistency of the test results.

To maximize the applicability of the test results, we wanted to use "real world" data. To do so in our test environment was not practical or would have introduced delays between blocks, thus ruining any throughput measurements. We obtained a compromise in the following manner. The SDS-3 tool we used is limited by a 64K-byte buffer for high-speed transfers. That buffer can be used repeatedly, and the direct memory access (DMA) pointers automatically "wrap around" back to the start when they reach the end of the buffer. We created a tool that takes the first 64K bytes from a file with the desired test data, reformats the data, and writes the data to an output file compatible with the SDS-3 software. This "buffer file" can then be uploaded into the SDS-3 tool's memory buffer, thus duplicating the first 64K bytes of the data from the test file in SDS-3 memory. The tool has an obvious limitation; the first 64K bytes of data might not be representative of the rest of the data in the file. Using this tool was, however, a practical way to transfer at least subsets of real data into the throughput test environment.

Benchmark Tests

Since preliminary results of our study indicated that the IDRC chip has a lower compression ratio than that indicated by previous studies, the benchmark tests were performed primarily to confirm the compression efficiency of the IDRC DLT2000 implementation.[4] For the benchmark tests, we tested two tape products that use IDRC compression implementations.

The first product tested was Digital's TA91 tape drive (which is compatible with an IBM 3480E tape drive) configured on a Hierarchical Storage Controller (HSC) in a VAXcluster configuration. A collection of chart and art files included with the standard Harvard Graphics software package composed the data

set. This identical data set was written to an IDRC DLT2000 tape drive for accurate comparison.

The second benchmark product tested was an EXB-8505 tape drive, which also uses IDRC compression.[9] We tested the EXB-8505 tape drive on an SDS-3 test system. The data set used was the first 64K bytes of the text of the U.S. Constitution. We compared the compression ratio obtained on the EXB-8505 with the compression ratio for the same data written to a DLZ1 DLT2000 unit and with text data compressed on an IDRC DLT2000 tape drive. (The text data on the IDRC implementation was different from the text data on the EXB-8505 and DLZ1 implementations because an IDRC prototype was no longer readily available when the U.S. Constitution data became part of the tests.) We also performed some throughput tests to compare the DLZ1 DLT2000 and the EXB-8505 drives.

We measured the native product capacity of the TA91 and EXB-8505 tape drives by writing to the end of tape (EOT) with compression disabled. We then repeated this test with compression enabled.

TEST RESULTS

The compression ratios shown in the test results are calculated by dividing the number of bytes of uncompressed data by the number of bytes of the same data when compressed. Therefore, a compression ratio of 2.0:1, or simply 2.0, means that the data compressed to one-half its original size, and if maintained for that whole tape, such compression would effectively double the data capacity of the tape drive.

Operating System Test Results

Figure 2 shows the measurements of compression ratio on the OpenVMS and UNIX systems. The difference between the compression ratios of the DLZ1 prototype and those of the IDRC prototype is striking on the graph. The DLZ1 prototype had significantly higher compression ratios for all the data types tested. Note that these results, as compared to the results of the SDS-3 testing, are more representative of the real world, since most of these data sets came from live multimegabyte databases.

[Figure 2 (Operating System Data Compression Ratios) is not available in ASCII format.]

We tested the ULTRIX dump utility on the same system and data on which we ran the tar utility. The dump utility compression ratios were almost identical to those obtained with the tar utility. This result was not surprising since the bulk of the data stored was identical -- only the metadata created by the utility varied. For comparison purposes, the average compression ratio for these data types was 2.76 for the DLZ1 prototype and 1.54 for the IDRC

prototype.

Although compression measurements were the focus of the operating system--based tests, for general information, we also took some throughput measurements. The DECsystem 5500 system running the dd utility achieved write rates of approximately 0.85M bytes/s for the data types. Running the tapex utility's performance test (which is not disk or file system limited) on a similar machine resulted in rates of more than 3M bytes/s. The 3M-byte/s rate implies that, when running dd or tar, the disk and/or file system is the likely bottleneck, since the ULTRIX drivers, SCSI channel, and tape driver were capable of three times the throughput. (Other possibilities are inefficiencies within dd and/or tar, inefficient handling of two devices on the SCSI bus, insufficient CPU horsepower, etc.)

OpenVMS tests showed similar results for the BACKUP utility, but the throughput is likely to have been limited by the KZQSA adapter. Other tests indicate that the KZQSA has a limit of 0.8M bytes/s to 0.9M bytes/s with the OpenVMS system.

The informal operating system throughput testing confirms that the particular configurations tested are not suitable for measuring the bandwidth limits of the DLT2000 tape drive, when using the standard backup utilities. Note that the newer VAX and the Alpha AXP platforms have much higher throughput capabilities and are able to more fully utilize the capabilities of the DLT2000 product. These platforms were not available when we performed this study.

SDS-3 Test Results

The SDS-3 tests measured compression ratios and data throughput rates.

Compression.

Figure 3 shows the SDS-3 data compression ratios. The ratios for the first four data types are in the normal range, i.e., the DLZ1 prototype averaged approximately 2.4 and the IDRC prototype averaged approximately 1.5. For the Paintbrush bitmap file, both prototype versions compressed at about the same efficiency.

[Figure 3 (SDS-3 Data Compression Ratios) is not available in ASCII format.]

Although the 30:1 compression ratio for the Ones pattern data is not representative of normal data, the ratio gives a sense of the maximum efficiency of the algorithms. The Repeat pattern test ratios highlight the ability of the DLZ1 algorithm to capitalize on redundant strings of moderate length (24 bytes, in this case). The IDRC algorithm lacks this ability. None of the many data sets tested compressed better with the IDRC algorithm than with the

DLZ1 algorithm. (We tested six other data sets but did not include the test results in this paper because they showed little variation from those presented.)

Throughput Rates. Figure 4 shows the data throughput rates for six of the data types; compression ratios are annotated at the bottom for convenience. The use of a line graph rather than a bar graph suggests some correlation between compression ratio and throughput. We tested variants of these data types to explore the strength of this correlation.

[Figure 4 (SDS-3 Data Throughput Rates) is not available in ASCII format.]

With the DLZ1 algorithm, we found data sets that had the same compression ratio but significantly different throughput rates. We saw variations of up to +/-0.3M bytes/s from the "expected" rate, which is the native drive rate (1.25M bytes/s) multiplied by the compression ratio.

The throughput rate with the IDRC algorithm tends to correlate more strongly with the compression ratio, but we did see variations. For example, the VAXcamera data at a compression ratio of 1.4 transfers about 0.1M bytes/s faster than Harvard Graphics data, which compresses at 1.6.

Even more striking is the difference on write and read transfer rates. The DLZ1 algorithm is almost always significantly faster on decompression. This feature is characteristic of this type of LZ algorithm. On the other hand, IDRC write and read rates match very closely, typically within 0.05M bytes/s.

The throughput limit of the SDS-3 system used was high enough to not usually be a factor. Knowing this fact was essential for the proper interpretation of test results. A bottleneck in the tape device must be distinguishable from an adapter or tester limitation. We measured the throughput limit of the SDS-3 system by writing and reading the Ones pattern and similar data patterns, which are highly compressible by the IDRC algorithm. With a 64K-byte block size, throughput on the SDS-3 system peaked at about 3.5M bytes/s. When we increased the block size 1M byte, the throughput jumped to nearly 4.5M bytes/s. This increase was due to reduction in the amount of command overhead for a given amount of data being transferred on the SCSI bus. None of the normal data types tested, except the Paintbrush bitmap files, could approach compression ratios high enough to begin to push the limits of the SDS-3 system.

These results indicate that at higher data rates, the SDS-3 system becomes a limiting factor. Analysis of SCSI protocol handling on the SDS-3 system shows that the nondata portions of a transaction (e.g., message, command, and status) are handled somewhat inefficiently. At high throughput rates, this overhead

is significant enough to affect throughput to the device. Using a larger block size reduces this per-command overhead for a given amount of tape data and allows a higher throughput to be achieved on the SCSI bus.

BENCHMARK TEST RESULTS

We wrote the Harvard Graphics data set repeatedly to the TA91 tape drive. With compression disabled, about 132M bytes fit on the media. With compression enabled, 216M bytes were written, giving a compression ratio of 1.64. This ratio compares closely with the 1.66 obtained on the IDRC DLT2000 prototype.

We then used the SDS-3 tool to repeatedly write the first 64K bytes of the U.S. Constitution to the EXB-8505 tape drive. With compression disabled, about 5G bytes were written. With compression enabled, 7.6G bytes were written, giving a compression ratio of 1.52. Again, this corresponds closely with the compression ratio of 1.54 achieved when writing text data on the IDRC DLT2000 prototype.

We performed more testing for general comparison between the DLZ1 DLT2000 product and the EXB-8505 product. The U.S. Constitution data compressed at 2.23 on the DLT2000 drive and at 1.52 on the EXB-8505 drive. Figure 5 shows the results of throughput testing with this data on these two products, using two block sizes, 10K-byte blocks and 64K-byte blocks.

[Figure 5 (EXB-8505 and DLT2000 Data Throughput Rates) is not available in ASCII format.]

CONCLUSIONS

The compression efficiency testing outlined in this paper indicates that, for most data sets, the DLZ1 algorithm usually achieves a higher compression ratio than the IDRC algorithm and, therefore, yields a consistent capacity advantage over the IDRC algorithm. The reader should carefully note that regardless of the algorithm used, the actual capacity increase that a user might realize with data compression depends heavily on the specific mix of data. The following summarizes the compression results presented in this paper. Based on the compression testing in the operating system environment, a DLT2000 product using DLZ1 compression has a typical capacity of 25G bytes to 30G bytes. A DLT2000 product using IDRC compression would typically hold about 15G bytes of data.

The data throughput testing showed that, in most cases, the DLZ1 DLT2000 prototype transferred data at a faster rate than the IDRC DLT2000 prototype -- even though the IDRC prototype's hardware implementation was capable of almost twice the data rate (5M bytes/s for the IDRC drive and 2.5M/3.0M bytes/s for the DLZ1

drive). The IDRC implementation did not perform better for two reasons.

1. Given the same data set, the compression ratio of the IDRC implementation is almost always less than that of the DLZ1 implementation.

2. The typical compression ratio of the IDRC implementation is somewhat low, in an absolute sense (less than 1.8).

Since data compression in the tape device has a multiplying effect on data transfer rates seen by the host, a low compression ratio limits the practical rate at which compressed data can be made available to the tape media.

To transfer data faster than the DLZ1 prototype, the IDRC prototype must achieve a compression ratio that multiplies the drive's native data rate beyond the throughput limit of the DLZ1 prototype. This limit is about 2.5M bytes/s for write operations. Calculating the approximate minimum compression ratio (Cr) needed is straightforward, as the following steps show:

```
Cr x (native data transfer rate) = throughput limit

Cr x 1.25M bytes/s = 2.5M bytes/s

Cr = (2.5M bytes/s)/(1.25M bytes/s)

Cr = 2.0 (or 2.0:1)
```

Thus, when the IDRC prototype compresses data at a rate greater than 2.0:1, its transfer rate should exceed that of the DLZ1 prototype. Indeed, with the Paintbrush and Ones data patterns, the compression ratio was more than 4.0:1, and the transfer rate measurements show the throughput potential of the IDRC implementation over the DLZ1 implementation. These data patterns are not typical, however, and more realistic data sets (e.g., binary, source files, text, and databases) show the IDRC algorithm compression ratios to be only in the 1.5 to 1.7 range. The benchmark testing confirms these results and, therefore, the correctness of the IDRC DLT2000 implementation. These low IDRC compression ratios for typical data are what prevent the IDRC implementation from achieving its throughput potential on the DLT2000 tape product.

The DLZ1 DLT2000 implementation was adopted for the actual DLT2000 tape product. As the development team completed the design, they made hardware and firmware improvements to enhance the data throughput characteristics of the final product. For example, they increased the clock rate on the compression chip by 10 percent and optimized critical firmware code paths.

ACKNOWLEDGMENTS

Other members of the firmware engineering team made contributions relevant to this paper. In particular, I would like to thank Brian LeBlanc for conducting performance SDS-3 test runs that confirmed my results and in some cases were incorporated into the data presented. I would also like to thank Haim Bitner for assisting me in digging into the theory behind the LZ and IDRC compression algorithms and for running the EXB-8505 benchmark tests.

REFERENCES

1. D. Whiting et al., Data Compression Apparatus and Method, U.S. Patent 5,016,009 (May 14, 1991).

2. "9705 Data Compression Coprocessor Data Sheet," Revision 1.00, STAC Electronics (December 1991).

3. T. Welch, "A Technique for High-Performance Data Compression," Computer 17 (June 1984): 8-19.

4. V. Chinnaswamy, "An Overview of Compression Techniques and TA90 Performance with Compression," internal report (Maynard, MA: Digital Equipment Corporation, July 1991). This internal document is unavailable to external readers.

5. D. Lelewer, Current Techniques in Data Compression (Irvine, CA: University of California, Instructional Television Network, 1993).

6. Compaction Algorithm, Binary Arithmetic Coding, 1st Draft, Proposed American National Standard X3B5 (November 17, 1989).

7. T. Bell, J. Cleary, and I. Witten, Text Compression (Englewood Cliffs, NJ: Prentice Hall, 1990).

8. J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," IEEE Transactions on Information Theory, vol. IT-23, no. 3 (May 1977): 337-343.

9. EXB-8505 8mm Cartridge Tape Subsystem User's Manual, Revision 002 (Boulder, CO: EXABYTE Corporation, November 1992).

TRADEMARKS

The following are trademarks of Digital Equipment Corporation: Alpha AXP, DECsystem, Digital, HSC, MicroVAX, OpenVMS, TA, ULTRIX, VAX, and VAXcamera.

EXABYTE is a registered trademark of EXABYTE Corporation.

Harvard Graphics is a trademark of Software Publishing Corporation.

Hewlett-Packard is a registered trademark of Hewlett-Packard Company.

IBM is a registered trademark of International Business Machines Corporation.

MS-DOS is a registered trademark of Microsoft Corporation.

Paintbrush is a registered trademark of Zsoft Corporation.

PostScript is a registered trademark of Adobe Systems Incorporated.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company, Ltd.

BIOGRAPHY

David C. Cressman A consulting software engineer in the Tapes and Solid State Disk Engineering Group, Dave Cressman is currently working on the development of digital linear tape (DLT) products. He developed the SCSI firmware for the TZ85 and TZ86 tape products and was responsible for the TMSCP firmware of the TF85 and TF86 tape products. Dave joined Digital in 1988 after seven years with Data General Corporation, where he developed a SCSI subsystem controller and operating system device drivers. He received B.S.C.S. and B.S.E.E. degrees (1981) from State University of New York (SUNY) at Stony Brook.