An Oracle White Paper
April 2014

How to Use Tape Tiering Accelerator (Automatically Linked Partition)

Introduction

Tapes hold hundreds to thousands of gigabytes of data. Typically, the data is “stacked” on the media as different data sets, and each data set has a different expiration. When the expiration occurs, there is wasted space on the tape. Over time the wasted space becomes so large that customers must reclaim the tape, which can consume many hours. In order to save customers time and money, a method to capture the wasted space was created: Tape Tiering Accelerator, a feature of Oracle's StorageTek T10000C and T10000D tape drives. The tape drive creates a hard disk-like format so that more of the tape can be used. Tape partitioning (nonlinked) has existed for many years; however, many current tape drives don't implement them because developers didn't think it was useful. This implementation goes beyond basic partitions and creates automatically linked partitions so that the data can span more than one Tape Tiering Accelerator and be noncontiguous on tape. Furthermore, the tape drives handle much of the formatting without host intervention.

It is important to identify some terminology. Tape Tiering Accelerator identifies the feature. Partitions and ALPs (Automatic Linked Partitions) mean the same thing for this document. They reference the units on the tape that make up the Tape Tiering Accelerator tape.

The StorageTek T10000C and T10000D tape drive Tape Tiering Accelerator specifications:

- Cartridge requirement: Long T2
- Number of Partitions for T10000C: 480
- Number of Partitions for T10000D: 600
- Partition size for T10000C: ~9GB

- Partition size for T10000D: ~11GB

This paper is written for storage administrators who want to use the host commands that apply to Tape Tiering Accelerator and review examples that demonstrate how to save space and improve access time. When the term *automatically linked partition* (ALP) is mentioned in the paper, it is synonymous with Tape Tiering Accelerator.

New Host Interface Commands

Following is a summary of the host commands added for Tape Tiering Accelerator or ALP mode. More specific information is provided in the latest version of the ALP interface API document. Throughout this paper there are references to the ALP API call.

Locate ALP

ALP_LocateALP()

The locate ALP command moves the tape into position on the target tape drive so that the application can write or read to a specific ALP on tape. The tape drive positions the tape to the nearside of the first block of the ALP.

Set ALP Mode

ALP_EnableDisableALPMode ()

The activate ALP function allows converting a standard tape to an ALP tape. Once converted to an ALP tape, the tape format cannot be reverted to normal mode. All down-level tape volumes prior to ALP feature implementation are “standard mode” by default. The process of converting to an ALP tape is considered destructive; meaning any data currently on the tape is effectively erased.

Set Writable ALPs

ALP_SetWritableALPs()

This command tells the target tape drive which ALPs are available for write operations in the current logical volume. This command must be issued at beginning of tape (BOT) and has no effect on read operations. The command uses a bit mask is to identify which ALPs can be written with each bit representing an ALP. A bit value of 1 means the ALP is writable. If 480 ALPs are available, then 60 bytes are used to represent the mask. Bit 7 of the least significant byte represents ALP 0 and bit 0 (zero) of the most significant byte represents ALP 479. If any bits are set beyond byte 60 during the writing of a mask, the command will be rejected by the drive.

A write mask is not permanently saved on the tape and must be rewritten after an IPL of the drive.

Report ALP Mode

ALP_GetALPCharacteristics ()

This command reports whether the currently mounted tape is an ALP tape or not, and whether the drive is capable of creating or reading ALP tapes.

Report Current ALP

ALP_RetrieveALPIndex()

This command reports the position of the write/read head by ALP number, much like getting the current host block ID with the SCSI read position command.

Report ALP Linkage

ALP_GetALPLinks()

The report ALP linkage command reports the current ALP mode tape linkage (mapping). The API call returns 1,024 bytes of information. The information represents how the ALPs are forward linked together. The information represents each of the ALPs (2 bytes of data for each ALP). The 2 bytes of data are the next ALP number in the link or one of four special values. The four special values are 0xFFFF (ALP not linked), 0xFFFE (ALP not used), 0xFFFD (ALP link unknown), and 0xFFFC (ALP blank). "ALP not linked" means there are currently no more ALPs linked to this ALP. "ALP not used" means the ALP isn't used in the current tape format. While 480 ALPs are defined, the current format may not have that many. "ALP unknown link" means the drive doesn't know if the ALP is linked to something else. This only happens when a tape is loaded and the drive is power cycled or receives a SNO. "ALP blank" means the ALP hasn't been written since the last time the tape was converted to an ALP tape.

ALPs erased following a DSE erase function will revert to a not linked state (0xFFFF). DSE erase will only erase ALPS set as writable via the current write mask.

Start New Logical Volume

ALP_StartNewLogicalVolume()

This command causes the drive to start a new logical volume. This command is intended to be used when positioned at the start of an ALP where the application would like to start writing from logical block 0 for a new partition. The command has the effect of breaking the link to previous and next ALPs.

Set ALP Locks

ALP_SetALPLocks ()

This command allows users to protect their data on a physical partition and ALP granularity by locking down the desired ALP(s) to a read-only mode. A lock bit of logical 1 locks the respective ALP to read only, thus preventing the ability of writing to the ALP even if the mask bit for that bit is enabled. Respectively, setting the lock bits to a value of 0 enables the ALP for write capability if the write mask bit for that ALP is set to a logical 1. Before a set locks command can be issued, it is required that an ALP tape must be in ALP mode, initialized by starting a volume on ALP 0, and by setting the mask bit for ALP 0 to a logical 1 and by writing a small amount of data. As in the write mask, the lock mask contains 60 bytes of data for a 480 ALP format. The lock byte logical values are the inverse of the write mask bits values. In other words, in order to lock an ALP (prevent writing), a bit of one is set for each ALP for which it is desired to prevent writing.

The current lock mask is persistent and saved on the tape cartridge after being issued. A lock mask command can be written only after a tape is initialized by issuing a start volume and writing a small amount of data to ALP 0. Locks do not affect writable ALPS set via the current write mask. Locks only affect the ability to write or change a new mask.

Get ALP Locks

ALP_GetALPLocks ()

This command returns the ALP lock mask that was previously written to the tape. This information is stored on the tape, and it can be used to generate a write mask based on last written lock values if it is desirable to set a mask based on last known locks. Because the lock values are the logical inverse of the write mask values, it is possible to read the previously written lock values, invert the bits, and write a mask using the inverted bits. This may be useful after an IPL of the drive if it is desirable to set a write mask based on lock values that were set prior to an IPL of the drive.

ALP Examples

These examples demonstrate typical uses for ALP to save space and improve access time.

Save Space

To format an ALP tape, the application issues a set ALP mode command. The tape should be empty or a scratch tape since initializing it may prevent access to data previously written on the tape. Formatting an ALP tape divides the tape into a fixed number of empty ALPs with a guard band allocated between each partition. Next, the application sets a write mask by issuing a set writeable ALPs command; this command tells the tape drive which ALPs to link together during subsequent write operations. In figure 1 the first 20 ALPs on tape have been mapped for writing. Each block is an empty ALP that can be used by the application.

Files	**Empty – No data has been written to the tape**																			
ALPs	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**

Figure 1. An empty ALP tape

The application now sends a locate ALP command to ALP #1 and a start new logical volume command. The application now can begin writing to this logical volume on tape using standard tape write commands. This process may occur over multiple mount and dismount cycles and across several drives.

In figure 2 below the application writes nine files. Those files span all 20 ALPs. The data shown in the top row are application files, and the bottom row indicates the physical relationship of the 20 ALPs to those files. The files can span one or more ALPs or be contained within a single ALP.

Files	***A***			***B***		***C***		***D***	***E***	***F***			***G***		***H***				***I***	
ALPs	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**

Figure 2. Host files and ALP allocation

Over time the files that are written to tape expire and become obsolete or invalid. As the expired files become obsolete, the tape begins to resemble "Swiss cheese," with holes created throughout the tape. In figure 3, host files B and F expire and become obsolete, which frees ALPs 2, 3, 4, 9, and 10 for space reclamation.

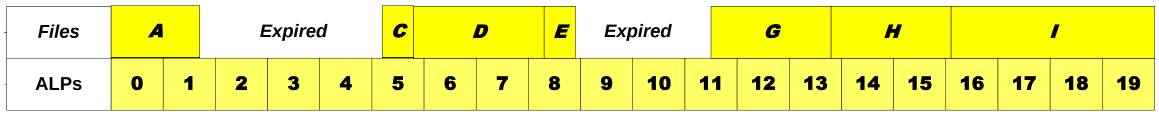


Figure 3. Host files becoming invalid

The application now can reuse the ALPs containing the expired files by starting a second logical volume and over writing them. Initially the application should send a new set writeable ALPs command that maps partitions 2, 3, 4, 9, and 10 as writeable. Then the application positions to the first ALP that was freed when record B expired. In this case, it is ALP 2. The locate ALP command is used to locate to partition 3. Finally, the application sends a start new logical volume command to identify this new logical volume. The application now can write files M and N. As these files are written to tape the drive automatically positions to the next available ALP. This process may occur over multiple mount and dismount cycles and across several drives. Although read access is still permitted to all ALPs, it is only possible to write to partitions that form the current writable ALPs (2, 3, 4, 9, and 10) shown as green blocks below. File N links from partition 4 to partition 9 and then continues into partition 11. The space between file A and M (end of partition 1), file N and C (beginning of partition 5), and file E and N (end of partition 8) are wasted space.

This space is not empty; it contains the residual data from the expired files B and F. If the application attempts to read the old file B data, the beginning of that file will be read. Then the drive will report an EOD error at the end of ALP 1. As a result, the application will stop reading at the end of file A. This process is discussed in detail below.

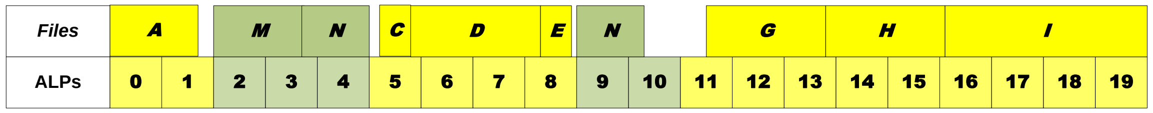


Figure 4. New files written to freed partitions

The application has to retain a record of the start of each record (host block ID and ALP number) to navigate through logical volumes where files have expired and partitions are reallocated to a new logical volume. Users can follow these steps to read the first logical volume (yellow) in figure 5:

1. Locate to ALP #0

2. Locate to the start block ID for record A

3. Read record A

4. Locate to ALP#5

5. Locate to the start block ID for record C

6. Read record C, D, and E

7. Locate to ALP#11

8. Locate to the start block ID for record G

9. Read record G, H, and I

Users should notice that it is necessary for the application to track which set writeable ALP command map corresponds to each logical volume. There is a different write map for the first (yellow) and second (green) logical volumes.

Now the assumption is that files D and H shown in figure 5 expire. Following the same process as described above, the application can start a third logical volume (purple). Again, users send a new set writeable ALPs command that maps partitions 6, 7, 14, and 15; a locate ALP command to ALP #6; and a start new logical volume command. As shown below, the host then can write this third logical volume with files S and T.

Files	***A***		***M***		***N***	***C***	***S***	***T***	***E***	***N***		***G***			***T***		***I***			
ALPs	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**

Figure 5. New files written for third logical volume

To read the second logical volume (green) in figure 6:

1. Locate to ALP#2

2. Read files M and N

To read the third logical volume (purple) in figure 6:

1. Locate to ALP#6

2. Read files S and T

It is worth noting that the application only needs to track ALP# and host block ID when space for expired files is reclaimed and used by another logical volume. In logical volumes that have not had space reclaimed, like the green and purple examples in figure 6, the ALP feature of the tape drive will move from ALP to ALP as if the logical volume was written sequentially in a single tape partition.

Improve Access

This example describes features that are only supported in the StorageTek T10000C tape drive. The following representation of an ALP formatted tape has five sections with 5 ALPs in each section, for a total of 25 ALPs. This format does not actually exist, and it is described for illustration purposes only. The actual StorageTek T10000C tape drive ALP format has 480 ALPs and is too large to show in a small drawing. In this illustration, ALP 0 is located at the beginning of tape (BOT) at the bottom left side, and the ALPs serpentine through the tape in a linear fashion from BOT to end of tape (EOT) and back again.

	Section 0	Section 1	Section 2	Section 3	Section 4	
	← ALP 39	← ALP 38	← ALP 37	← ALP 36	← ALP 35	
	ALP 30 →	ALP 31 →	ALP 32 →	ALP 33 →	ALP 34 →	
	← ALP 29	← ALP 28	← ALP 27	← ALP 26	← ALP 25	
BOT	ALP 20 →	ALP 21 →	ALP 22 →	ALP 23 →	ALP 24 →	EOT
	← ALP 19	← ALP 18	← ALP 17	← ALP 16	← ALP 15	
	ALP 10 →	ALP 11 →	ALP 12 →	ALP 13 →	ALP 14 →	
	← ALP 9	← ALP 8	← ALP 7	← ALP 6	← ALP 5	
	ALP 0 →	ALP 1 →	ALP 2 →	ALP 3 →	ALP 4 →	

Figure 6. Representation of an ALP-formatted tape.

Section 0 in figure 7 contains ALPs 0, 9, 10, 19, 20, 29, 30, and 39. Since all of these ALPs are in a single section the average access time to locate to any ALP in section 0 is relatively short. A StorageTek T10000C tape drive has an average access time for any ALP in the same section of about 10 seconds.

If the application requires fast tape access, this can be achieved by constructing logical volumes within the same section. To build a logical volume in section 0, the following commands should be sent to the drive.

1. Load a StorageTek T10000C ALP-formatted tape

2. At BOT issue a set writeable ALPs command that maps ALPs 0, 9, 10, 19, 20, 29, 30, and 39

3. Issue a start new logical wolume command

4. Begin writing this logical volume

As files are written to this logical volume the drive will automatically link the mapped ALPs starting with the lowest numbered ALP. If application files are written to the first three ALPs in the logical volume to the ALPs shaded in yellow will contain the logical volume as shown below. A wrap turn will be performed at the end of ALP 0, and ALP 9 will be written in the opposite direction. At the end of ALP 9, another warp turn will be performed and direction reverses again.

	Section 0	Section 1	Section 2	Section 3	Section 4	
	← ALP 39	← ALP 38	← ALP 37	← ALP 36	← ALP 35	
	ALP 30 →	ALP 31 →	ALP 32 →	ALP 33 →	ALP 34 →	
	← ALP 29	← ALP 28	← ALP 27	← ALP 26	← ALP 25	
BOT	ALP 20 →	ALP 21 →	ALP 22 →	ALP 23 →	ALP 24 →	EOT
	← ALP 19	← ALP 18	← ALP 17	← ALP 16	← ALP 15	
	ALP 10 →	ALP 11 →	ALP 12 →	ALP 13 →	ALP 14 →	
	← ALP 9	← ALP 8	← ALP 7	← ALP 6	← ALP 5	
	ALP 0 →	ALP 1 →	ALP 2 →	ALP 3 →	ALP 4 →	

Figure 7. ALPs 0, 9, and 10 are written in section 0

If users desire to set a write mask for an entire section, the mask table below can be referenced for the binary values. For any section, the mask values are a repeating binary sequence indicated by the values show in the table below. For example: To mask the entirety of section 0 as writable, a writable mask containing 60 bytes with the hexadecimal values of 80,60,18,06,01,80,60,18,06,01……should be written to the drive. Lock values must be set which allow writing the specific mask values.

Section mask table for full sections - repeating binary sequences (HEX)										
	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10-59
Section 0	0x80	0x60	0x18	0x06	0x01	0x80	0x60	0x18	0x06	0x01	…..
Section 1	0x40	0x90	0x24	0x09	0x02	0x40	0x90	0x24	0x09	0x02	…..
Section 2	0x21	0x08	0x42	0x10	0x84	0x21	0x08	0x42	0x10	0x84	…..
Section 3	0x12	0x04	0x81	0x20	0x48	0x12	0x04	0x81	0x20	0x48	…..
Section 4	0x0c	0x03	0x00	0xc0	0x30	0x0c	0x03	0x00	0xc0	0x30	…..

Drive Operation

The following picture is taken from the ALP example, and it is useful for a discussion of drive operations. It also may be helpful to refer to the ALP example to see how the tape got into this condition.

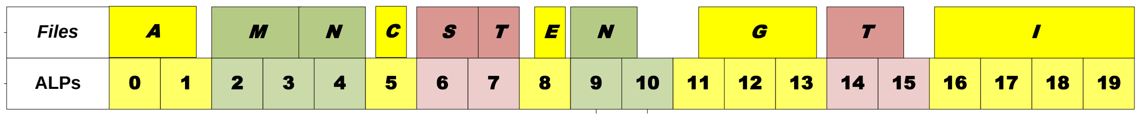


Figure 8. Three different logical volumes with ALP

Logical Volume

A logical volume is defined as a group of ALPs linked together that have both a block 0 and a physical EOD. In the above picture there are two complete logical volumes. The first logical volume maps ALPs 2, 3, 4, 9, and 10 (files M and N), and the second logical volume maps ALPs 7, 8, 15, 16 (files S and T).

Partial Logical Volume

A partial logical volume is a group of ALPs linked together with either no block 0 or no physical EOD, or neither a block 0 or an EOD. In the above picture there are five partial logical volumes. ALP 0 and 1 are a partial logical volume with a block 0 but no physical EOD. ALPs 16, 17, 18, 19 make up a partial volume with no block 0 but which does contain an EOD. ALP 5, ALP 8, and ALPs 11, 12, 13 are partial volumes without a block 0 or a physical EOD. These partial logical volumes contain the files that were not expired for the originally written logical volume shown in figure 2. When combined they still represent that logical volume.

Writing

When writing, the drive only allows writes on ALPs identified with the set writable ALPs command. On load the tape defaults to no ALPs being writable. The host should give the set writable ALPs command mask after load and must be done at BOT. If the set writable ALPs command isn't issued, the tape will only operate in a read-only mode.

The drive always chooses the lowest ALP after the current ALP when linking ALPs. The drive does not allow the next ALP to wrap around the writable list. For example, ALP 255 will not be forward linked to ALP 20.

The drive will report LEOV on the last free ALP in a manner similar to a normal tape. The amount of space between LEOV and PEOV has not changed. When starting a write, the drive uses the current block as the append point, just like in normal tape operation, with one exception: if the write command was proceeded by a start new logical volume, the drive writes at block 0 at the start of the current ALP. The host should be located at the start of the ALP before issuing the start new logical volume command. Regardless of where the host is logically positioned, the write will take place at the start of the ALP.

When the application is finished writing a file, the application should issue a read position command with the ALP bit set in order to get the ALP number where the last block was written.

Reading

When reading the tape, the drive automatically moves to ALPs linked together. For example, if the host is positioned to read ALP 11, the drive transitions from ALP 4 to 5, 10, and 11 with no host intervention.

It is assumed that the host does not read old host files as a matter of course. But if the host requests to read the start of host file B in ALP 2, for example, the data is returned until the end of ALP 2 at which point the drive returns EOD.

Spacing and Locating

The drive operates on (partial) logical volumes when processing space block/file and locates commands. When spacing/locating prior to first block in a partial volume, the drive reports back that BOT has been crashed. For example, if record G in ALP 11 starts at block 10,000, then any operation that ends up before 10,000 reports a crash BOT.

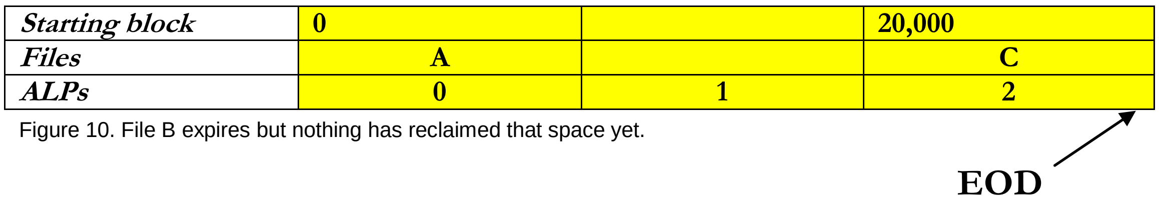
Similarly when spacing/locating beyond the last block in a partial volume EOD is reported. For example, if record E in ALP 8 ends at block 1,500, then any operation that ends beyond 1,500 reports a crash EOD.

Figure 2 and figure 5 serve as examples. In figure 2, file A starts in ALP 0 and for this example, the starting block ID for ALP 0 is zero. File B spans ALP 2, and for this example, the starting block ID in ALP 2 is 20,000. Now, file B expires. In figure 5 the user decided to write file M starting in ALP 2. The result of putting file M into ALP 2 caused an EOD to be put down at the end file A.

Starting block	**0**	**10,000**	**20,000**
Files	**A**	**B**	**C**
ALPs	**0**	**1**	**2**

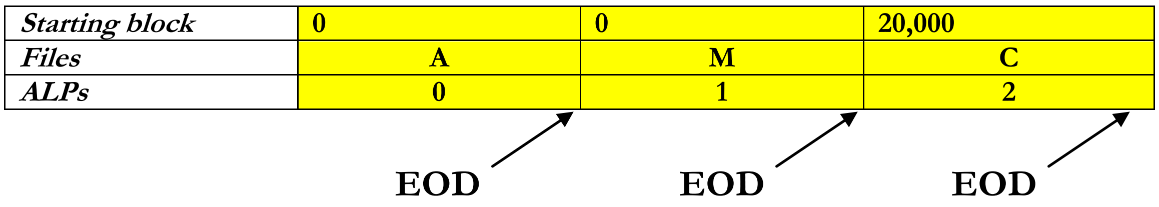
Figure 9. Three files written into one logical volume

EOD



Starting block	**0**		**20,000**
Files	**A**		**C**
ALPs	**0**	**1**	**2**

Figure 10. File B expires but nothing has reclaimed that space yet.



Starting block	**0**	**0**	**20,000**
Files	**A**	**M**	**C**
ALPs	**0**	**1**	**2**

Figure 11. File M reclaims the space that file B used to occupy.

Normal Spacing and Locating

In figure 10, the user can locate to any blocks in the range of 0 thru 29,999. It is no problem because the ALPs are linked. If the user locates to 10,000, then the tape will end up in ALP 1.

Crash into EOD Example

Figure 10 shows files A, B, and C. ALPs 0, 1, 2 are all linked together into one logical volume. The EOD is at the end of ALP 2. In figure 11, file B expires at the host. In figure 12, the host reclaims ALP 1. When that reclaim occurs, the links are broken and each ALP has its own EOD. If the user now tries to locate to block 10,000, the tape drive will generate a crash into EOD because ALP 0 is no longer forward linked to ALP 1.

Crash into BOT Example

Figure 10 shows Files A, B, and C. ALPs 0, 1, 2 are all linked together into one logical volume. The EOD is at the end of ALP 2. In figure 11, file B expires at the host. In figure 12, the host reclaims ALP 1. When that reclaim occurs, the links are broken and each ALP has its own EOD. If the user locates to ALP 2 with the locate ALP command, the tape drive will be positioned in ALP 2. If the user now tries to locate to block 10,000, then the tape drive will generate a crash into BOT because ALP 1 is no longer backward linked to ALP 2.

ALP Layout

A StorageTek T10000C tape drive tape formatted for ALP is represented in the StorageTek T10000C format specification. It has five sections of 96 ALPs for a total of 480 ALPs. The ALPs are organized in a linear serpentine pattern starting at ALP #1 at the beginning of tape (BOT) at the bottom left side, increasing sequentially down the length of tape to end of tape (EOT) on the right side, and then returning to BOT.

ALP Capacity

For a StorageTek T10000C tape drive tape, each ALP is approximately 9 GB. (480 * 9 GB is not equal to the full 5,000 GB that can fit on a tape.) This loss of full capacity is needed to allow the ALPs to be written in any order.

Potential ALP Benefits and Effects

Increased Access Times

If an application identifies data for frequent access, that data can be located in ALPs at the beginning of the tape for fast access. The host can control where data is located on tape by setting the writable ALPs correctly.

Like data sets can located together, so that access to multiple files of like data can be performed more efficiently.

Efficient Use of Tape Capacity

Depending on the application, much of the data on a particular tape is old, not needed data. This data now can be reclaimed without having to rewrite the valid data on the tape. With tape capacities ever increasing, the effective tape capacity will continue to worsen.

Theory of Operation

While there are many valid ways to use the set of commands to perform an individual application's requirements, this section shows how Oracle envisions the commands will be used.

Initializing an ALP Tape

To initialize an ALP tape, the only thing required is to issue an activate ALP command on a tape that is in the free pool (users should remember that the activate process is destructive). According to best practices, users should set the writable ALP mask for byte 0, bit 7 to a logical 0x1 and write some sort of tape identifier or tape mark at ALP 0. ALP 0 must be initialized before volumes may be created or accessed that utilize remaining ALPs or before a lock mask can be written.

Loading an ALP Tape

After loading an ALP tape, a set writable mask command should be issued so the tape can be written. The writable mask is kept by the host application based on which ALPs no longer hold valid data.

Positioning to a Host File

The host is responsible for keeping track of where each host file is on tape in the form of ALP and host block ID. Positioning to a host file is as easy as issuing a locate ALP to the proper ALP and then issuing a standard tape locate command to get to the start of the host file.

ALP tape read position byte level references are relative to a start of volume position for a particular partition. If, for example, a second or third partition is created at ALP 10 by issuing a start volume at that location 10, then once a file is written beginning at ALP 10, any subsequent read position information for that partition will be relative to the volume beginning at ALP 10. The same is true regardless of how many partitions are created. Issuing a start volume at a particular ALP does not immediately set the beginning LBA to LBA 0; a write to that volume must occur first.

Reading a Host File

After positioning to a host file, reading is as simple as issuing standard tape read commands. The drive will automatically transition to next ALPs as needed.

Appending a Host File

When appending a new host file to an existing one, the host first positions the drive to the end of the existing host file. Next, the host issues a report current ALP command and notes this value in the host's file location structure. Next, the host issues a standard read position command to get the host block ID; this also should be noted in the host's file location structure. These values are used later when the new host file is accessed or for determining when an ALP is free to be reused. Next, the host writes the new file. Another report current ALP command is issued to determine the current ALP. Based on the starting ALP and current ALP, the host can determine which ALPs the file resides in. This information is kept in the host file location structure and later used to determine when an ALP is free to reuse.

Writing a Host File to a Free ALP

When starting a new logical volume, the first command to issue is a locate ALP to the first free ALP. Next the host file location structure is updated with the first free ALP and a host block ID of 0. Next, the host sends a start new logical volume command and writes the new file. A report current ALP command is issued to determine the current ALP. Based on the starting ALP and current ALP, the host can determine which ALPs the file resides in. This information is kept in the host file location structure and later used to determine when an ALP is free to reuse.

Conclusion

The Tape Tiering Accelerator feature of Oracle's StorageTek T10000C/T10000D tape drives helps users capture wasted space that results from data set expiration. By using Tape Tiering Accelerator, users can save time, reduce labor cost, increase access times, and enable more efficient use of tape capacity. The feature automatically links partitions so the data can span tiers or partitions and be noncontiguous on tape. Additionally, the tape drive handles much of the formatting without host intervention.

ORACLE®

Oracle is committed to developing practices and products that help protect the environment

How to Use Tape Tiering Accelerator
(Automatically Linked Partition)
April 2014

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com

Hardware and Software, Engineered to Work Together