The IBM Virtual Tape Server: Making tape controllers more autonomic

G. T. Kishi

The IBM Virtual Tape Server provides a revolutionary tape storage solution that emulates tape drives using a disk cache, providing up to 256 virtual tape-drive images and stacking the virtual tape volumes onto a maximum of 12 physical tape drives. The high ratio of virtual to physical drives, combined with the stacking of virtual volumes, provides tape storage that is both cost- and space-efficient. Existing IBM products are used as microcode building blocks to provide the core functions of the virtual tape server. These core functions are integrated into a reliable, high-performance storage subsystem by autonomic computing controls.

Introduction

The year 2002 marked IBM's 50th year in the computer tape business, a half-century that began with the introduction of the IBM 726 magnetic tape drive [1]. Initially, tape drives were directly connected to their host computers. In 1984, IBM developed the first buffering tape-control units, which attached between host computers and tape drives. These control units provided improved performance, protocol conversion, and improved error handling. As both control-unit and tape-drive performance improved, it became apparent that many applications were unable to make efficient use of the bandwidth provided by the tape drives.

At that time, IBM analyzed the way in which mainframe customers were using tape drives. Although customers were able to use the full tape-cartridge capacity in some applications, most applications were using smaller volume sizes that left a large percentage of physical tape-cartridge capacity empty. It was determined that IBM customers had an uncompressed average tape-volume size of only 250 MB at a time when tape capacity was approaching 10 GB per cartridge. In addition, many of the applications customers were running could make use of less than 1 MB/s of the tape-drive bandwidth.

Only a small minority of customers were fully utilizing the capacity and throughput provided by IBM high-end tape drives. The majority of IBM customers were using tape applications in a manner that prevented them from taking full advantage of the throughput of IBM tape drives and the capacity of IBM tapes. In fact, according to an estimate by IBM and the research firm International Data Corporation (IDC), of the customers using IBM 3490 technology, only approximately "10% of each data center's tape cartridge is filled with data . . ." [2]. The advent of IBM 3590 technology with its increased tape-cartridge capacity further reduced average tape use to only 2–3% of the tape cartridge.

Many customers did not want to drastically alter their tape applications. They preferred to continue using these reliable, fully debugged applications for the long-term storage of their data. Tape is a physical medium that offered safe and reliable long-term storage at a lower cost of storage and a higher storage density per square foot of floor space than other media. A tape solution was required to support the large installed base of existing applications while providing more cost-efficient physical drive use and higher storage densities.

The virtual tape server

The IBM Virtual Tape Server (VTS) provided a revolutionary solution. The VTS emulates up to 256 customer-usable tape drives on a direct access storage device (DASD) cache, with each virtual tape emulated by a virtual tape daemon operating on a DASD file. It can

0018-8646/03/$5.00 © 2003 IBM

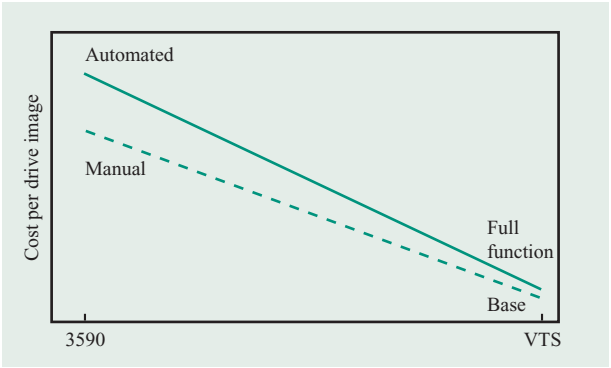

Figure 1

Comparison of cost per drive image for IBM 3590 tape drive and IBM Virtual Tape Server.

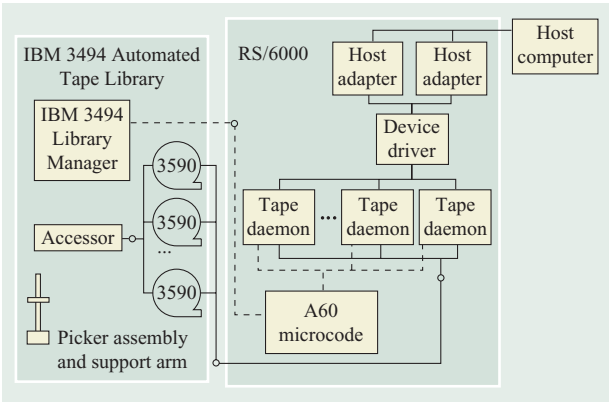

Figure 2

A60 tape controller microcode.

access up to 12 physical IBM 3590 tape drives to move the customer data onto and off physical 3590 tape cartridges.

The I/O throughput of the virtual tape device is buffered by the DASD cache. This buffering allows the consolidation of the virtual-device I/O onto the VTS back-end physical tape drives. In addition, the buffering of virtual volumes on DASD allows the VTS to *stack*, or concatenate, multiple virtual volumes on each physical tape cartridge using the back-end drives, thus increasing storage efficiency. This stacking enabled "users of cartridge tapes to better maximize the storage potential of each tape in their library" [3]. Using IBM 3590 Model B drives, the VTS improved system performance, increased storage capacity, and reduced the number of tapes required "by as much as a factor of 59 . . . " [4]. Customers switching to the newer IBM 3590 Model E drives, which write twice as much data on a tape as did the 3590 Model B drive, can achieve reduction factors of almost 120.

The virtualization provided by the IBM VTS isolates tape applications from the physical media used on the VTS. This allows migration to improved media without requiring changes to host applications.

Another advantage is the fact that VTS provides improved scratch-mount performance. A scratch mount is the mounting of a virtual volume with the intent of completely overwriting the volume. On a physical tape drive, a scratch mount requires 20 to 30 seconds to mount and load a physical tape cartridge. On a VTS, a scratch mount can be as quick as opening a new file in the DASD cache, reducing the overall mount time to only a few seconds.

In addition, the VTS helps provide improved access to recently written virtual tape volumes. Because the DASD cache on the VTS stores up to six terabytes of virtual volumes, these volumes are available for immediate access as soon as they are written to cache. On a real tape drive, mounting one of these cached files would require mounting a physical tape cartridge. In a VTS, a cached mount offers the same time savings as a VTS scratch mount, because both are accessed from cache.

The virtualization provided by the IBM VTS makes possible the mapping of more than one virtual drive to a physical drive on the VTS. On traditional tape-drive systems, the ratio of host-drive images to physical drives is 1:1. On a larger IBM VTS, the ratio of host virtual-drive images to VTS physical drives is in the range of about 20:1 to 40:1, depending on the number of physical drives.

The physical tape drive used in the current IBM VTS is the IBM 3590 tape drive. Many customers have made significant investments in enterprise-level tape drives. The high ratio of host virtual-drive images to VTS physical drives can provide customers with a significant cost savings in tape-drive purchases by reducing the number of physical drives required. A virtual drive costs approximately one-seventh as much as an IBM 3590 tape drive, as shown in **Figure 1**.

The reduced cost per virtual drive provided by the VTS creates new opportunities for automating tape processing. IDC found that unautomated, unmanaged tape storage systems cost 17 cents per megabyte in personnel costs. In contrast, the personnel cost for fully automated, fully managed sites with virtual tape systems is one cent per megabyte [5]. The advantages of tape virtualization were also recognized by major high-end tape vendors, who quickly followed IBM to market with their own virtual tape systems [6].

The IBM VTS was designed using a building-block design philosophy. This approach is premised on assembling existing hardware and microcode components

with a history of reliability into a new product by "gluing" them together with custom microcode. The major hardware components chosen for the VTS were the IBM RS/6000* series servers, the IBM 3494 Automated Tape Library, and the IBM 7133 SSA RAID Array. The major microcode components used in the IBM VTS were the IBM A60 control unit microcode, the AIX* operating system, and the IBM ADSTAR* Distributed Storage Manager (ADSM) AIX hierarchical storage management (HSM) client and server.

The IBM A60 control unit microcode (**Figure 2**) converts host adapter commands into IBM 3590 SCSI commands for up to ten tape-drive images. The A60 microcode has one microcode tape daemon per physical tape drive. Host data and commands are transferred to the tape daemon through a custom device driver using customized adapter microcode. The tape daemon transfers the host data and executes commands by executing the appropriate SCSI commands with the attached physical tape drive.

In the IBM VTS, this code was expanded to support up to 256 virtual tape-drive images, and a new component was written to convert host adapter commands into DASD file I/O operations. This component replaces the A60 SCSI interface to a physical tape drive with the appropriate file operations on the VTS DASD file.

These modifications provided the virtualization of host tape-drive images onto the VTS DASD cache. ADSM was used to manage the VTS cache data on and off the large physical tape inventory in the IBM 3494 Automated Tape Library, as shown in **Figure 3**. The building-block design and the reuse of the A60 microcode enabled the VTS to provide tape virtualization with minimal changes to the existing microcode (**Figure 4**).

Virtual tape server design

In the VTS design, when a virtual volume is written by the host, the tape daemon transfers it into a file on the VTS DASD cache. At this time, the only copy of the file is the one in the cache. This virtual volume is referred to as a *resident* virtual volume (**Figure 5**). At a later time, the VTS will copy the virtual volume onto IBM 3590 physical tapes using the ADSM HSM client and the ADSM server. The virtual volume will have one copy left in cache for potential cache hits, while another copy is on physical tape. A virtual volume in this state is referred to as *premigrated*. Eventually, the space in the DASD cache occupied by the virtual volume will be needed for other, more recently written virtual volumes. When that occurs, the ADSM HSM client truncates the file into a short data segment (approximately 3 KB) and appends a pointer to its ADSM server's database entry for the virtual volume. A volume in this state is referred to as *migrated*. If the host requests access to a migrated virtual volume, the

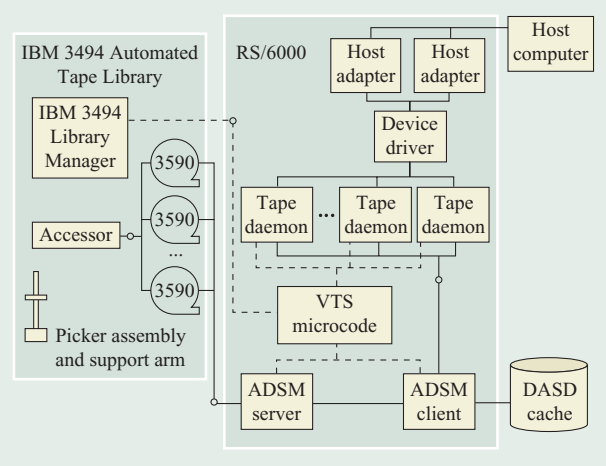

Figure 3

Virtual tape server microcode.

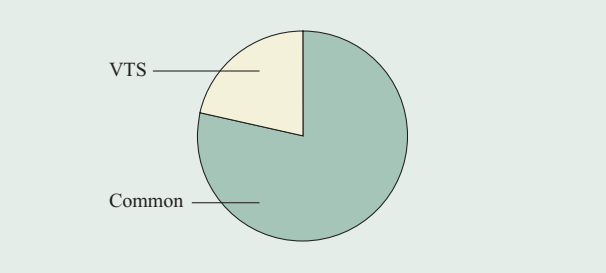

Figure 4

Virtual tape server functional microcode: IBM 3494 B16. (Does not include library manager or ADSM.)

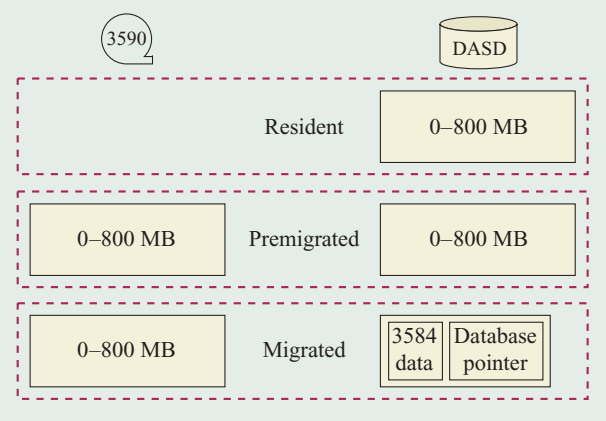

Figure 5

Virtual tape server file terminology.

ADSM HSM client and ADSM server transfer the data back into the VTS DASD cache from the appropriate physical tape. This operation is referred to as a *recall* of the virtual volume.

ADSM was chosen because of its performance history; however, integrating ADSM into the IBM VTS proved to be a significant challenge, because ADSM was designed to be serviced by a human operator. The ADSM HSM automigration daemon required that a significant amount of the VTS cache be empty. A quiet period for reconciling with the server was needed because the ADSM HSM automigration daemon was unable to premigrate or migrate data during this reconcile period.

The VTS design goals were to use almost 100% of the DASD cache and to run at full performance, 24 hours a day, seven days a week. The subsystem had to run with little or no human intervention. These design goals were in direct conflict with the limitations of the ADSM automigration daemon and the operator requirements of the ADSM server.

Integrating ADSM into the VTS presented many of the same challenges found in autonomic computing. The code had to be *self-configuring* and *self-healing*, replacing all of the configuration and repair activities performed by a human operator in a normal ADSM application. It also had to have *self-optimizing* characteristics to maximize the use of resources within the RS/6000.

The first step in integrating ADSM into the VTS was to limit the scope of the task by constraining the design of the system. Using an RS/6000 with ADSM required using the AIX HSM client and AIX server. Coupling the VTS with the IBM 3494 Automated Tape Library dictated the use of the IBM 3494 Library Manager (LM) for library control.

The IBM 3494 LM provides information to the VTS about the status of the physical tape cartridges. The LM also provides status information about the success or failure of physical tape cartridge mounts into the IBM 3590 tape drives and it tracks the status of the host computer's virtual volumes. The LM associates an inventory of virtual volumes with the VTS. When a host computer requests the mount of a virtual volume on a VTS, the LM verifies that the volume is in the inventory of the VTS. The LM also prevents the volume from being accessed simultaneously by multiple tape daemons on the VTS.

Additionally, the LM tracks the host category assignment for each virtual volume. The customer is allowed to assign any of these categories a *fast ready* attribute. This attribute is used by the VTS to determine how to open a virtual volume for the host. When a category is flagged as not *fast ready*, the mounts of virtual volumes in that category are assumed to be specific mounts; the data in the volume will be used by the host computer. When the host computer requests a mount of one of these virtual volumes on a virtual device, the VTS microcode checks to confirm that the entire volume is in the VTS cache and then returns a *ready* status on the virtual device. If the virtual volume is not in cache at the time of the mount, a recall will be performed, and the *ready* response to the host will be delayed.

When a category is flagged as *fast ready*, the mounts of virtual volumes in that category are assumed to be scratch mounts; the data in the volume will be completely overwritten, and the old version of the virtual volume will be destroyed. If a *fast ready* virtual volume is not in cache, it will not be recalled. Even though the volume will be overwritten, many host operating systems (for example, the IBM MVS* operating system) will read the IBM standard tape label at the front of the virtual volume to verify that the internal label is the correct one and that there are no data records of interest on the volume. The tape label is contained within the 3 KB of truncated virtual-volume data left in the VTS cache for migrated volumes. This small amount of header data is used to satisfy operating system requests without having to recall any additional data for migrated *fast ready* virtual volumes.

Automated administrator

One of the key functions of the VTS microcode is to automate the function of the ADSM system administrator. An example of one self-configuring function is cartridge insertion. When cartridges are added in a standard ADSM installation, it is up to the system administrator to determine whether they are new cartridges or reinserted cartridges, then check them into ADSM as appropriate. Within a VTS, the microcode detects the insertion, determines whether it is a scratch or reinserted private cartridge, and checks it in for appropriate handling.

Another area in which the administrator is replaced is error handling (**Figure 6**)—a self-healing aspect of the VTS microcode. When the VTS accesses a logical volume in its cache, it does so using AIX read and write commands. The ADSM HSM client intercepts these commands and automatically recalls virtual volumes into the VTS cache if required. Most errors that occur are labeled either *not ready* or *text busy* by the HSM client. These errors are reported on a virtual-volume basis. For example, a recall failure is reported to the VTS microcode by the HSM client as a *not ready* error on the virtual volume. The actual error that occurred on the physical tape being recalled is reported by the ADSM server as a tape error.

One thread of the VTS microcode is dedicated to monitoring the ADSM server output. It builds a volume status table of reported physical tape errors. When the

virtual tape daemon receives a *not ready* error, it calls a VTS error-handling thread that finds the physical tape on which the virtual volume is located and retrieves the appropriate physical tape error from the volume status table. This enables the VTS to take appropriate corrective action based on the error reported on the physical tape volume. In many cases, the errors are administrative errors and can be self-healed by the VTS microcode by taking the appropriate administrative actions. In other cases, the errors cannot be corrected by the VTS microcode, but information on the type of failure (e.g., missing or misplaced physical volume) can be provided to the user, who can then take appropriate action.

In the case of a *text busy* error while opening a virtual volume, the VTS microcode determines whether the file error occurred because another VTS microcode thread commanded ADSM to transfer the virtual volume from DASD to tape. Whenever the VTS microcode transfers a file from DASD to tape, it stores the filename in a common memory structure accessed by all VTS microcode components. The VTS error-handling thread checks the virtual volume that received the *text busy* error against these filenames and, if an error is found, it cancels the transfer process and the internal ADSM server session corresponding to that process. It waits until the file is freed and then instructs the virtual tape daemon to retry opening the virtual volume.

Occasionally, a physical tape has permanent read/write errors or enough temporary errors that it should no longer be used. The ADSM server detects this and makes the tape read-only. In addition, the VTS thread monitoring the ADSM server output makes tapes read-only on specific ADSM errors. The VTS microcode periodically checks for these tapes and processes them. It determines the current status of each virtual volume on the physical tape and attempts to recover it into cache. Virtual volumes that cannot be recovered are reported to the user. When the tape has had all recoverable data removed from it, it is ejected from the VTS, and the customer is notified. The data recovered into cache is premigrated onto other, good physical tapes.

The administrator also self-configures the physical tape drives used in the VTS. If too many permanent errors have occurred on a particular physical tape drive, the ADSM server will stop using it. The VTS microcode detects these unusable drives by detecting specific server error messages that indicate that a drive is no longer being used. The microcode also periodically queries the ADSM server to determine any additional drives that are no longer being used. The LM can also make tape drives unavailable on physical load and unload failures. When any of these conditions are detected, the VTS microcode deconfigures the drive from ADSM and reconfigures the ADSM *mountlimit*. Then, if appropriate, it sends a message to the LM to post an intervention that a drive has failed. When the drive is repaired and made available through the LM user-interface panels, the VTS is notified by the LM. The VTS microcode then configures the drive to the ADSM server and reconfigures the ADSM *mountlimit*.

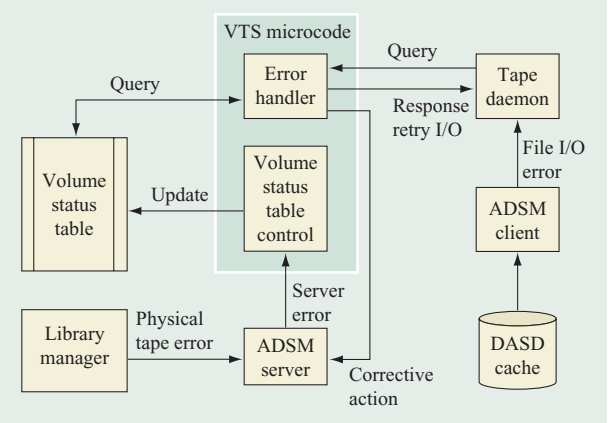

Figure 6

Virtual tape server error handling.

Cache management

A key self-optimizing function provided by the VTS microcode is cache management. The VTS microcode is designed to monitor all virtual volumes in the VTS DASD cache and instructs the HSM client to transfer them to tape on an individual basis. This function helps improve the HSM client automigration function in two ways. First, it allows the VTS to manage files in orders other than least recently used (LRU). Second, it allows the VTS to continue to transfer files from cache to physical tape while the ADSM HSM client and ADSM server are reconciling, a process that can take more than an hour. (The original ADSM automigration process suspended transfers to tape while reconcile was running.)

Each virtual volume is assigned a pseudo-time value used for ordering logical volumes in cache. The virtual volume with the oldest pseudo-time is the preferred file for both premigration and migration. The VTS allows the host to specify the desired cache management policy applied to a particular virtual volume every time it is accessed. Currently these choices are *keep in cache* (LRU) and *remove from cache* (preferential migration). For virtual volumes that are to be treated on an LRU basis, the pseudo-time assigned is the time the virtual volume was last accessed by a host computer. Virtual volumes that will be preferentially migrated from the cache are assigned pseudo-times near the AIX epoch time, making them appear very old.

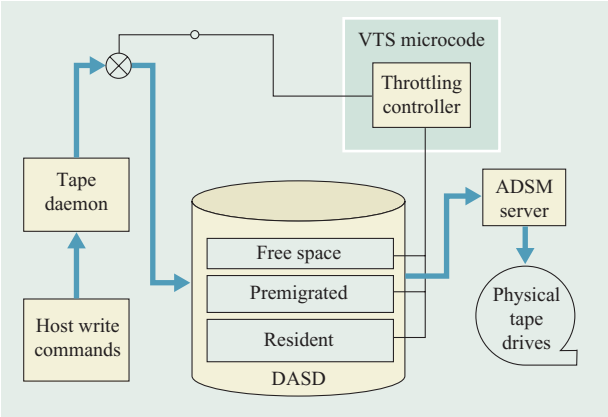

Figure 7

Virtual tape server host write-throttling.

The pseudo-time mechanism allows the VTS to treat all files with the same algorithm (oldest pseudo-time is premigrated and migrated first), allowing host-controlled ordering of files and LRU ordering of other files to be managed using a single value.

Dynamic load balancing

Another key function supported by the VTS microcode is balancing the data transfer on and off the VTS DASD cache. When the host computer is writing data to the VTS, the tape daemons fill the DASD with newly written resident virtual-volume data. Space must be created within the cache to store this data by truncating previously premigrated DASD virtual volumes. To have virtual volumes available for truncating, other resident virtual volumes must be premigrated to physical tapes.

ADSM performs the premigration of the virtual volumes from DASD to tape under control of the VTS microcode. Under heavy host loads, the DASD can fill faster than it can empty. If it should fill up completely, the VTS could lock up or suffer other operational problems that can have a serious impact on the jobs being run by the host.

The solution implemented in the VTS is *host write-throttling* (**Figure 7**). Every 30 seconds, one thread in the VTS microcode checks the DASD free space and the amount of resident virtual-volume data.

Host write-throttling is controlled by VTS microcode. On the basis of a throttling algorithm, VTS microcode loads a write delay time into a memory variable shared with the tape daemon. The actual host write-throttling is handled by the tape daemon. After every write operation processed by the tape daemon (typically 32 KB), the tape daemon *sleeps* for a specified delay time indicated by the VTS microcode before signaling that it has completed the operation. If throttling is not required, the tape daemon branches around the sleep (rather than sleeping for zero time) to avoid giving up any processor time slices, thus maintaining maximum throughput.

The maximum throttling time delay is set at a value that might not abort a host job, but is large enough to effectively stop host write activity to the DASD. The VTS currently uses 120 seconds as the maximum throttling time delay; this reduces the effective maximum host write rate to the VTS by several orders of magnitude.

The VTS microcode controls the ADSM host recall data flow, which also fills the cache with premigrated data, by reducing the number of ADSM recall daemons (and therefore the number of tapes that can be recalled by ADSM). Using a throttling algorithm, VTS microcode sets the appropriate number of recall daemons that ADSM can use. In normal (not throttled) operation, VTS microcode sets the number of recall daemons to allow ADSM to recall up to $n - 1$ physical volumes, where n is the number of 3590 drives in the VTS. At the maximum threshold, VTS microcode allows only one recall to occur at a time (the minimum number of recalls with which the ADSM can continue to operate).

The algorithm currently used to implement thresholding is linear. The time delay is zero at the minimum threshold and 120 seconds at the maximum threshold. The actual free space on the DASD is determined by a VTS microcode time poll. The time delay is linearly interpolated on the basis of the actual free disk space between the thresholds. Similarly, the number of simultaneous recalls allowed at the minimum threshold is $n - 2$ (one less than when the VTS is not throttled). The number of simultaneous recalls allowed at the maximum threshold is one. The number of recall daemons is linearly interpolated on the basis of free disk space.

In addition, when the amount of resident data reaches 30 GB from the top of the cache, the VTS starts a linear throttle on the amount of premigration data. This begins at zero and reaches a maximum value of two seconds if resident data fills the cache. This throttle establishes a minimum buffer of premigrated files in the VTS cache. These premigrated files can be migrated quickly, freeing space in cache. This buffer is intended to isolate the host write data flow from variations in the premigration data rate, which occur between virtual volumes and as physical tapes load and unload. This throttle is the predominant throttle in the VTS with the performance accelerator feature.

The throttling mechanism results in two distinct modes of operation for the VTS: *peak* and *steady state*. Peak mode occurs when the majority of the virtual volumes in the cache are premigrated. The VTS allows the host to write data without any throttling until the minimum

premigration throttle threshold is reached. During this period, data is premigrated to tape from the cache, but, typically, the premigration does not significantly reduce the host write rate. When the minimum premigration throttle threshold is reached, the VTS maintains a stable amount of premigrated data using the host write throttle. The host write rate is reduced to exactly match the premigration rate of the VTS. By reducing the host write rate, the premigration processes receive more system resources and run faster until the DASD is fully utilized. This throttling mechanism is used by the VTS microcode to self-optimize the data flow in the VTS.

Another advantage of the VTS is better use of the cache because of its tighter control mechanism. If the VTS used the ADSM automigration client and suspended automigration while reconcile was running, several of the VTS configurations would end up overrunning their caches, even if they were empty at the start of the reconcile. The VTS, using its cache-control algorithm, requires so little free space that the larger cache models report the cache continually at 100% full and support continuous full-throughput operation.

Background process control

The VTS also self-optimizes background processes to minimize their impact on the user. Most customers use the VTS in a cyclic manner, with heavy batch use periods alternating with periods of much lower use. The VTS uses its microcode in conjunction with customer input through the LM to minimize the impact of these background processes on the customer's batch window.

The VTS controls the reconcile of the VTS client and server. This DASD and processor-intensive process is used by ADSM to cause down-level versions of reused or deleted virtual volumes on the physical tapes to expire. When enough data on a particular physical tape has expired, the remaining data can be removed from the tape through a reclamation process, and the physical tape can be reused in the VTS.

The reconcile process can interfere with the performance of the VTS if it occurs during a period of heavy VTS use. This process can take more than an hour on a VTS that has a full complement of virtual volumes in use. The VTS attempts to match its reconcile timing with the periods of lower use by the customer.

The design goal of the VTS was for a reconcile to occur at least once every 24 hours. The VTS monitors its loading by keeping track of the amount of time that at least m physical drives have been continuously idle for all values of m from 1 to the number of physical VTS drives. This table can be used to determine periods of low resource use. If enough drives have been continuously free for an empirically determined period of time, and it has been at least 18 hours since the last reconcile, the next reconcile is executed. If the idle time criterion is not met within 24 hours after the last reconcile ended, a reconcile is executed.

This mechanism allows the VTS to shift the reconciliation time to a quiet period up to six hours ahead of the scheduled reconcile, allowing the reconcile process to shift backward in time to an optimal quiet period. If no such period is found, the reconcile occurs 24 hours after the end of the previous reconcile. Thus, the next reconcile will occur at a period slightly greater than 24 hours, allowing the reconcile to creep forward in time to an optimal quiet period. This permits the VTS to eventually self-optimize the reconcile to the customer's low-use periods.

When the reconcile operation completes, the ADSM server causes the older tape copies of customer virtual volumes that have been updated or deleted from the VTS to expire. This helps reduce the amount of active data on the physical tapes in the VTS. The active data on the physical volume can be reclaimed by appending the remaining active data onto a filling volume, helping to eliminate the gaps created by the reconcile process and converting the physical volume into an empty tape. This process requires physical drives and reduces the ability of the VTS to perform customer data transfer. Therefore, customers can specify an active data threshold at which they want the reclamation process to begin and determine a daily schedule that tells the VTS when it can perform this process. The VTS microcode can use this information to help optimize the reclamation process. If the VTS determines that the user has ample scratch tapes, the VTS monitors the use of the physical drives in the VTS and performs reclamation only if enough drives are free for a significant period of time and reclamation is allowed. If there are not enough scratch tapes, the VTS cycles reclamation on and off with the customer-supplied schedule. If the user is low on scratch tapes, the VTS temporarily adjusts the reclamation threshold high enough to remedy the low scratch-tape condition and reclaims continually, ignoring the customer-supplied reclamation schedule until the customer has enough scratch tapes.

Enhanced database backup

The VTS provides enhanced data protection with a design that stores a complete ADSM metadata database backup at the end of every physical tape. At every physical volume mount, the VTS microcode instructs the 3590 tape drive to return a logical end-of-tape signal when there is just enough space at the end of the tape to store a full backup of the ADSM database. When the ADSM server receives the logical end-of-tape signal, it stops writing to the tape, writes its closing data, and passes the device handle to the VTS microcode. The VTS microcode writes a database

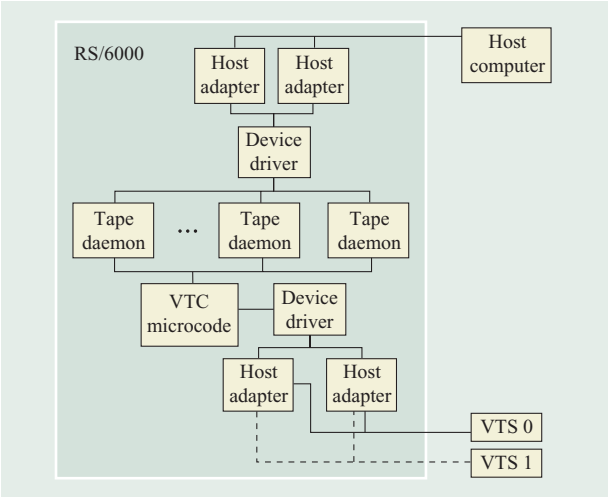

Figure 8

Virtual tape controller microcode.

backup header to the tape that contains backup version identifiers and the time at which the backup was taken. The VTS microcode instructs the ADSM to create a full database backup and transfers it to the end of the physical tape, then returns control of the physical tape drive to the ADSM server. The server closes the tape and dismounts it from the tape drive.

Under normal operation, every full physical volume contains a full ADSM metadata database backup representing the point in time at which the physical tape filled up completely. This means that any subset of physical tapes from the VTS can have a point-in-time restore from the physical tape that has the oldest backup time, and that backup will contain the database that describes the contents of all the other tapes in the subset. In the event of a complete VTS failure, or a scenario in which a VTS must be abandoned and a subset of the physical tapes can be removed (such as a flood), the database can typically be recovered by a VTS process that scans the tapes and finds the newest database, then restores it.

Peer-to-peer virtual tape server

When a single physical volume is written on a physical tape drive, only a single copy of that volume is written to a physical tape. Similarly, the VTS provides only one physical copy of a virtual volume on one of the VTS stacked volumes. Although rare, it is possible for that physical volume to be destroyed or become unreadable for other reasons. If this should occur, the virtual volume would be permanently irretrievable, just as it would be if it were a physical tape volume with the same damage.

The best way users can protect access to all tape volumes—virtual or physical—is to maintain dual copies of the data. This can be accomplished by running host tape-backup applications or by having host applications write two copies of every volume. Many customers prefer that this copying operation take place in a subsystem that is external to the host, offloading the work from the host onto the subsystem controller.

For these customers, IBM added the peer-to-peer virtual tape controller (VTC). Two virtual tape servers can be combined to create a peer-to-peer virtual tape server by connecting them using four or eight VTCs. The host computer is connected to the VTC, and the VTC in turn is connected to both VTS subsystems. The VTC confirms that each virtual tape server contains a copy of every virtual volume written to the VTC. If the VTC cannot access a virtual volume on one VTS, it can access it on the other and replace the damaged virtual volume on the first VTS. This redundant self-healing aspect of the architecture provides greater data availability in the event of a failure than either a VTS or a physical tape.

The VTC reuses many microcode components from the VTS (**Figure 8**). Custom VTC microcode was written to mimic a host computer attachment to the VTS to control the virtual volumes and manage mounts on the attached VTS subsystems. To control the virtual volumes on the two VTS subsystems and to manage the replication and synchronization of data between them, the VTC uses tokens that indicate data and property levels of each virtual volume managed by the VTC.

In a peer-to-peer VTS, the VTC selects the I/O VTS that will be used for host I/O operations. Host data is written through the VTC to the I/O VTS. The VTC then manages the duplication of the virtual volume to the other VTS in the peer-to-peer subsystem. Two virtual-volume-synchronizing modes of operation are provided in the peer-to-peer VTS. The first mode of operation is called *immediate mode*. In this mode, after an entire virtual volume is written to the I/O VTS and the host has issued a rewind/unload operation to the virtual device, the VTC creates a copy of the virtual volume on the other VTS before it reports the completion of the rewind/unload operation to the host. Therefore, when the VTC indicates that the rewind/unload operation of the virtual volume has completed, the copy has been replicated on the other VTS.

In the second mode of operation, called *deferred mode*, copies to the non-I/O VTS are deferred. The customer configures the VTCs to increase the priority of these deferred copies if they have not been performed within a certain period of time. This allows the VTC to provide all of its processing power to support the host computer during peak batch windows. This mode of operation provides the customer with improved batch performance and defers the copy activity to off-peak periods. As each

deferred copy is added to a VTC copy queue, it is also broadcast to the other VTCs in the subsystem, allowing the copy to be completed by the VTC with the most bandwidth available.

However, in deferred mode, if the amount of data written in the batch window should exceed the capacity of the VTS cache, some of the uncopied virtual volumes would have to be migrated to physical tapes. To complete the copy to the other VTS at a later time, these virtual volumes would have to be recalled from tape—a very inefficient process. This is prevented in a peer-to-peer VTS by having each VTS dynamically track the amount of uncopied virtual-volume data it has in cache. If this amount of uncopied data exceeds a threshold based on the size of the VTS cache, the VTS slows down all incoming cache write activities until the rate generated by uncopied data matches the rate of data copied to the other VTS.

In addition, the uncopied virtual volume is kept preferentially in cache by adding to the pseudo-time cache preference value for the virtual volume, a pseudo-time offset that is typically larger than the cache residency time. This makes the virtual volume appear newer than all of the other volumes in the cache, and it is therefore the last file to be removed from the cache. The VTS monitors the copy status of this source virtual volume and, when it is copied, subtracts the pseudo-time offset from the cache preference value of the source logical volume, restoring it to its original LRU order in the cache.

In a peer-to-peer VTS, when a virtual volume is copied from one VTS to the other, the target VTS sets a cache pseudo-time value near the AIX epoch for its copy. The source VTS maintains a pseudo-time value of the last host access time of the virtual volume. This results in the virtual volume being promptly purged from the target VTS and maintained on an LRU basis on the source VTS. This automatic processing maintains only a single copy of the virtual volume between the two VTS caches, essentially doubling the amount of unique data stored in the combined VTS cache when compared with operating both caches on an LRU basis.

If, in spite of the controls, uncopied virtual volumes have to be migrated to physical tapes (for example, if the other VTS subsystem has a long service outage), the virtual volumes will be staged back in to the cache for copying. These files will also have a pseudo-time offset added to their recall time as their cache preference value to keep them preferentially in cache. When these files have been copied, a larger pseudo-time offset will be subtracted from their cache preference values to give them less preference in cache than the LRU-managed files in the cache, but more preference than the *remove from cache* virtual volumes.

The staging of the uncopied virtual volumes is optimized by the VTS. The VTC requests the staging of a particular virtual volume on the basis of its copy queue. The VTS determines the physical volume that contains the requested virtual volume. The VTS builds a list of virtual volumes that also require staging based on their tokens and are also on the same physical volume. The VTS recalls the entire list of virtual volumes and notifies the VTC which virtual volumes have been staged for copying. The VTC copies the staged virtual volumes to the other VTS on a priority basis. This mechanism greatly reduces the number of physical mounts required for the copy operation.

Peer-to-peer VTS token manager

In a peer-to-peer VTS, the tokens used for tracking the status of the virtual volumes in each VTS are critical to keeping the data synchronized and must be kept in a database within the peer-to-peer subsystem. Because these tokens must persist and be tightly coupled with the virtual-volume data on each VTS, it was decided to add a token manager to the VTS. This token manager stores the tokens for the VTC and provides query and update functions to the VTC. The token manager synchronizes a token database backup and writes it to the end of every physical tape along with the ADSM database to provide a full point-in-time restore of all critical data for a peer-to-peer VTS.

The VTS token manager also independently tracks the data and property levels of the copy of each virtual volume in cache as well as the data and property levels of the copy of the virtual volume that was last written to physical tape. During a disaster recovery, after the point-in-time databases are restored, the VTS microcode determines which volumes were recovered from tape and which remained in cache. It restores the proper data and property levels for each recovered volume depending on its recovery method.

The token manager is the first component in the VTS designed to be completely self-healing. It services other VTS components by responding to request messages on an AIX message queue. Every message received is immediately copied to an *in-process* message queue. The request is processed by the token manager; then a response message is returned to the requester and the in-process message is flushed. This mechanism has every *in-flight* message in one message queue or another. A watchdog component in the VTS periodically sends "ping" requests to the token manager and will halt and restart the token manager if problems are detected.

This means that the token manager can run problem-free, even with a slow memory leak. Eventually, the token manager would perform a core dump, and the watchdog component would restart it. The new token manager would reexecute the request that was in progress, then continue processing requests. The calling processes would

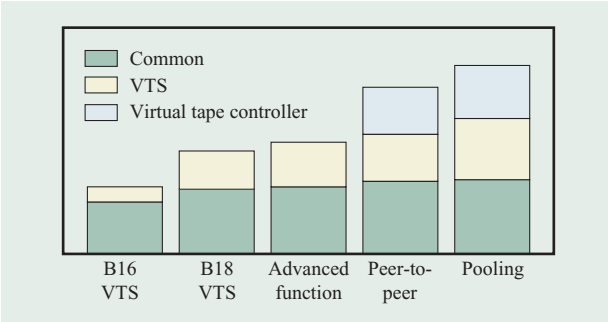

Figure 9

Virtual tape server microcode growth. (Does not include library manager or ADSM.)

experience a delay in response times, but otherwise could continue uninterrupted. In case the in-progress request caused the core dump, in-progress requests are retried only on the first error restart.

In practice, restarts on the token manager are not expected, but designing the code to be restartable at any time has been a benefit for code development. If a change is made to the token manager, it can be stopped, reloaded with a newer version, and restarted with no interruption to the running VTS.

Microcode growth

With each new VTS microcode release, the function, capacity, or performance of the VTS has improved. These changes all built on the autonomic computing aspects in the original VTS microcode. Each new function required specific VTS microcode to provide additional automated management controls within the VTS. The growth in the VTS functional code has consisted primarily of additions to the VTS specific code, while the common control unit code has remained stable (**Figure 9**).

The VTS microcode was initially envisioned as the autonomic computing glue that bound the major VTS code components together. Since then, it has grown from mere glue to a major controlling element of the VTS. It provides the self-configuring and self-healing control of the ADSM. The code also improves the database backup process and provides token support in the peer-to-peer VTS. It self-optimizes the VTS by controlling the background VTS operations, dynamically load-balancing the other VTS components, and managing the contents of the VTS cache.

Summary

The IBM Virtual Tape Server reduces the cost of tape operations through tape-drive virtualization and physical tape-volume stacking. The VTS microcode architecture has incorporated existing, historically reliable microcode building blocks to provide a stable data processing foundation. The addition of autonomic controls has enabled the integration of these building blocks into a reliable, high-performance, standalone storage subsystem.

Looking forward

The next major functional improvement in the VTS is the addition of pooling: the ability for customers to control which sets of physical tapes are used to store their virtual volumes and to select whether they want their virtual volumes written to only one or to two different storage pools. These copies, if chosen, provide additional protection against rare, but possible, physical tape failures.

Each pool can have specific types of physical media assigned to it by the user. This gives customers the advantage of managing virtual volumes for performance or storage capacity by pool. Or, if desired, customers can separate groups of virtual volumes for security reasons. Some enhancements are being provided to automate the transition from one physical medium type to another, further enhancing the *virtual* nature of the VTS.

As IBM moves more of its storage onto storage area networks (SANs), the VTS will have to evolve into a network-attached storage subsystem. The VTS could become more virtual itself as its component subsystems are separated and scatted across the SAN.

Recent world events have sharpened customers' focus on disaster recovery. The peer-to-peer VTS helps support customers' continuous access to their data in the event of a total loss of one of the VTS subsystems. Many customers also desire the ability to seamlessly verify their disaster-recovery processes. IBM continues to develop additional VTS features to improve the efficiency and ease of use of disaster-recovery verification.

Finally, as IBM rolls out the results of the autonomic computing initiative throughout its product line, it is expected that the VTS will become even more reliable as its building blocks improve. The plan for the VTS is to couple this improvement with improvements to its own autonomic computing base to sustain our technology leadership in virtual tape subsystems.

Acknowledgment

The author thanks R. Bradshaw for his help in the preparation of this manuscript.

References

1. J. P. Harris, W. B. Phillips, J. F. Wells, and W. D. Winger, "Innovations in the Design of Magnetic Tape Subsystems," *IBM J. Res. & Dev.* **25,** No. 5, 691–699 (September 1981).

2. T. Ouellette, "Virtual Tape Crams In Data, Cuts Storage Costs," *Computerworld* **31,** No. 14, 1–2 (April 1977).
3. *Ibid*., p. 1.
4. W. Greeley, "Virtual Tape—A Year of Experience," *Computer Technol. Rev.* **18,** No. 7, 28–34 (July 1998).
5. N. Dillon, "Virtual Tape Sets Save Money, Time," *Computerworld* **32,** No. 5, 61–62 (February 1998).
6. N. Dillon, "IBM Speeds Up Virtual Tape System," *Computerworld* **32,** No. 43, 59–60 (October 1998).

Received July 23, 2002; accepted for publication January 7, 2003

Gregory T. Kishi *IBM Systems Group, 9000 South Rita Road, Tucson, Arizona 85744 (motorman@us.ibm.com).*
Mr. Kishi is a Senior Technical Staff Member in the VTS/Controller Development Department in the Systems Group. He received his B.Sc. degree in chemical engineering from the University of California at Davis. Mr. Kishi received an IBM Corporate Award for the IBM Virtual Tape Server. He has also received two IBM Outstanding Innovation Awards and four IBM Outstanding Technical Achievement Awards. He has more than 50 patents issued or filed and has received 17 IBM Invention Achievement Plateau Awards.