The Cray Tape Management System for Solaris

John C. Falkenthal and *Karen D. Reynolds*, Cray Business Systems, San Diego, CA

ABSTRACT: *The Cray Tape Management System* (Cray TMS) *manages system tape resources including the volumes of tape media and the devices used to access them. The devices under Cray TMS management are protected from unauthorized use, and are automatically allocated to users as needed. Cray TMS works in conjunction with a tape catalog manager, Cray/REELlibrarian* (CRL). *This paper discusses the technical challenges of implementing this product on a Solaris system base as well as the functional capability of the product and its relationship to the UNICOS Tape Subsystem.*

1 Introduction

Throughout the evolution of the UNIX operating system, support for tape intensive data-processing has remained deficient. The typical UNIX tape implementation provides uncontrolled single-user access to pre-loaded manual tape drives. No access control for tape volumes, nor sharing of drive resources, is enforced by the operating system. This was also true of Solaris[1], until now. Cray TMS provides sophisticated tape handling capabilities to the enterprise computing customer. Cray TMS runs on the Cray Superserver 6400 (CS6400) system under the Cray Solaris 2.4 or higher operating system.

Cray TMS is targeted to serve customers who, because of cost considerations, are moving from mainframe and minicomputer systems with proprietary tape management systems onto open systems servers (like the CS6400) running a standard UNIX environment with no tape management system. These customers require a replacement tape management system to administer their tape libraries and to control access to the data on their tapes. The primary goal of Cray TMS is to serve the needs of these customers.

Cray TMS supports both manually loaded tape drives, like those normally found in open-systems environments (*i.e.* 8mm, 3480 cartridge, 9-track), as well as SCSI-based media changers, and the Storage Technology ACS family of robotic tape systems.

A future goal of Cray TMS is to provide integrated tape management services for third party applications which use tapes for data processing and storage. Currently, each of these applications is required to implement their own internal tape management system to service the application. With an autonomous tape management system in each of these applications, the system tape resources cannot be managed effectively.

2 Cray TMS Design

Cray TMS was designed by using the field-proven UNICOS Tape Subsystem[2] as the functional model of a tape management system for the UNIX operating system. The UNICOS Tape Subsystem provides access protection services for tape data by controlling tape drive allocation and by performing system label processing for IBM standard[3] or ANSI standard[4] labeled tapes. The UNICOS Tape Subsystem optionally uses the services of a tape catalog for specification of the access permissions granted to volumes in the tape library. Cray Research released Cray/REELlibrarian (CRL) in 1992 to provide stand-alone UNICOS tape catalog services. Porting CRL to the CS6400 to provide tape catalog services for Cray TMS was the obvious choice.

Also in 1992, SunSoft, announced the availability of another technology which Cray TMS would leverage, *Solaris volume management.* Papers describing volume management were presented at the USENIX Winter Conference[5] and the UK UNIX User Group/Sun UK User Group Conference[6] in January, 1993. Although Solaris volume management supported only removable disk media (floppies and CD-ROMs), it was designed such that it could be extended to support other types of removable media, such as tapes. Volume management provides access protection for data on removable

media. Cray TMS required a mechanism for providing similar access controls for tape data, and so, leveraged existing technology by extending Solaris volume management to support tapes.

What volume management did not provide was any formal mechanisms for drive allocation. An essential function of the UNICOS Tape Subsystem is to allocate tape drives in a deadlock-free manner among all system and user processes. It was decided to further extend Solaris volume management by adding a tape drive reservation system that employs Dijkstra's Banker's Algorithm[7], in a manner similar to that of the UNICOS Tape Subsystem.

Commands were needed to access the Cray TMS functionality. Since that functionality was modeled after the UNICOS Tape Subsystem, another obvious choice was to adopt the UNICOS Tape Subsystem user command interfaces. It was apparent that the similarities between Cray TMS and the UNICOS Tape Subsystem may lead a casual observer to believe the two products are totally compatible, or that Cray TMS is a "port" of the UNICOS Tape Subsystem. Given the radically different architectures of the two products, and the different customer base for which they are targeted, it was clear during design that total compatibility was not achievable. Although vast similarities do exist between the two products, and therefore, compatibilities, they are different in many ways.

The functionality which was still missing once all of the obvious design decisions had been made, was system label processing, which is required for accessing data from labeled tapes. In the UNICOS Tape Subsystem, this function is accomplished by cooperation between the tape daemon and the tape driver. The decision was made to develop a label processing daemon and to develop a special interface to the managed tape driver (MTD) which would allow the standard label processing daemon (and in the spirit of open systems, any proprietary label processing daemon) to monitor user tape accesses and perform the necessary I/O to process the label records on the tape

3 Cray TMS Architecture

Cray TMS is composed of three distinct elements: Cray/REELlibrarian (CRL), the StorageTek ACS client interface (for configurations with STK tape silos), and the tape management system, or TMS. Both CRL and the StorageTek ACS client are software components that were ported to the CS6400. Both are fully described in other publications. Only the TMS architecture is addressed here.

3.1 Solaris Volume Management

At the architectural center of Cray TMS is Solaris volume management, whose significant components are a multi-threaded daemon (vold) and a pseudo-driver (vol). Cray TMS duplicates and extends these components, re-naming them tvold and tvol, respectively. The tvold daemon provides access to volumes of tape media via character special device files in an NFS file system, referred to as the tvol file system, or **tvolfs**(7). The file system structure is contained within a memory resident, configurable database. The tvold daemon mounts the file system and establishes itself as the server for the file system. To respond to file system operations, the database information is translated into the appropriate NFS server responses. By this method, the file system appears to contain unique devices for every volume in the database. The volume devices belong to the tvol pseudo-driver, which performs physical I/O by passing the I/O requests down to the Cray Solaris SCSI tape driver, **st**(7).

Cray TMS extends the Solaris volume management architecture in two significant ways:

1. Tape drive reservation and allocation services (an implementation of the Banker's Algorithm) were added to the tvold daemon. Reservation services are requested via a new RPC protocol.
2. A new *pseudo-driver*, the Managed Tape Driver (MTD), was developed to provide device-special files for labeled tape volumes in the tvol file system. MTD monitors all user I/O requests to labeled tapes.

The tvold daemon opens the **st**(7) driver device files for TMS managed tape devices. These devices must be opened in order to fulfill the I/O requests passed down from the upper driver layers (MTD and tvol). tvold opens a special tvol driver device, the control device, which it uses for communication with the tvol driver. This communication is in the form of **ioctl**(2) system calls.

The tape drive reservations server collects requests for drives from user sessions. As volume management performs the tasks of establishing access for a tape volume, it queries the drive reservations server for permission to allocate a drive in accordance with the reservations that have been granted to the user.

The tvold daemon is structured such that certain functional modules are dynamically linked to the daemon in accordance with the tvold daemon configuration file, **tvold.conf**(4).

Table 1. tvold Daemon Functional Modules

Shared Object	Functional Module Description
db_crl.so	Volume management database module
label_ansi.so	ANSI label recognizer module
label_ibm.so	IBM label recognizer module
label_unlab.so	Unlabeled tape recognizer module
dev_st.so	Manual-loaded device interface module
dev_sga.so	SCSI media-changer device interface
dev_acsapi.so	STK Silo device interface module
lyr_mtd.so	Layered device management module

The modules listed in Table 1 include the database module, the media label recognizing modules, the device interface

modules, and the layered device management module. The last of these modules, lyr_mtd.so, was added to manage MTD device allocation for labeled tape volumes. Each module in Table 1 has a "type". Each module type has a unique entry and structure of function vectors.

New functional modules were written to replace the Solaris database module, the media recognizing modules, and the device interface modules. To support tapes, the device interface module was extended with the addition of new function entry points. These new functions were necessary to address differences between tapes and disks. Unlike disk volumes, tape volumes have multiple devices (representing different available access modes) which must be managed such that at most one user can open any of these devices and at most one device is in use at any time. [Disk volumes can have multiple devices (representing different disk partitions), but due to the random access nature of the drives, the driver allows any number of these devices to be simultaneously open by any number of users].

3.2 tvol File System Organization

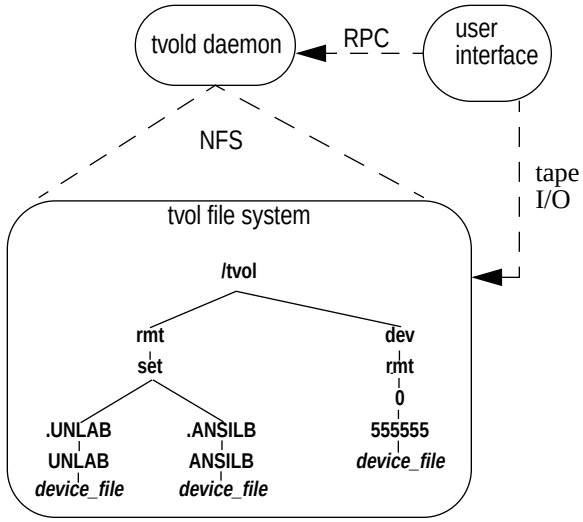

Figure 1: tvol File System

As shown in Figure 1, there are two distinct directory hierarchies in the /tvol file system, /tvol/rmt/set and /tvol/dev/rmt. /tvol/rmt/set contains "logical" paths to tape volumes. This means there are no tape drive specific components in the pathname. /tvol/dev/rmt contains "physical" paths to tape volumes. Physical pathnames contain the tape drive upon which the volume has been mounted, and are therefore only present in the tvol file system while the volume is mounted. Allowing legacy Solaris tape applications to run in the presence of Cray TMS is one use of the physical paths in the tvol file system. The physical paths represent a method of tape access that is compatible with standard Solaris tape access.

/tvol/rmt/set/.ANSILB/ANSILB/*device_file* is the "logical name" of the tape volume with volume serial number (VSN) 'ANSILB'. The '.ANSILB' component of the pathname is the CRL volume-set identifier (or VSID), for this tape volume. The CRL volume-set ID is the VSN of the first volume of the multi-volume set, prefixed with a dot "." character. /tvol/dev/rmt/0/555555/*device_file* is the "physical name" of the tape volume 555555 after it has been mounted on drive 0.

device_file at the leaf of each path represents the actual character special device file for the tape volume, as described in **st**(7). For labeled tapes, the *device_file* belongs to the MTD driver. For unlabeled volumes, the *device_file* belongs to the tvol driver. Also found in the tvol file system are several symbolic links which are used as aliases for accessing tape volumes. These aliases make it simpler for the user to specify certain types of tape volume access. These aliases are described more fully in **tvolfs**(7).

This organization of device files is unique to Cray TMS. The UNICOS Tape Subsystem creates device files at locations specified by the user (tpmnt -p <file>). With Cray TMS, the user specification is created as a symbolic link to the actual device file in the tvol file system.

3.3 The Cray TMS Database Module

The file system representation of the volume database is intrinsic to the volume management design. All normal file system activities (**chown**(1), **chgrp**(1), **chmod**(1), **mv**(1), **cp**(1), **ln**(1), **rm**(1), **mkdir**(1), **rmdir**(1), etc.) are supported by the database module interface. In fact, all database functionality (add, update, delete) is triggered by file system activity. A new volume is added to the internal volume management database when it is inserted into a drive, recognized (and named) by one of the media label recognizer modules, then placed into the file system. A volume is removed from the database when an **rm**(1) command causes invocation of the database "remove" function. Volume attributes which could not be derived from file system activity, were supported by tvol device **ioctl**(2)s which allows arbitrary character strings to be stored in, and retrieved from, the database record for that volume.

In the Cray TMS database module (which uses CRL as the external tape catalog), there were three reasons for backing away from the one to one relationship between the file system and database.

1. The available number of volume devices (minor device numbers) in the file system could not support the design target one million volume tape library.
2. The CRL database supported the data objects required for its task as tape catalog; it could not be easily extended to also store information for other tvol file system objects (*e.g.*, directories and links).
3. The rules by which CRL grants permissions for database updates do not match the rules by which a file system grants permission for the corresponding updates. For example, CRL can be configured so that an ordinary user cannot add a new volume to the tape catalog without administrator action.

Because of CRL's inability to "store" the directory/file hierarchy of the file system, a very limited, immutable hierarchy is created by the database module logic during system initialization. All of the CRL volume sets (multi-volume tapes) are located in directory /tvol/rmt/set; the file system hierarchy below that directory is determined by the number and names of the CRL volume sets and their constituent tape volumes.

A limitation derives from the decision not to add special interfaces to allow for a database-unique form of permission check before a database update (and related file system change) was allowed. Without this permission check, volume management cannot enforce the permissions established for CRL database record updates; the solution is for the database module to deny permission for all requested updates where the CRL permissions cannot be checked. Since CRL has a user interface with more update capabilities than those provided by the file system, there is no actual loss of functionality and a minimal impact to system usability.

An additional limitation of the Cray TMS tvol file system is that only a small subset of the directory/file hierarchy is present at any one time. The directory/file hierarchy is "pruned" by only maintaining a "file" representation for the most recently used volumes. Hence, the tvol file system contains a "cache" of selected CRL volumes. Volume device unit numbers are recycled among the "cached" volumes, which allows at least 16K tape volumes in the cache at any time.

A volume becomes "cached" in one of two ways:

1. Its tvol file system path is referenced. *e.g.*, `open("/tvol/rmt/set/.ANSILB/ANSILB/mn", O_RDWR);`.
2. The volume is inserted into a Cray TMS drive, recognized by one of the media label recognizer modules, and found to be in the CRL catalog.

The algorithm used to "cache" a volume required one change to the volume management database module interface. The name of the "file" (volume) sought by a directory lookup had to be provided so that the database module could "cache" it. (A directory lookup is performed to update the "files" in the directory before the directory contents are examined.)

An attempt is made to "de-cache" a volume after it has remained unreferenced for a period of time in the database module cache. All copies of the object must be removed to "de-cache" it. Since the tvold daemon keeps copies of objects which it retrieves from the database module, the database module cannot know whether it is safe to "de-cache" a volume.

The algorithm used to "de-cache" a volume utilizes the existing interfaces between the tvold daemon and the database module. When the tvold daemon (usually in its role as NFS server) performs a directory lookup, it expects the database module to mark the "files" in the directory as they are updated. Any "files" not marked are assumed to have been removed from the database, and the tvold daemon will remove its copies of the "file" objects unless they are still in use (accessed by a user, waiting to be mounted, or mounted in a drive). If the copies cannot be removed, the tvold daemon will continue to treat them as normal file system objects. This implies that the objects will be passed to the database anytime they are updated for the updates to be stored, and the objects will still be present in a directory when the database is asked to perform a directory lookup.

Using this interface, the database module attempts to "de-cache" a volume by not marking its "file" as updated on a directory lookup (*i.e.*, the database module claims that this object has been removed from the database). If that volume is not in use, the tvold daemon will remove all of its copies after the directory lookup completes. If not, the database module will see the tvold daemon copy of the volume object on a subsequent update or directory lookup request. At that point, the database module will make its own copy of the object, restore the object to the database "cache", and start aging the object once more for a later "de-cache" attempt.

3.4 Device Drivers

Cray TMS employs a layered driver architecture. The tvol and MTD pseudo-drivers (*pseudo* because they do not actually control a hardware device), are front-end interfaces to the SCSI tape driver, **st**(7). Unrecognized user I/O requests received by the tvol pseudo-driver are passed down to the st driver, unaltered. One additional tape control operation is provided by the tvol pseudo-driver (see the "volume seek" description in section 3.6, Volume Set Support). The MTD pseudo-driver is actually a front-end to the tvol pseudo-driver. MTD provides an interface that allows user I/O requests to be dynamically limited by a privileged process.

When calling another driver to perform I/O for it, a pseudo-driver must know the driver to call and the device to use. tvold determines the mapping between driver devices, and both the mtd and tvol pseudo-drivers support special I/O control operations that are used to establish the device mapping to use for I/O operations that are passed on to another driver.

The SCSI tape driver, **st**(7), has been enhanced on Cray Solaris to provide a new control mechanism which detects tape insert and eject events. The tvold daemon creates one thread for each tape transport device being managed. These threads remain blocked until insert or eject events take place. When one of these threads detects media insertion, tvold calls, in turn, each of the media label recognizing modules until one of them identifies the volume. With the volume identity, tvold can look up its assigned mapping between the tvol pseudo-device and the st device that correspond to the drive containing the volume. This mapping is passed down to the tvol driver when a user tries to open one of the pseudo-devices for the tape volumes in the tvol filesystem.

3.5 Label Processing Support

Cray TMS is designed to support processing of both IBM and ANSI standard labeled tapes. Labeled tapes were designed to promote information interchange between foreign computer systems by specifying volume and file structures, recorded

labels for identifying volumes and files, and basic characteristics of the blocks containing the records constituting the file.

The information from the label records is augmented by (or overridden by) information maintained in the tape catalog. Since this information determines who can read and write the user data, it is considered to be privileged and should not be accessible by an unprivileged user.

Enforcing different levels of protection to different records on the same reel or cartridge of tape media requires an interface which allows the system to intercept user I/O requests at points where label records would be accessed, and perform the necessary I/Os to position the tape at the targeted user data. The architecture used by Cray TMS for this purpose is pictured in Figure 2.

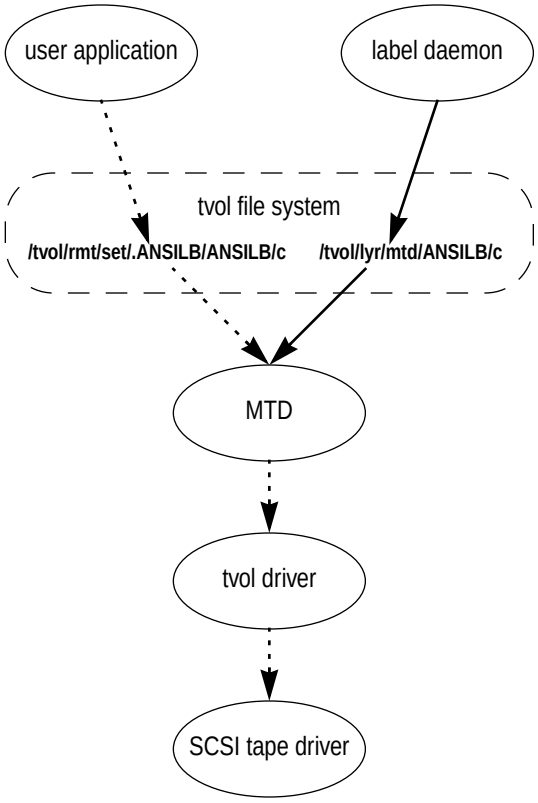

Figure 2: Label Processing

The figure shows the user application opening a volume device (/tvol/rmt/set/.ANSILB/ANSILB/c) which belongs to MTD. The label daemon is a privileged process which opens another MTD device (/tvol/lyr/mtd/ANSILB/c). MTD devices occur in pairs, with one device called the "control" device and the other, the "application" device. The application device interface accepts all of the tape I/O commands (**read**(2), **write**(2), and **mtio**(7)), but the behavior of those commands can be altered from the paired control device. The control device interface is a superset of the application device I/O repertoire. Additional **ioctl**(2) functions are provided to control the I/Os on the paired application device.

The label daemon indirectly interfaces with volume management. The volume management configuration allows specification of a program to run in response to events that happen to volumes. Among those events are:

- The volume has been inserted into a drive.
- The volume has been ejected from a drive.

The volume management configuration for Cray TMS specifies that a program is to be launched when a labeled tape volume is inserted. The purpose of this program is to provide the label daemon with the information required to perform label processing for the volume. This information includes the tvol file system path of the control device which the label daemon must open. Similarly, the eject event configuration specifies that when a labeled tape is ejected a program is launched to contact the label daemon with information that causes the control device to be closed.

Although the MTD interfaces allow the label daemon to perform the label processing functions required by IBM standard or ANSI standard labels, MTD was designed as a general interface which knows nothing about tape label formats or the processing requirements thereof. Conceivably, a third-party tape application which supported a proprietary labeled tape format could use Cray TMS for application tape management by:

- writing a media label recognizer module for the proprietary tape labels,
- writing a proprietary label daemon which uses the MTD interfaces to perform the required label processing for the proprietary label format, and
- configuring volume management to launch a program to contact the proprietary label daemon at events such as insertion or ejection of a tape with proprietary labels.

The MTD control device interface includes a control **ioctl**(2) with a data structure that specifies which user I/Os are of interest. This specification allows I/Os to be described not only in terms of operation (*e.g.*, **open**(2), **read**(2), **write**(2), **mtio**(7), **close**(2)), but also in terms of context (*i.e.,* detect the I/O before it is issued, detect it after it completes, or detect it only after it terminates with an error). MTD monitors the I/Os on the paired application device until it detects an instance of one of the interesting I/Os. At this point, the control data structure is updated with information describing the intercepted I/O and parameters and the **ioctl**(2) returns. The controlling process is then able to examine the updated data structure and take whatever action is dictated, including issuing any number of I/Os via the control device to the user's media. Once the indicated processing is complete, the fate of the user I/O (*i.e.,* continue, retry, or abort) must be determined by setting values in the control data structure. Also, the controlling process may wish to establish a different set of interesting I/Os in the control data structure before issuing the control **ioctl**(2) once more.

In order to effectively use the MTD interfaces to control access for every labeled tape on a Cray TMS drive, the label daemon has a multi-threaded architecture, where one thread is

spawned to supervise each labeled tape device, and a separate thread reads and services label daemon requests from the communication **socket**(3N). At any given time the vast majority of threads are blocked, waiting for the MTD control **ioctl**(2) to return. Service for any individual user is not impacted by the processing state of other users.

The following serves as a simplified example of how the label daemon uses the MTD control **ioctl**(2) to monitor a user's labeled tape access:

1. The label daemon communication thread receives a specification of the data the user wishes to access on the labeled tape from **tpmnt**(1) (a Cray TMS command with a user interface modeled after the UNICOS version).
2. The label daemon communication thread receives notification that a labeled tape has been inserted and spawns a device monitor thread for the application device.
3. The device monitor thread sets up a control data structure requesting detection of user **open**(2) termination and issues the control **ioctl**(2).
4. The **ioctl**(2) returns indicating that the user **open**(2) completed successfully and the **open**(2) parameters requested read/write access.
5. The label daemon uses another MTD **ioctl**(2) to identify the user. With the user identity, the label daemon looks to see if any data specification was provided by this user. With the data specification (or a default specification), the label daemon begins reading the label records on the tape until it finds the label records which identify the specified user data. It then examines the label records, and the CRL file records (if any) to determine if the user has the necessary permissions to read/write the specified data.
6. If the user does not have access permission, the label daemon will update the control data structure to request that the user **open**(2) be terminated with an error. If the user does have access permission, the label daemon will update the control data structure to request that it be notified in the event of a user **write**(2) initiation or **close**(2) termination; user **read**(2) requests are allowed to pass without interception. It also indicates that the intercepted user **open**(2) should be allowed to continue to normal completion, and issues another control **ioctl**(2).
7. If a user **write**(2) is detected before **close**(2), the label daemon will remove **write**(2) from the set of interesting I/Os and reissue the control **ioctl**(2). Upon detection of **close**(2), the label daemon will know to write the necessary label records to terminate the new user data because it had detected the fact that the user was writing data.

3.6 Volume Set Support

In Solaris 2.*x* volume management, each volume is an independent entity. Since IBM standard and ANSI standard label tape formats support multi-volume tapes (*i.e.*, an ordered set of related volumes), it was necessary to introduce support for CRL volume sets into volume management.

By supporting only CRL volume sets, the difficult problem of designing a generic volume management interface which allowed for the creation of multi-volume objects was side-stepped. CRL provides an interface for creating a volume set, and once the volume set has been created, it is accessible (when named) from the tvol file system.

To lessen the impact of volume sets on the volume management design, the devices for accessing the volume set were assigned, as usual, to the constituent volumes. The wrinkle with volume set access is that selecting a volume device to open also determines the initial tape position (at the start of that volume) in the volume set.

The relationship between volume set and constituent volumes is analogous to that between tape volume and volume devices. A volume set consists of one or more tape volumes, but only one may be opened at any time. So, logic was added to the tvold daemon to deny any attempt to open a volume device in a volume set once one had already been opened.

The main technical interest in volume set support lies in the design to allow the opened volume device to access any volume in the set. The solution proved to be relatively easy given the layered driver architecture in Cray TMS.

Although, the tvol driver passes user I/Os through to the physical device driver, the tvold daemon provides the tvol driver with the identity of the physical device to use. This occurs by the following mechanism:

1. A user opens a volume device in the tvol file system.
2. The user open is directed to a specific logical device in the tvol driver, either directly (because a tvol device was opened), or indirectly (because an MTD device was opened and the label daemon instructed MTD to direct I/O to the tvol driver device).
3. The tvol driver checks to see if it has received a mapping to a physical device for the logical device. If so, it will invoke the mapped driver to perform subsequent user I/Os on the mapped device.
4. If, on the other hand, the tvol driver has no physical device mapping, it adds a "no map" event for the logical device to the queue of work for the tvold daemon.
5. Using an **ioctl**(2), the tvold daemon collects and processes tvol driver events in turn. When it gets a "no map" event, it looks up the volume object which has the logical device as one of its devices in the tvol file system. It checks to see if the volume is mounted on a drive. If not, actions are taken to get the volume mounted (*e.g.*, request an autoloader device to mount it, or send email to the system administrator) and the "no map" request is queued awaiting mount completion.
6. When a volume is mounted, it is first detected as a media insertion by the device interface module which provides access to the physical drive. Once detected, the media label recognizer modules are called in turn, until the volume is identi-

fied. At this point the tvold daemon knows the identity of the volume and on which physical drive it is mounted.

7. Once the volume with the "no map" device has been mounted, the tvold daemon is able to pass the physical device number back to the tvol driver by using a "map" **ioctl**(2). The tvol driver can then pass the user's subsequent I/O requests to the mapped physical device's driver.

Given that this logic was already in Solaris volume management, it was only necessary to introduce a new event which would cause the physical device mapping for a tvol driver device to be changed. The new event takes the form of a new "volume seek" **ioctl**(2) supported by the tvol driver for volume set devices. The **ioctl**(2) data includes a specification of the targeted volume position.

The processing of this **ioctl**(2) is similar to that (described above) which established the initial physical mapping for the volume set device. The differences are:

- This occurrence is reported to the tvold daemon as a "volume seek" event for the opened volume set tape device.
- This event provides the tvold daemon with the **ioctl**(2) data specification which is used to calculate the identity of the volume targeted by the "volume seek".
- Once the target volume is found to be mounted so that the physical device is known, the tvold daemon can respond to the "volume seek" event. A new tvol driver "remap" **ioctl**(2) is used to change the physical device mapping of the opened volume set device to point to the physical device of the target volume.

Once all of this processing is complete, the "volume seek" **ioctl**(2) returns and subsequent user I/O to the open volume device will be directed to the physical device containing the targeted volume in the volume set.

This method of redirecting a user-opened volume device to use a different physical drive without disturbing the user application should prove useful in the future as a foundation for implementing sophisticated error correction strategies. Physical device remapping allows for the option of substituting another drive in the hopes of retrieving data from a volume with unrecoverable data errors.

3.7 Cray TMS Command Interface

As previously described, the Cray TMS command set is based largely on the UNICOS Tape Subsystem command interface. The following table presents a synopsis of the command set provided by Cray TMS. This list does not include any of the commands provided by CRL.

Table 2. TMS Commands

Command	Description
rsv(1)	reserve tape drive resources
rls(1)	release reserved tape resources
tpfrls(1)	forcibly release tape drive reservations
tpmnt(1)	setup access to labeled tapes
tpdevlist(1)	display drives and device groups
tpstat(1)	display status of drive resources
tprst(1)	display reservation status
tplabel(1M)	apply labels to a tape volume
tplist(1)	display internal label records on tape
tvolcancel(1)	cancel request to load a volume

Table 2. TMS Commands

4 Significant Features

Cray TMS contains numerous operational features, many of which are unique to Cray TMS. Some of these are:

- The Default Reservation

Cray TMS attempts to simplify typical tape operations by providing a "free" tape reservation for all user jobs. The user is not required to make an explicit reservation with the "rsv" command, if a single tape device is required. In this instance, the "tpmnt" operation can begin a user tape session. Also, the reservation is automatically released when access to the tape is closed.

- Bypass Label Processing

Privileged users may request labeled tapes be accessed such that the entire contents of the tape are accessible, including the header and trailer label records.

- Dynamic Tape Drive Reconfiguration

Tape devices can be brought under Cray TMS management, or removed from Cray TMS management during runtime by modifying the tvold configuration file, **tvold.conf**(4), and then signalling the tvold daemon. Also, if runtime errors are detected on a tape device, that device will automatically be removed from TMS control, or "deconfigured".

- Operator-less

Unlike mainframes and supercomputers, the CS6400 is designed to be an operator-less system. There is no operator message facility on Solaris as exists on UNICOS, in which the operator can assign user jobs to particular tape devices. Due to this fact, Cray TMS maintains the notion of *identifiable* and *unidentifiable* volumes. An unidentifiable volume is a non-labeled tape on a manually loaded tape drive. All other tapes are identifiable. The only method of access for an unidentifiable tapes on Cray TMS is by first mounting the volume, then using the physical pathname to the volume, in the tvol file system.

- CRL is Required

Unlike the UNICOS Tape Subsystem, CRL is a required element for the proper operation of Cray TMS.

5 Futures

As Cray TMS nears release, and general availability, several product enhancements have been committed for the next release. These include:

- An enhanced unlabeled tape interface

The MTD driver will own all device files in the tvol file system, including unlabeled tape volume device files. **tpmnt**(1) can then be used for both labeled and unlabeled tape access. The labeld daemon will automatically apply or remove tape labels as required. The result of which is a more consistent user interface between labeled and labeled tape volumes.

- The “tpmnt -O <offset>” option will be supported

Users can specify “volume offsets” on which to begin processing multi-volume tape sets. This alleviates the need to scan past unnecessary tape volumes in order to locate the desired dataset.

- User EOV Processing

Users can optionally gain control of labeled tape volume transitions as each volume hits logical EOT. This allows optional trailer information to be written to each volume, and requires the user initiate multi-volume tape transitions using the volume-seek **ioctl**(2). (See **tvolio**(7) for more information regarding volume-seek operation.)

- Volume-set Concurrency

Multiple user access to the same volume set will be permitted and, where possible, will be allowed to process in parallel.

6 References

[1] Stringer, Phil, “Tapes and Robots for the CS6400 SuperServer,” in *1995 Spring Proceedings*, Cray User Group (CUG) Inc., Denver, CO, 1995.

[2] *UNICOS Tape Subsystem User’s Guide*, Cray Research, Inc. Document Number SG-2051 7.0, November 1992.

[3] *MVS/370 Magnetic Tape Labels and File Structure Administration*, Release 1.2, IBM Publication Order Number: GC26-4064-2.

[4] *American National Standard for Information Systems - File Structure and Labeling of Magnetic Tapes for Information Interchange*, ANSI Standard Number: X3.27-1987.

[5] Alt, H., “Removable Media in Solaris,” in *Proceedings of the USENIX Winter Conference*, USENIX Association, San Diego, CA, January 1993.

[6] Alt, H., “CD-ROM’s and Floppies in Solaris”, *Proc. Jan. 1993 UK UNIX User Group/Sun UK User Group Conference*, January 1993.

[7] Dijkstra, E. W., “Cooperating sequential processes,” technological University Eindhoven, The Netherlands, 1965. (Reprinted in *Programming Languages*, F. Genuys, ed., Academic Press, New York, NY 1968.)