FUJITSU COMPUTERS SIEMENS

Reliant UNIX ONLINE Documentation

Reliant UNIX 5.44

Virtual Disks

RM200, RM300, RM400, RM600

Edition September 1997

Identification: U25634-J-Z915-1-7600

Copyright and Trademarks

1 Preface

Reliant® UNIX® supplies virtual disk users with a management concept for hard disks and memory disks. Compared to conventional approaches, this concept not only offers a richer functionality, but in some cases can provide significantly higher performance levels. The virtual disk concept is thus fully transparent to the user and to all of the applications. As a result, it is no longer necessary to adapt existing applications in order to utilize the benefits of virtual disks.

A virtual disk functions in the same way as a normal physical disk. A so called dummy device driver is inserted between the highest level of the Reliant UNIX logical I/O system and the physical device driver. This dummy device driver now maps all logical I/O requests on physical disks.

These days it is quite common for several disk drives to be installed in a single system. Typical MTBF times (MTBF = Mean Time Between Failure) for drives are currently about 200,000 hours (around 20 years). It is therefore reasonable to expect a drive failure every two months in systems with 100 drives, and every eight days in systems with 1,000 drives. Even assuming that the reliability of drives will increase in the future, this situation is far from acceptable.

This is precisely the problem that the various RAID techniques are designed to deal with. *stripe* and *mirror* type virtual disks (RAID0 and RAID1 respectively) in particular have been used to implement RAID concepts in the Reliant UNIX operating system. RAID (Redundant Array of Independent Disks) is a term used to refer to disk subsystems that increase the reliability, availability and performance of mass storage systems.

1.1 Summary of contents

This manual consists of two main sections:

- The first section contains an in-depth description of all types of virtual disk currently available. In particular, it lists their advantages and ideal application areas.
- The second section provides a detailed explanation of how to configure virtual disks, and includes a special section on root disk mirroring.

1.2 Notational conventions

The following notational conventions are used in this manual:

Italics	Names of files, programs, commands, variables, options and screen items such as input fields, text fields, menus, etc. in the body text
Typewriter text	System output, such as error messages, notifications, information and file extracts
Bold typewriter text	User input in examples
"Quotes"	References to other chapters of manuals
	User activities
	Additional information and tips
	Warnings

1.3 Target group

This manual is intended for system administrators who want to use the many advantages and opportunities offered by virtual disks in their system. It describes the various types of virtual disks and how to configure them.

2 Types of virtual disk

Various types of virtual disk are available for different application areas. The spectrum ranges from *simple* virtual disks which are similar to partitions, through mirror disks which improve the security and availability of disk systems, right up to memory disks.

2.1 The structure of virtual disks

Virtual disks represent a concept for data medium management. This concept makes handling data media much more convenient and flexible. Virtual disks operate with complete transparency for end-users and programs. Once installed, they offer all the input/output interfaces also provided by physical disks. So-called dummy device drivers are inserted between the highest level of the Reliant UNIX logical input/output system and the physical device drivers. These dummy device drivers map all input/output requests on the underlying physical disks.

With the exception of memory disks, virtual disks always have a number of parts. They can have one, two or more parts, each of which can be a physical disk, part of a physical disk or even another virtual disk.

2.2 Simple virtual disks

Simple virtual disks define either an area within a physical disk partition or an entire partition. A partition can be further subdivided by specifying the offset and length of a virtual disk of this type. If the offset and length specifications are omitted, then the whole partition is defined as a virtual disk. *Simple* virtual disks enable a physical disk, for example, to be divided into more pieces than are possible with partitions (up to16 partitions).

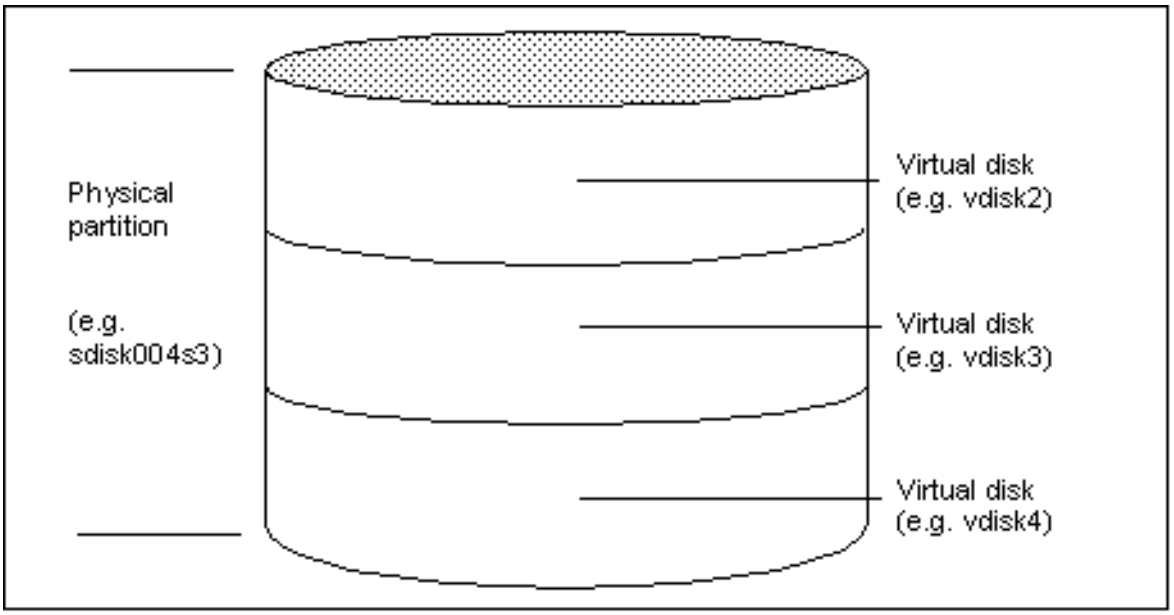

Figure 1: Dividing a physical partition into several simple type virtual disks

Since the partition table need not be modified on system level when creating virtual disks , it makes sense to subdivide physical disks into *simple* virtual disks. In the example in Figure 0, reference is made only to partition *s3* in the system kernel. Users and application programs, however, can access three separate virtual disks (*vdisk2*, *vdisk3* and *vdisk4*).

2.3 Concatenated virtual disks

Concatenated virtual disks consist of two or more pieces on one or more disk drives. They correspond to the sum of their parts. Unlike *simple* virtual disks where the disk is subdivided into small pieces, the individual disks or partitions are combined to form a single large logical disk.

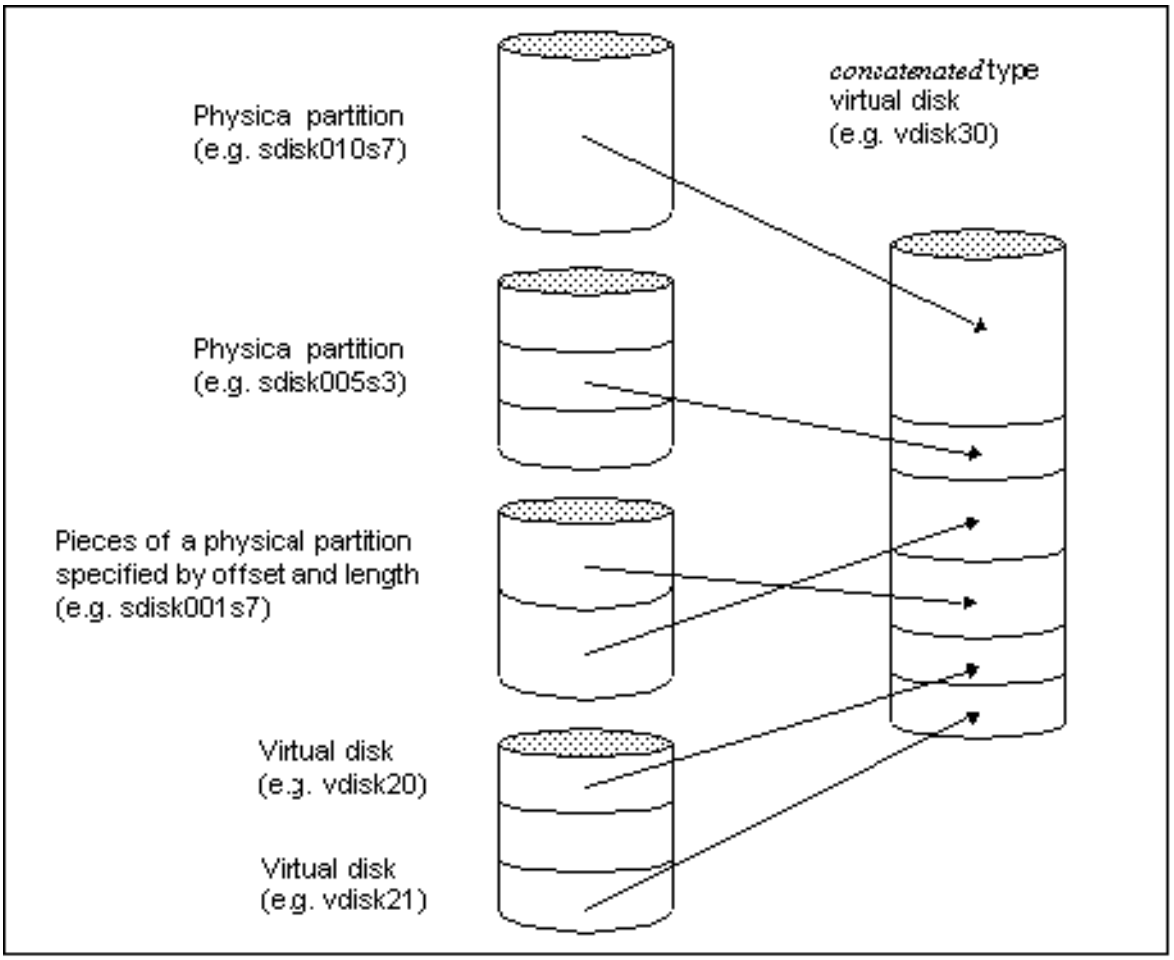

Figure 2: Composition of a concatenated type virtual disk

The use of *concatenated* virtual disks mean, for example, that file systems of up to 4 Gbytes *(ufs),* and file systems of up to 16 TByte *(vxfs)* in size can be set up, even if only 2 Gbyte disk are used in the system. To be precise the file systems size must be reduced by 16 KBytes (which is the page size).

A concatenated virtual disk can consist of partitions of a physical disk, pieces of a partition, or further virtual disks. Disk organization is thus extremely flexible. All input/output operations on a virtual disk correspond exactly to the input/output operations on a physical disk whose capacity equals the sum of the parts of the virtual disk. The pieces of a *concatenated* disk must be less than □
4 Gbytes in size because there are at least two parts. The pieces of a *concatenated* disk can be arranged as desired. By specifying the offset and length when defining the virtual disk, physical disks or partitions can be further subdivided.

2.4 Striped virtual disks (RAID0)

Under certain conditions, *striped* virtual disks offer the advantage of a much higher data throughput rate for input/output operations than normal disks. For instance, if application programs execute a large number of input/output operations on a virtual disk, these operations can be distributed evenly over several disk drives and/or controllers.

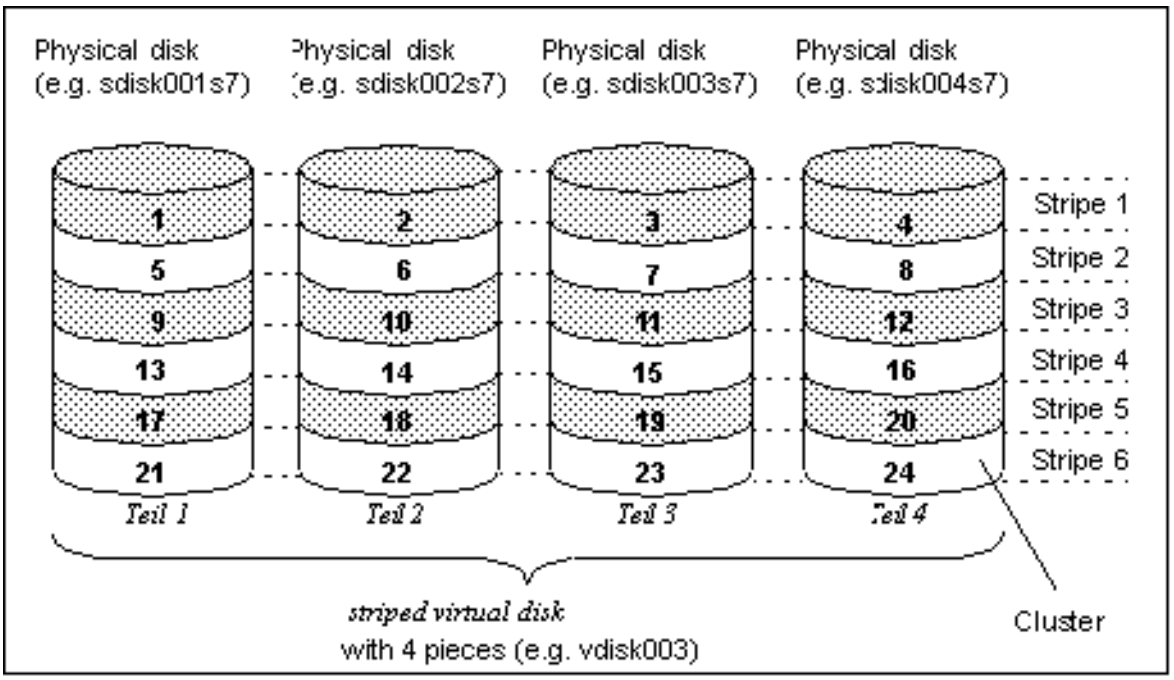

Figure 3: Structure of a striped virtual disk

Striped virtual disks (or *striped vdisk*) consist of two or more pieces. These can be physical partitions or further virtual disks (typically a mirror disk). The offset and length can be specified for each piece of the *striped vdisk*. For performance r easons, all pieces should be located on different physical disks or even on different SCSI channels, if possible. With *striped* virtual disks, the block numbers of the virtual disk are converted to those of the physical disks. Thus, sequential input/output operations on the virtual disk can be converted to input/output operations on two or more physical disks. This corresponds to RAID Level 0 (RAID0).

Each piece (in figure 3, each physical disk) of a *striped vdisk* is subdivided into clusters. In figure 3, for example, each of the physical disks is subdivided into six clusters. The conversion algorithm then processes the clusters in sequence; it begins by accessing the first cluster of the first piece (cluster *1* of physical disk *sdisk001s7* in the example), then the first cluster of the second piece (cluster *2* of disk *sdisk002s7* in the example), and so on until all first clusters have been allocated. Access to the second cluster of each piece of the virtual disk then begins (all clusters assigned to *stripe 2*; in this case *5*, *6*, *7*, and *8*).

All pieces must be the same size, and this must be a multiple of the cluster size.

A few points need to be taken into account if you wish to achieve the maximum data throughput rate:

- The most important dimension for a *striped* virtual disk is the cluster size. The cluster size should be selected to correspond with the applications that mainly access this virtual disk.
- If you select a cluster size that is too big so that a major part of the input/output operations always access the same cluster, then the actual advantage of a *striped vdisk* is wasted.
- You should avoid setting the cluster size so small as to increase the administrative work on the virtual disk driver.
- To take particular advantage of a *striped* virtual disk, you should run several jobs simultaneously. You can distribute input/output jobs among several physical disks by selecting a suitable cluster size. The cluster size should then be at least as large as the largest block size used in the application that accesses this disk.

For example, if an application uses blocks of up to 16 Kbytes in size, then set the cluster size of the *striped* virtual disk to *n* x 16 Kbyte, where *n* is an integer.


However, applications that create large, sequential input/output jobs are an exception to this. A single

read or write process (e.g. physical data backup or restore) with an input/output size of 64 Kbytes achieves the best performance if 4 disks are set up on 4 different ports with a cluster size of 16 Kbytes. Four jobs of 16 Kbytes each are then started are are run simultaneously (multi threaded striping). This increases throughput considerably .

You should ascertain the optimum partitioning for your system at an early stage by means of a test phase.Cluster sizes of 16 Kbytes and 32 Kbytes have proven to be particularly effective.

2.5 Memory/vmemory virtual disks

The *memory* and *vmemory* virtual disks (memory disks) are the only type of virtual disks that are not based in physical disks, but which instead use part of the main memory under Reliant UNIX as a data medium. Memory disks offer the same external input/output interfaces as all other virtual and physical disks. The memory disk simply acts as a fast virtual disk for an application.

There are two types of memory disk: *memory* and *vmemory*. In the case of *memory* type disks, the required memory is immediately reserved for the virtual disk, so that it cannot be put to further use after this. In other words, it is lost as free main memory, irrespective of whether or not anything is ever written on this memory disk.

vmemory type virtual disks function differently. In this case the memory area required to configure the memory disk is only reserved logically, but not physically. The memory is only occupied step-by-step as this disk is accessed, thereby blocking it for other uses. This means that only as much system memory space is used as is actually required for the memory disk. The value specified during configuration is therefore just an upper limit that cannot be exceeded even by the *vmemory* virtual disk. However, this concept also entails disadvantages. If a lot of the system memory is in use, then it may be that there is no longer enough memory to expand the memory disk. In this case, Reliant UNIX will also use the swap area as the memory disk, i.e. a hard disk for swapping out data. This means that it is possible to enlarge the memory disk at any time. The benefits of fast access times and higher transfer rates are lost in this case.

There is no doubt but that the greatest benefits of the memory disk lie in the extremely short access times and the high transfer rates, which are unchallenged by any other data medium. Memory disks are also equally suited to block-oriented and character-oriented data transfer.

Like all data stored in the main memory, the contents of a memory disk is transient. All information stored on memory disks is lost in the event of a controlled or uncontrolled system shutdown. For this reason, memory disks are particularly suited to the short-term storage of temporary data that needs to be accessed very quickly.

The size of the memory disk is simply specified in 512 byte blocks for the purpose of configuring the memory disk.

Thus, for example, the declaration line

/dev/vd/vdisk4 memory 2000

in the */etc/dktab* file requests 1000 Kbytes (2000 x 512 bytes) of system memory for memory disk *vdisk4* (see also "Configuring virtual disks"). No other configuration data is required for creating memory disks. If you wish to use a memory disk as part of a mirror disk, you should ascertain the number of blocks required from the *dkpart -lb* command output. Specify the name of the physical disk the memory disk is to mirror as the device name (see ...).

You should only set up memory disks if you really need to, and can use the short access times and high transfer rates. Because the system memory is an expensive and important system resource, large memory disks are only recommended in exceptional cases. For example, an RM system with 128 Mbyte main memory could be configured with a 16 or 32 Mbyte memory disk. A 100 Mbyte memory disk would take too much main memory from this system, thereby considerably reducing the system's performance.

2.6 Mirror virtual disks (RAID1)

With *mirror* virtual disks, all output operations are performed simultaneously on two or more physical devices.

In addition to enhancing disk performance, mirror disks offer the advantage of significantly increasing the security and availability of systems. The original data is stored in one or more identical copies. It is impossible to distinguish the originals from the copies. With mirror disks, therefore, we refer only to the pieces of a mirror disk and not to master, source, or copy.

If a piece of the mirror disk fails, the system can continue operating reliably and without interruption since the data and copies are stored on different disks. If the originals and the copies are stored on disks on different controllers, users can continue working even if one of these controllers fails.

RAIDLevel 1 is implemented in the operating system in the case of *mirror* virtual disks. Like all other types of virtual disk, mirror disks are fully transparent to the user and to all of the applications.

2.6.1 Structure of mirror disks

A mirror disk can comprise two or more pieces, but in practice generally has a maximum of three pieces. It can consist of physical partitions or further virtual disks. All the pieces of a mirror disk must be the same length. Unlike *concatenated* and *striped* virtual disks where the size of the virtual disk corresponds to the sum of its pieces, the size of the entire mirror disk is equal to the length of one of its pieces.

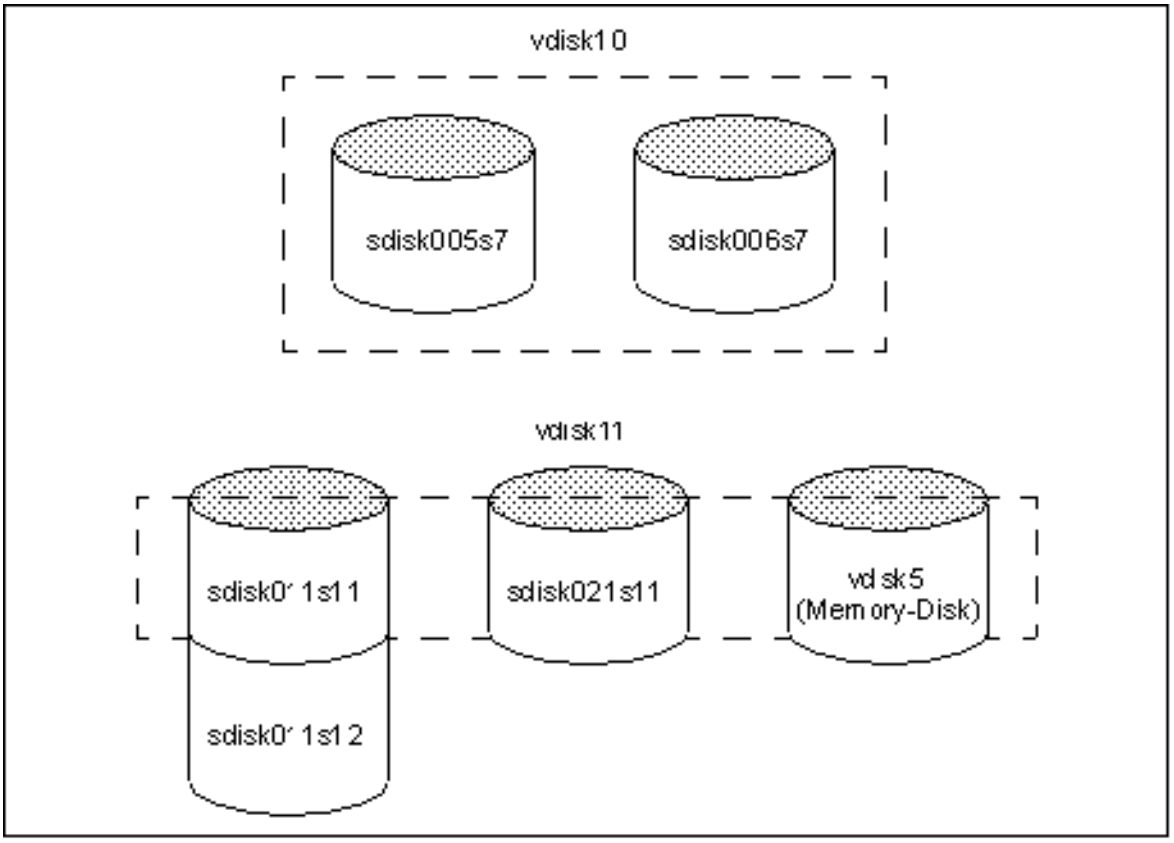

Figure 4: Structure of mirror disks

2.6.2 Functionality of mirror disks

Write access to the mirror disk implies write access to each of its pieces. This ensures that each piece contains identical data. Since users have write access to all pieces, write operations are slightly slower than with physical disks. This is immaterial, however, if the pieces are located on different disks on different controllers, as write operations are performed in parallel.

Conversely, read access to the mirror disk implies read access to only one of its pieces. Since the data contained in each piece is identical, the read operation can be performed in any piece. In practice, therefore, the piece that can provide the fastest read access is used. This is determined by the number of pending input/output operations for the underlying physical disks. The disk with the least number of operations in its

queue is assigned the read operation. If all queues are equally long, read access is alternate.

Exceptions:

- Sequential reading:
  Since the read cache of the disk can be exhausted in this case, only one piece of the mirror disk is read.
- Memory disk as part of a mirror disk:
  A memory disk can be read faster than a physical disk. All read operations are therefore performed on the memory disk.

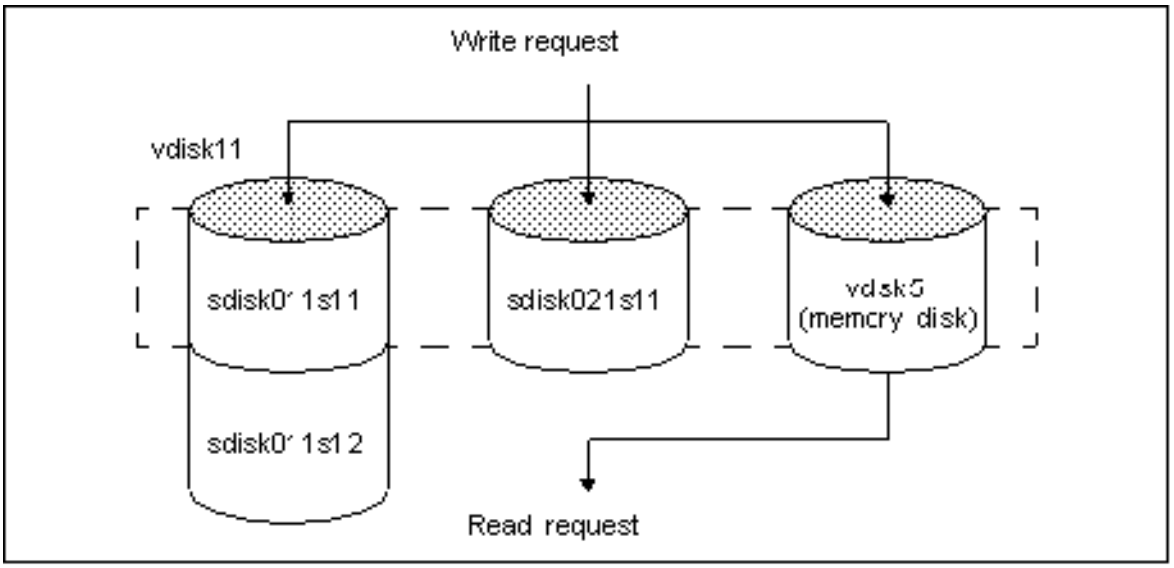

Figure 5: Writing to and reading from a mirror disk

Using these mechanisms, read access to mirror disks is generally faster than with physical disks. Particularly when using memory disks as part of a mirror disk, the speed of read operations is increased considerably. This links the benefits of both types of virtual disk. All data is available on this disk immediately after the system is powered up and the mirror disk is configured because it exists on at least one physical piece of the disk. This data is transferred to the memory disk by means of a catch-up process that **must** be started by a startup script (see Section "Catch-up process"). The catch-up process is completed relatively quickly due to the high speed with which the memory disk is accessed.

All further read accesses are only carried out on the memory disk after this process is terminated because this is certainly the fastest part of the memory disk. However, the speed of the memory disk cannot be exploited during write accesses because write accesses to mirror disks must be executed on all parts of the mirror disk to ensure data redundancy.

If a piece of the mirror disk is itself a virtual disk, it always returns a constant number when the length of the queue is queried. If all pieces are virtual disks, they all return the same constant number, and read access is alternate irrespective of the actual load on the underlying physical disks. This is one of the reasons why a mirror disk should consist of physical disks rather than further virtual disks (apart from memory disks).

2.6.3 Implementing mirror disks

If you wish to create more complex structures, e.g. mirrored *striped* virtual disks or mirror disks consisting of physical disks and memory disks, you should pay particular attention to the procedure in the event of errors. During configuration, for example, you can ensure that failure of one of the pieces of a mirror disk has a minimal effect on the security of the entire mirror disk.

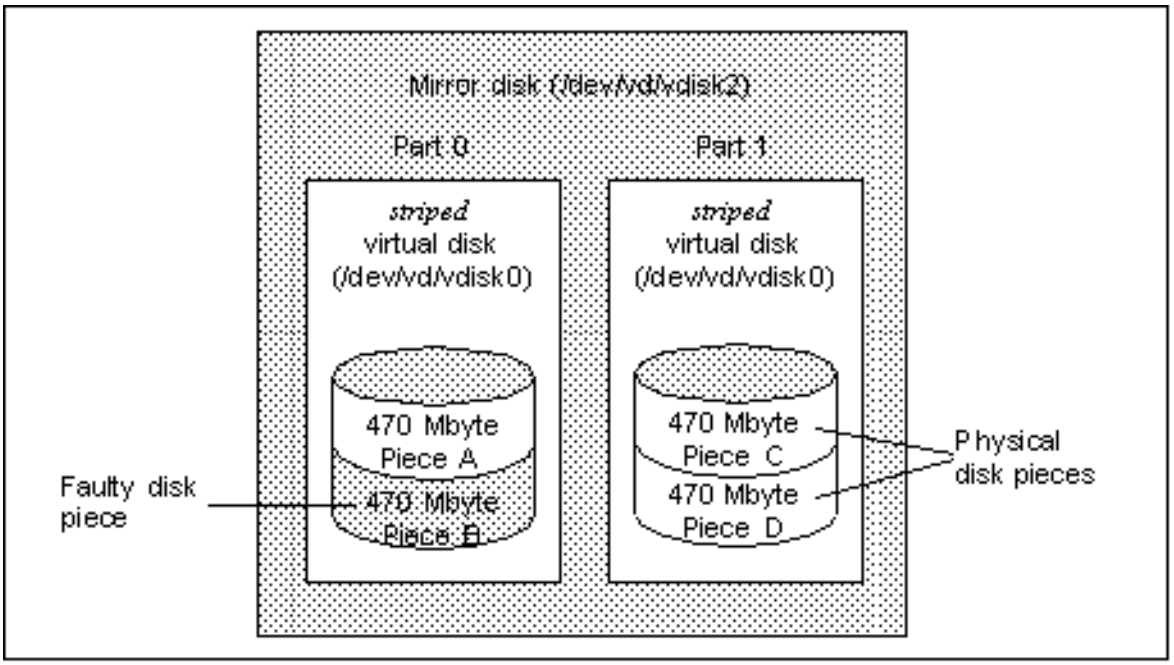

Figure 6: Faulty mirror disk design

The figure above shows a faulty version of the connection between a mirror disk and a *striped* virtual disk. If one of the pieces of the physical disk (e.g. piece B) fails, the entire piece 0 of the mirror disk is identified as *disabled* (see also section on "Mirror disk statuses"). Thus, even the intact piece A is disabled. The entire mirror disk is now in *NOT-MIRRORED* status. If an error subsequently occurs in piece C or D, piece 1 of the mirror disk also switches to *disabled*. This mirror disk is then in *DOWN* status, and does not permit any further input/output operations.

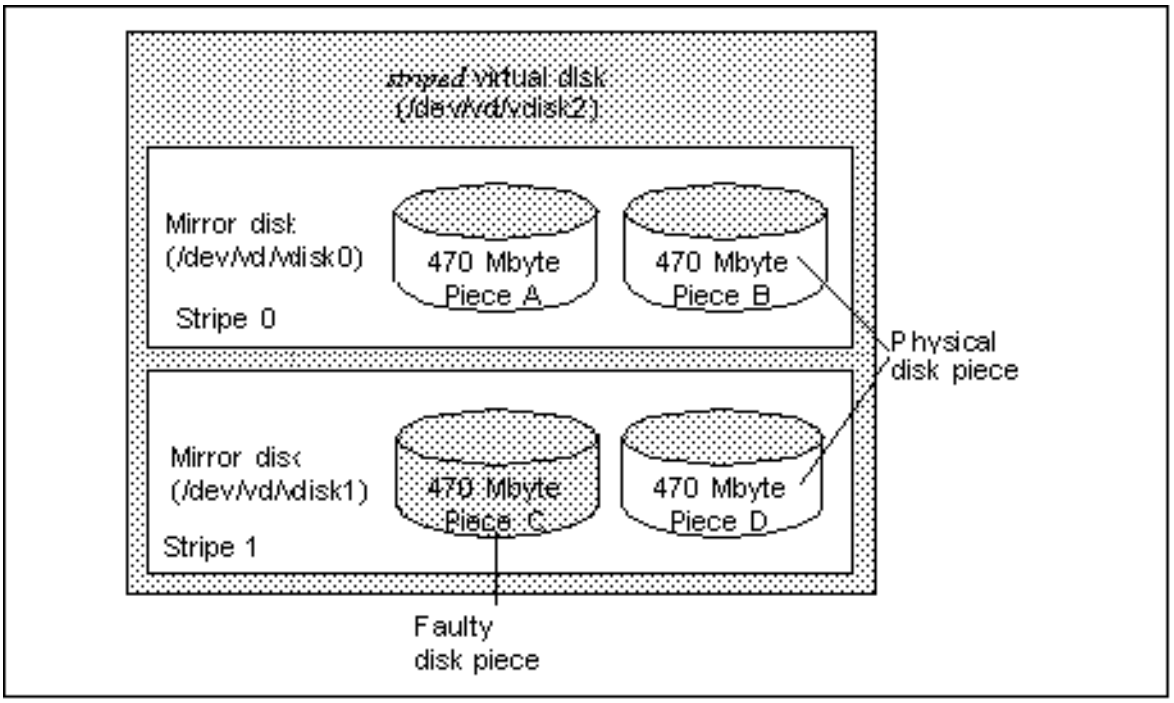

Figure 7: Correct design of a mirror disk

The disk design in Figure 7 is much better. Firstly, two mirror disks have been configured and linked to a *striped* virtual disk. RAID Level 0 and RAID Level 1 are thus implemented.

This modified design has no effect on the capacity or speed of the virtual disk. The security and availability of the disk, however, are considerably increased. If a piece of the disk (e.g. piece C) fails, only mirror disk *vdisk1* is switched to *NOT-MIRRORED* status. The *striped* virtual disk is still fully functional. Even if an error subsequently occurs in piece A or B, the user process does not detect this. It is not until both pieces of a mirror disk fail that this piece of the *striped* virtual disk becomes inaccessible.

The design in Figure 7 also allows a mirror disk to be restored easier and faster than the design in Figure 6. Once the error has been rectified, a catch-up process must be executed in both cases. In the first example, 940 Mbytes (2 x 470 Mbytes) must be copied, as the entire piece 0 of the mirror disk was disabled. In the second example, only 470 Mbytes need be copied, as the catch-up process only runs in mirror disk *vdisk1*.

2.6.4 Mirror disk statuses

Normally, all the pieces of a mirror disk contain exactly the same data. If a piece fails for some reason, you can assume that it is defective. Since a valid copy of the mirror disk data is no longer contained in this piece, the piece cannot be used. If a piece fails, access is restricted to the mirror disk data located in the other error-free pieces. As long as one error-free piece exists, the mirror disk remains fully functional.

To ensure that data integrity can be checked in a piece at any time, each piece of the mirror disk is assigned a status. This distinguishes the mirror disk from all the other types of virtual or physical disk.

A piece of a mirror disk can be in one of four statuses: *online*, *disabled*, *enabled*, or *Master*.

online
: This is the normal status of a piece of a mirror disk. An *online* piece contains a complete valid copy of the mirror disk data, and can be used for both read and write operations. It remains in *online* status until input/output errors occur in this piece, or until the system administrator uses a command to switch it to another status. Each write operation on the mirror disk is performed on all *online* pieces. Read operations, however, are performed on only one *online* piece.

disabled
: If an unrecoverable input/output error occurs in an *online* piece, the piece is automatically switched to *disabled*. Further input/output operations are not performed on a disabled piece, since the integrity of the data in this piece can no longer be guaranteed. The piece remains in *disabled* status until the system administrator changes its status manually. This normally occurs after the cause of the error has been determined and rectified. During initial configuration, all the pieces of the mirror disk are assigned *disabled* status by default, as they do not contain any valid copies of data immediately after configuration.

enabled
: If a *disabled* piece of a mirror disk is to be reactivated, it is assigned *enabled* status. An *enabled* piece can be selected by the user, but is not in *online* status until a new copy of the mirror disk data is created in it. The copy procedure is performed automatically by activating a system process that reads the data from *online* pieces and copies it into the enabled pieces. This copy procedure is possible only if the user is simultaneously running an input/output operation on the mirror disk. Once the copy procedure is complete, the enabled piece is reset to *online* as it is now guaranteed to contain an identical copy of the mirror disk data. All write processes that access the mirror disk are performed on all *online* pieces and all *enabled* pieces. However, only *online* pieces can be read.

Master
: If a mirror disk was set to *DOWN* status by an error or by the system administrator, the *-m* option of the *dkmirror*(8) command can be used to define the piece of the mirror disk which is to be used as the master piece the next time this mirror disk is configured. This piece is then set to *online* and acts as the source for a catch-up process. Otherwise, this status corresponds to the *enabled* status.

Similarly, the virtual mirror disk as a whole also has a status. Depending on the status of its pieces, a mirror disk can have one of the following three statuses:

MIRRORED
: A mirror disk is identified as *MIRRORED* if at least two of its pieces are *online*. In this case, the user is safeguarded against errors in individual pieces of the mirror disk.

NOT-MIRRORED
: A mirror disk is identified as *NOT-MIRRORED* if only one of its pieces is *online*. The mirror disk can still be used, but does not provide data redundancy.

DOWN
: A mirror disk is identified as *DOWN* if none of its pieces are *online*. No further input/output operations can be performed on a mirror disk in *DOWN* status.

The status can be switched automatically or manually. The mirror disk driver can change the status of a piece automatically if an input/output error occurs in this piece (from *online* to *disabled*), or when data has been copied in full (from *enabled* to *online*). The status of a piece can be changed manually by the system administrator using the *dkmirror*(8) command. This command can also be used to query the current status of a mirror disk or its pieces.

The status of each mirror disk (*MIRRORED*, *NOT-MIRRORED*, or *DOWN*) and all of its pieces (*enabled*, *disabled*, *online*, *or Master*) is stored on a *statesave* or *statsv* virtual disk (see Section "Statesave and statsv virtual disks"). This is known as a *statesave* device.

A *statesave* device is always located on one or more physical disks, so that the status information is nonvolatile. When the system is booted, therefore, it is restored to the same status as when it was shut down.

The information stored in the *statesave* device is modified when the status of a mirror disk or piece is switched (access error, end of a catch-up process (see Section "Catch-up process"), or manual change with *dkmirror*(8)).

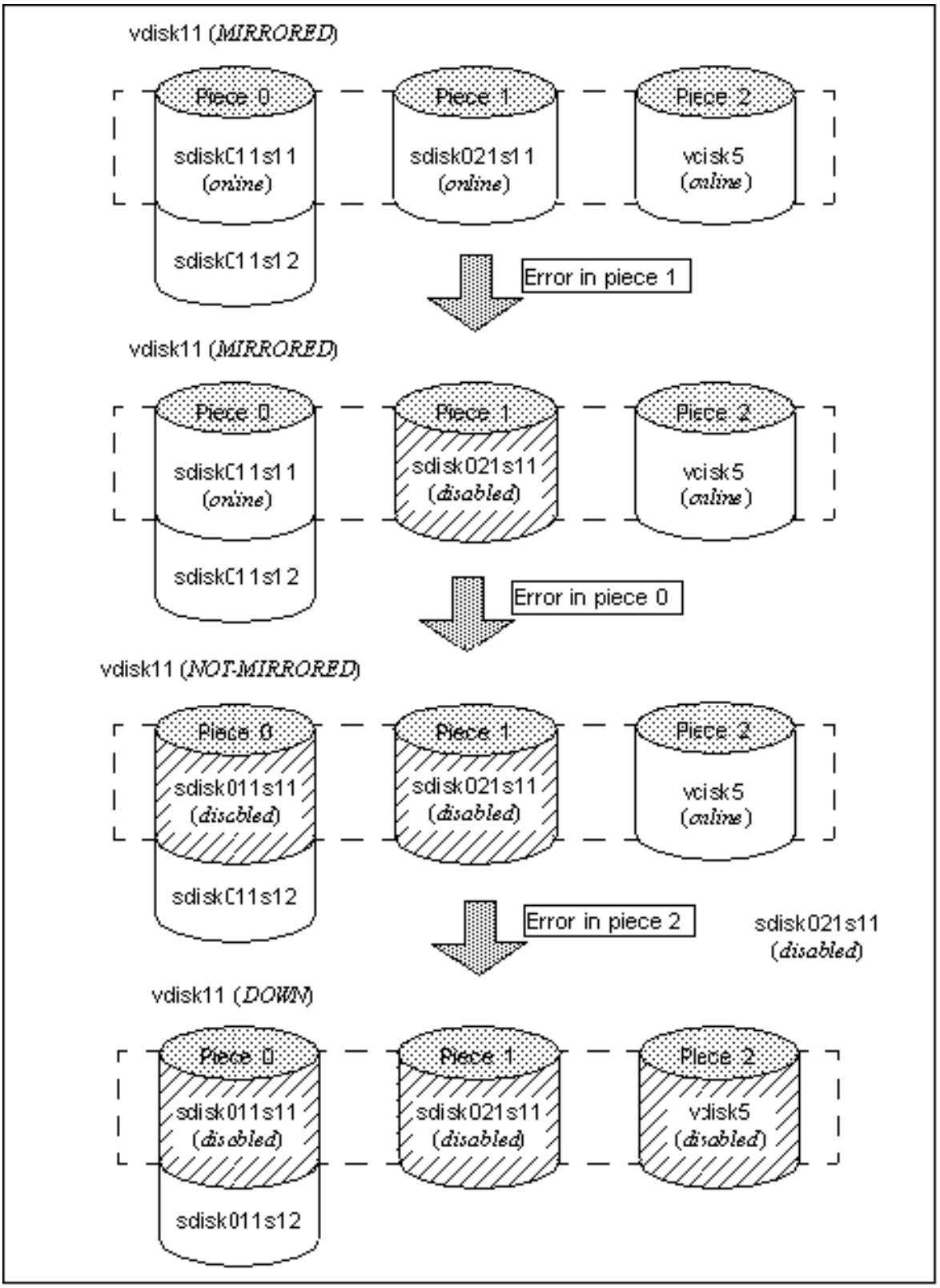

Figure 8: Behavior of a mirror disk in the event of errors in individual pieces

2.6.5 Catch-up process

One of the most important features of a mirror disk driver is its ability to synchronize the individual pieces of a mirror disk. This is necessary either at startup (immediately after configuration of the mirror disk), or after a piece has been set to *enabled*. The synchronization process involves copying the data from *online* pieces to *enabled* pieces. Once the copy procedure is complete, pieces that were previously *enabled* are switched to *online*. This copy procedure is known as the "catch-up process". During the catch-up process, the mirror disk can be accessed by other processes. Read operations are performed on a single *online* piece, while write operations are performed both on all *online* pieces and on all *enabled* pieces. This means that after the catch-up process, all pieces contain identical data. However, please note that disk performance may be restricted while the catch-up process is running. The priority with which a catch-up process occupies the disk resources can be set using the *-p* option of the *dkmirror*(1M) command.

A catch-up process is started automatically as soon as at least one *online* piece (the source of the copy procedure) and at least one *enabled* piece (the target of the copy procedure) exists. The process is initiated by the mirror disk driver. Once it is complete, the status of the target is changed from *enabled* to *online*.

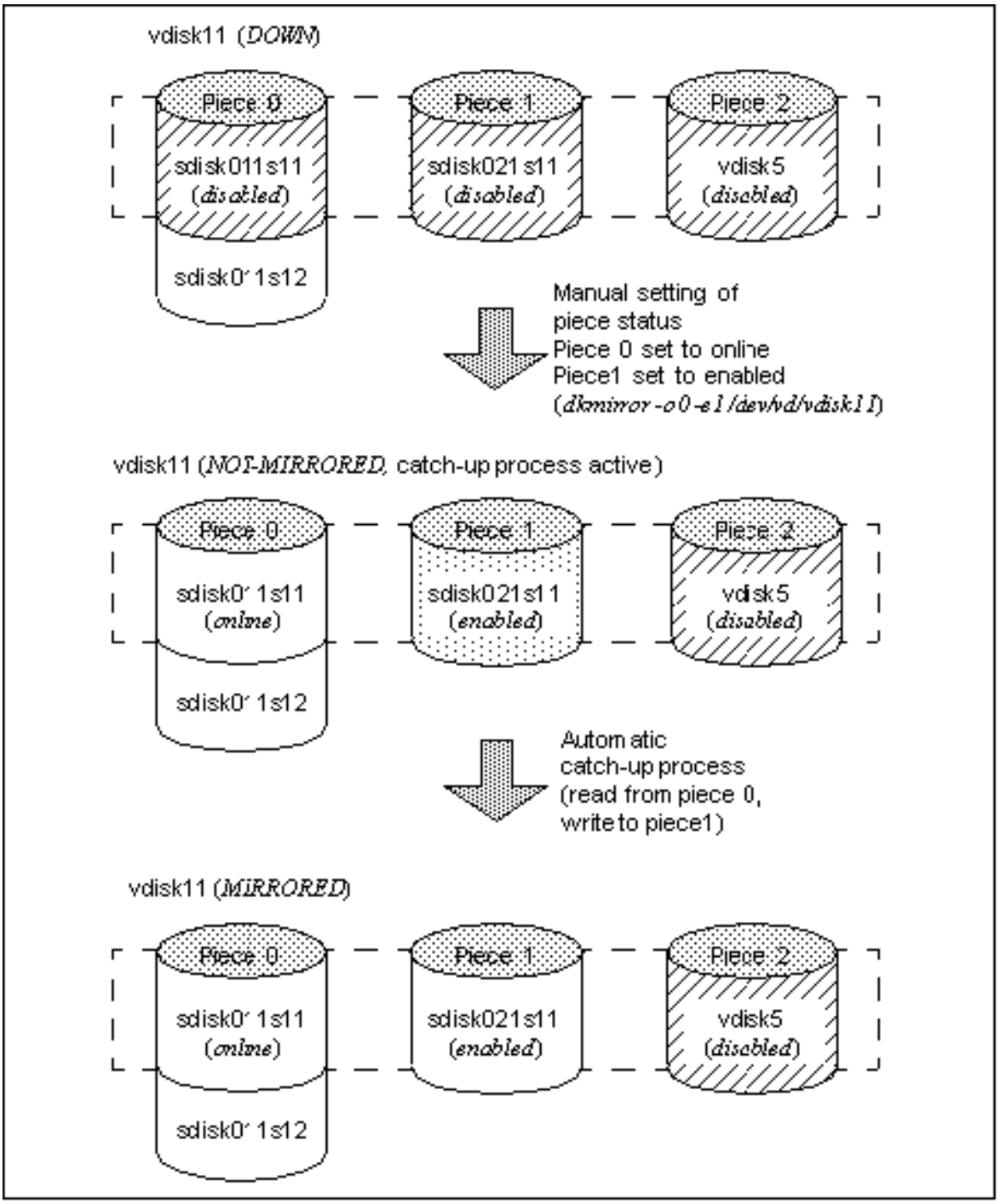

Figure 9: Behavior of a mirror disk when initializing the status of its pieces

Only one piece of each mirror disk can be manually set to *online*. The other pieces achieve *online* status by means of a catch-up process. If two pieces are manually set to *online*, there is a danger that the mirror disk could be inadvertently configured with different data in the various pieces.

The catch-up process executes its input/output operations without buffering. It runs asynchronous to user input/output and is completely separate from the other input/output operations performed on this or another mirror disk. The catch-up process communicates with a special system process, the *mirror* daemon. This daemon is displayed as *mr_daemon* following a *ps*(1) call.

The catch-up process terminates successfully if all *enabled* pieces are set to either *online* or (due to failed write operations) *disabled*. A catch-up process cannot be executed successfully if it cannot read an *online* piece or modify an *enabled* piece, which means that there are no further *online* pieces for read operations or *enabled* pieces for write operations.

During the catch-up process, all write operations are performed on all *enabled* pieces. If a write operation cannot be performed, the *enabled* piece is reset to *disabled*. With the catch-up process, read operations are performed on a single *online* piece. If a read operation cannot be performed, the catch-up process does not set the *online* piece to *disabled*. Instead, this area is assigned an internal identifier, indicating that it is unsuitable as a source for catch-up read operations. The read operation is then retried in another *online* piece. If there are no further *online* pieces, the catch-up process is terminated.

If an error occurs in an *online* piece during a user input/output operation, the respective piece is generally set to *disabled*. This is not the case for input/output errors during the catch-up process. Instead of setting the mirror disk to *DOWN* following a failed read operation in the catch-up process, the mirror disk is left in *NOT-MIRRORED* status and the catch-up process is aborted. The internal identifier of the *online* piece set in this case (indicating that this piece is not suitable as a catch-up source) can only be reset manually by the system administrator. This piece of the mirror disk must be reset to *online* using the *dkmirror -o* command.

2.6.6 Error recovery on mirror disks

If an error occurs on a piece of a mirror disk, this piece is automatically switched to *disabled*. No further input/output operations are performed on the piece until the system administrator resets it to *enabled*. This simple rule applies without restriction, thus guaranteeing the consistency of all data contained in all *online* pieces. No distinction is made between fatal errors on the disk drive and a simple non-recoverable ECC error (Error Correction Code) in an individual sector.

In practice, most *disabled* pieces can be reset to *enabled* following rectification of the error. **Header mismatch disk errors**

can be corrected by assigning alternate sectors. Most ECC errors can be eliminated by rewriting the relevant sector. Defective disk drives can frequently be reset successfully. Faults in the controller hardware, adjustment of the read/write head, and other drive errors are normally not the responsibility of the system administrator. With errors of this type, the mirror disk piece must remain disabled until it has been replaced.

In order to replace a disk, the system must be powered down for maintenance purposes. However, this does not apply to **high-availability configurations** in which drives can be replaced online (see also Section "Replacing a defective OLR disk").

Additional security can be provided if the mirror disk is configured with three pieces at the outset, where the third piece acts as a *disabled* reserve drive. This third *disabled* drive does not place any additional load on the system. Each logical write operation on the mirror disk is converted to only two physical operations. If one of the drives fails, the third drive becomes available. This can be switched to *enabled* without any problems. Once the automatic catch-up process is complete, the mirror disk is returned to *MIRRORED* status and effective data protection is restored.

This concept also simplifies the creation of backup copies. You can set the addition drive to *enabled*, thus starting the catch-up process. Once the catch-up process is complete, reset the drive to *disabled*. The additional drive is thus deactivated and contains an up-to-date complete copy of all data on the mirror disk. This data can now be backed up without affecting disk performance. If the additional disk has two connections to two systems, it can also be used to back up the other system.

2.6.7 Working with mirror disks

Following configuration and initialization, the effort involved in managing the mirror disk is minimal. When switching to multiuser mode, the system automatically calls to *dkconfig*(8) from the *brc* script. The status of the mirror disk is retained following a controlled (and generally an uncontrolled) system shutdown and reboot.

In the event of a system crash (e.g. due to a power failure), you must reinitialize the status of the mirror disk. The catch-up process must also be started manually with dkmirror . If the power should fail

on a system which is not equipped with a BBU (battery backup unit), this may give rise to inconsistencies in your mirror disks which the mirror disk driver fails to recognize. You should therefore proceed as follows:

After restarting the system, but before you start the applications or mount the file systems, check whether there is more than one piece of the mirror disk(s) with the state online. If there is, set the state of all but one of them to disabled, and then set all the pieces with state disabled to enabled, in order to start the catch-up processes which will equalize the data again. You may need to adjust the priority of the catch-up process first (see the section Using priorities to control the catch-up process on page ...).

The configuration of the mirror disk is determined from the */etc/dktab* file, and is therefore seldom lost. You can continue using the mirror disks until there are no further *online* pieces.

Please note that direct access to partitions that are configured as pieces of a mirror disk is not permitted under any circumstances. This is similar to a mounted file system in whose partition direct write operations are prohibited.

2.6.8 Messages of the mirror disk driver

The following messages (examples) are displayed on the system console by the mirror disk driver. These messages are output when the status of a piece is changed using *dkmirror*(8), when an input/output error is corrected, and when pieces are automatically set to *disabled* due to input/output errors.

Mirror Disk initialization complete.□
Mirror Disk shutdown complete.□
□
Mirror vdisk0: Piece 0 online.□
Mirror vdisk0: Piece 1 enabled.□
Mirror vdisk7: Piece 1 disabled.□
□
Mirror vdisk7: piece 2 Write error at block 784.□
Mirror vdisk7: piece 2 is now disabled.□
Mirror vdisk7: 2 online pieces left – this Mirror Disk is still MIRRORED.□
Mirror vdisk7: Recovered from the write error.□
□
Mirror vdisk7: piece 0 Read error at block 24.□
Mirror vdisk7: piece 0 is now disabled.□
Mirror vdisk7: 1 online piece left – this Mirror Disk is now NOT-MIRRORED.□
Mirror vdisk7: Retrying the read on another piece.

If a catch-up process is currently running, you are kept informed of the individual steps in the process. Here is an example of some typical messages:

Mirror vdisk7: CATCH-UP starting – from piece(s) 0 1 to piece 2.□
Mirror vdisk7: No more enabled pieces to catchup to.□
Mirror vdisk7: No more online pieces to catchup from.□
Mirror vdisk7: CATCH-UP aborted.□
Mirror vdisk7: CATCH-UP attemped – piece 2.□
Mirror vdisk7: CATCH-UP failed – piece 2 is NOT online.□
□
Mirror vdisk1: CATCH-UP starting – from piece 0 to piece 1.□
Mirror vdisk1: CATCH-UP succeeded – piece 1 is now online.□
Mirror vdisk1: 70.0 MB (8960 I/O 16k) 40.3 Sec (222 IO/sec,□
1778 kB/sec)

(with catch-up priority 3 (*dkmirror -p3*). With priority 2 the process would take 85.1 sec. for□
example (105 IO/sec, 842 kB/sec), and 5.6 mins. (26 IO/sec, 210 kB/sec) with priority 1)

The catch-up process outputs statistics in the last line. The figures in the example above indicate that the total

capacity of the mirror disk is 70 Mbytes. The catch-up process required 8,960 input/output operations, where 16 Kbytes (or 32 blocks) were read or written for each operation. The total duration of the catch-up process was 40.3. From this value, the average input/output transmission speed was calculated as 222 operations per second (or 1778 Kbytes per second).

2.6.9 Root disk mirroring

The root disk (also known as the system disk) is a particularly sensitive area. If the root disk fails, the entire system comes to a standstill. By mirroring the root disk, therefore, you can significantly increase the security and availability of the entire system. However, mirroring of the root partition is practical only if all the other pieces of this disk are also mirrored. This is simplified considerably by using two identical disks (same size and partitioning) for mirroring purposes.

With a mirrored root disk, additional management tasks must be performed when booting the system. The boot process in the case of a mirrored root disk is described below.

When the system is started, the boot software evaluates the */etc/default/boot* configuration file. The entries in the configuration file are transferred as parameters, and include information on whether or not the root disk is to be mirrored.

Reliant UNIX checks the transferred parameters and initializes root disk mirroring, if this is desired. One mirror part of the mirrored root disk (e.g. */dev/ios0/sdisk000s0*) is mounted for this purpose.

The configuration files for mirrored root disks *mrconfig.1* and *mrconfig.2* are read out (see Chapter "Configuring virtual disks"). The virtual disks for the root and swap pieces, including the associated *statesave* or *statsv* virtual disks, are configured. The just mounted file system can then be unmounted, and the file system of the mirrored root disk mounted in its place. All the options of normal mirror disks are now also available for the root disk.

Further information can be found in the Chapter "Configuring virtual disks".

When reinstalling the operating system, the system disk is overwritten. Information on mirroring is thus lost. Following reinstallation, therefore, you must reconfigure all virtual disks. If you have not yet reactivated root disk mirroring, an operable copy of the previous operating system will still be available on the other piece of the root mirror disk. You could therefore declare the pieces of this copy as (master pieces, and thus restart the old operating system.

Faulty disks can be replaced online in the case of RM400-Cxx and RM600-Exx systems with mirrored root disks (see Section "Replacing a defective OLR disk"). The same applies to RM300-Cxx systems where the root disk and root mirror disk are in auxiliary cabinet BG42.

2.7 Statesave and statsv virtual disks

Statesave devices are used exclusively for nonvolatile storage of the status of mirror disks. When configuring *statesave* devices, a distinction is made between *statesave* disks and *statsv* disks, although in principle they have the same format and function. During configuration, virtual disks can be divided into groups (see also "Configuring virtual disks"). *statesave* virtual disks are used to store the status of all mirror disks of a particular group, while *statsv* virtual disks perform the same function for non-grouped mirror disks. Only one *statesave* virtual disk is required for a group, and one *statsv* virtual disk for all non-grouped mirror disks.

Statesave devices can consist of one, two, or more pieces. They are organized in the same way as mirror disks, i.e. all of their pieces must be the same size. Since the data stored on a *statesave* device is extremely important for operating all mirror disks, you should configure the device with at least two pieces. Virtual disks of this type have a low memory requirement. A *statesave* device must be at least 8 Kbytes, but must not exceed 16 Kbytes.

The maximum number of entries of a *statsv* or *statesave* virtual disk is limited to 500. Here, all the pieces of all mirror disks of a particular group, as well as the mirror disk itself, count as one entry. If you require more

entries, you must create an additional *statesave* device (and a new group).

A *statesave* device must be configured before you can configure the first mirror disk. Only mirror disk drivers can access these disks. *statesave* and *statsv* virtual disks are not available to ordinary users.

As of SINIX V5.42, the partition table of each disk is configured such that partition 15 is reserved for *statesave* devices by default. Partition 15 is as small as possible and is not overlapped by any other partition.

Even if this partition can be used for other purposes, you should use partition 15 when using mirror disks as a *statesave* device.

Statesave devices can have one of the following statuses:

ONLINE
: Two or more pieces are online.

ENABLED
: Only one piece is enabled (the status must still be updated).

DOWN
: There are no *online* pieces. This *statesave* device cannot be used.

Other specifications:

(in use)
: The *statesave* device is currently being used to store the status of the configured mirror disk(s); any attempt to deconfigure this disk will fail.

Specifications on the pieces:

online
: The piece contains valid data.

disabled
: The piece cannot be used. Possible reasons are listed below:

  (I/O error)
  : An error occurred in an input/output operation.

  (open failed)
  : An attempt to enable the piece failed.

  (manual)
  : The piece was disabled manually using *dkmirror -d*.

enabled
: The piece is enabled and can be used. The next time the status is updated, status information will be written to this piece. If this write operation is successful, the piece is switched to *online*.

open
: The piece was opened successfully. The read/write operation is still to be executed. If the read operation is successful, the status is switched to *enabled*.

3 Configuring virtual disks

This section contains a detailed description of how to configure all types of virtual disk. It details the limit values and marginal conditions which must be taken into consideration when configuring and using virtual disks, as well as providing step-by-step instructions for configuring virtual disks. In principle, the configuration procedure is the same for all types of virtual disk. However, this does not apply for the configuration of mirrored root disks, which involves a few additional steps, the configuration of OLR mirror disks or mirror disks of an OLR group, the configuration of distributed mirror disks in a DLM/ OPS environment, and the configuration of mirror disks in an OBSERVE environment.

3.1 OLR Disks

The 3.5-inch disks of your RM system, which are integrated into special mounting frames, are online replaceable (OLR = Online Replacement). This is not the case with 5.25-inch disks.


Disk mirroring is required if data is not to be lost during OLR.

If the tiers for the SCSI devices (SCSI tiers) are not **"hot swappable"** (RM400-Cxx systems with a CS 35 or CS 19 controller and RM600-Exx as well as auxiliary cabinets BG70/BG71 or BG31, BG32 and BG42), then a disk can be replaced online on this channel, even if other disks are still active on the same channel. All activities on this disk must be terminated before replacement. In the case of "hot swappable" SCSI tiers , even non-mirrored disks can be replaced online, however, the data stored on the disk is no longer available in the system after replacement.

The disks must be mirrored if the SCSI tiers used **are not "hot swappable"** (in other words they are "w arm swappable"). If a disk on such a SCSI channel is replaced, then data may be corrupted on other disks on the same channel. For this reason, the **entire** channel must be brought to a standstill before a disk is replaced, i.e. **all** activities for **all** disks on this channel must be terminated. To ensure that the active applications are not affected by this, the data must be mirrored so that the applications can continue to operate on the remaining mirror. For this the virtual mirror disks must be divided among two or three symmetrical SCSI channels, each containing identical disks with identical partition tables. In addition, only disks may be connected to these SCSI channels (no other SCSI devices, such as a CD-ROM drive, floppy disk drive, etc.). The OLR group concept was introduced to ensure that all requirements are met in this case (see also Section "Configuring OLR mirror disks").

Replacing a faulty disk requires measures that are introduced by means of the *Configuration* menu item on the *SYSADM* system administrator's user interface or by means of the *XCONFIG*graphical user interface (see also Section "Replacing a defective OLR disk").

Please refer to the operating instructions for your RM system to find out whether or not your system supports the OLR function for disks.

3.2 DLM/OPS

A DLM cluster ((DLM = Distributed Lock Manager) allows for a larger number of transactions and speeds access to an Oracle database. It consists of up to four RM systems that share a common Oracle database. This product is frequently referred to as Oracle Parallel Server (OPS).

3.3 OBSERVE

OBSERVE can be used for vital-sign monitoring of one or more systems. Functions and processes are monitored by so-called observers. If one of the monitored systems fails, another system can take over all the tasks of the faulty system. Using software-driven changeover switches, it can also take control of the disks of the faulty system, provided they are located in an external peripherals cabinet.

Additional information can be found in the manual pages on *dktab*(4), *dkconfig*(8), *dkmirror*(8), *vdisk*(7), *mirror*(7), *mroot*(7), and *olr*(7).

3.4 Limit values and marginal conditions

Certain marginal conditions and limit values must be observed when configur ing virtual disks. These refer in particular to the number and size of the pieces of a virtual disk, and the connection of virtual disks.

- Theoretically, a virtual disk can have a maximum size of 2^{64} Bytes = 16 Ebytes. However, the sizes of the file systems are subject to the following restrictions:
  - (4 Gbytes - 16 Kbytes) (*ufs*).
  - 16 TBytes - 16 KBytes (*vxfs*)
- For performance reasons and in order not to put data security at risk, the pieces of a mirror disk should **not** be virtual disks (except for memory disks).
- The maximum number of virtual disks in a system is currently restricted to 1024. This is a preset value in systems with more than 16 Mbytes of memory. The preset value is 50 in systems with up to 16 Mbytes of memory. This can be changed by the system administrator by setting the *MAXVDUNIT* default value (see *stune*(4)), which currently must be in the range 1 – 1,024.
- The number of pieces on a virtual disk is restricted to 100 and cannot be changed.
- By default, there are 32 device nodes for virtual disks (vdisk0 – vdisk31). If you use *dkconfig*(8) to configure virtual disks with higher minor numbers, the associated device nodes are created automatically.
- Virtual disks can be nested, i.e. a virtual disk can be used as part of another virtual disk. However, the nested virtual disk must be configured before the superordinate virtual disk. Furthermore, the number of permitted nesting levels is restricted to **five** by the system, although this should be regarded as a theoretical value only. In most cases, it is impractical to have more than three nesting levels.
- With *striped* virtual disks, all pieces must be the same size. The piece size must be a multiple of the cluster size.
- Similarly, all the pieces of a mirror disk must be the same size. The size of the entire mirror disk corresponds to the length of one of its pieces.
- Only one type of virtual disk, a mirrored root disk, can be used for the root partition and the primary swap area. A mirror disk for the root file system or the primary swap area must not contain further virtual disks.

  

  In addition, all other swap areas must be located on mirror disks comprising physical disk partitions.
- A virtual disk cannot be defined as part of a *statesave* device.
- Avoid configuring striped virtual disks with small stripe sizes (less than 16 Kbyte). Stripe sizes of 16, 32 and 64 Kbytes have proven most satisfactory. Depending on the application (e.g. in the case of a single process which reads or writes sequentially), larger stripe sizes (e.g. 1 Mbyte) are also practical.
- If you wish to mirror *striped* virtual disks, first define the mirror disks, then the *striped* virtual disks.
- A *statesave* device for a group can contain up to 500 entries. Each mirror disk of the group and all the pieces of these mirror disks count as one entry.
- Disks belonging to an OLR group in "non-hot swappable" SCSI tiers are connected to two or three SCSI channels. These channels must have have identical disk types at identical addresses which must also have identical partition tables. The offset and length cannot be specified. It is only possible to mirror identical partitions of disks with identical SCSI addresses on different SCSI channels. The disks can be mirrored on other SCSI addresses on different or identical SCSI channels in the case of "hot swappable" SCSI tiers.

If the configuration data of a disk is modified, the data stored on that disk is lost. For example, if a *striped* virtual disk is defined with two pieces, and the definition is changed to three pieces, all data on this virtual disk is lost.

The information required to calculate the piece and cluster sizes of a virtual disk using the physical disks can

be obtained from the output of the *dkpart* command (*Blk Count* column). The command

dkpart -lb /dev/ios0/rsdisk000s0

produces the following output for example

```
ios0/sdisk000 "MP14"□□□□Geometry 4005-3:16:64 (cyls:heads:sectors)
Sector size = 512 bytes□□□□□□□□□2002.5 Total Mbytes (1MB = 1024 * 1024 bytes)
Partition □Start Rule□□□End Rule□□□□□First□Blk□□□Last□Blk□□Blk□Count□□Size□(MB)
□□□□□0□□□□□□□□□□□4□□□□□□□□□□□□□200□□□□□□□□□□□4096□□□□□208895□□□□□204800□□□□□□100.0
□□□□□1□□□□□□□□□□>p0□□□□□□□□□□□600□□□□□□□□208896□□□□□823295□□□□□614400□□□□□□300.0
□□□□□2□□□□□□□□□□>p1□□□□□□□□□□□600□□□□□□□□823296□□□□1437695□□□□□614400□□□□□□300.0
□□□□□3□□□□□□□□□□>p2□□□□□□□□□□□600□□□□□□□□1437696□□□□2052095□□□□□614400□□□□□□300.0
□□□□□4□□□□□□□□□□>p3□□□□□□□□□□□600□□□□□□□□2052096□□□□2666495□□□□□614400□□□□□□300.0
□□□□□5□□□□□□□□□□>p4□□□□□□□□□□□1397□□□□□□2666496□□□□4097023□□□□1430528□□□□□□698.5
□□□□□6□□□□□□□□□□0□□□□□□□□□□□□□2001□□□□□□□□□□□□□0□□□□2049023□□□□2049024□□□□□1000.5
□□□□□7□□□□□□□□□□0□□□□□□□□□□□□□4001□□□□□□□□□□□□□0□□□□4097023□□□□4097024□□□□□2000.5
□□□□□8□□□□□□□□□□>p6□□□□□□□□□□□2000□□□□□□2049024□□□□4097023□□□□2048000□□□□□1000.0
□□□□□9□□□□□□□□□□0□□□□□□□□□□□□□0□□□□□□□□□□□□□□□□□□□□*□*□*□□□□Unused□□□□*□*□*
□□□□10□□□□□□□□□□0□□□□□□□□□□□□□4□□□□□□□□□□□□□□□□□□□0□□□□□□□4095□□□□□□□4096□□□□□□□□2.0
□□□□11□□□□□□□□□□0□□□□□□□□□□□□□1000□□□□□□□□□□□□□0□□□□1023999□□□□1024000□□□□□□500.0
□□□□12□□□□□□□□□□>p11□□□□□□□□□1001□□□□□□1024000□□□□2049023□□□□1025024□□□□□□500.5
□□□□13□□□□□□□□□□>p12□□□□□□□□□1000□□□□□□2049024□□□□3073023□□□□1024000□□□□□□500.0
□□□□14□□□□□□□□□□>p13□□□□□□□□□1000□□□□□□3073024□□□□4097023□□□□1024000□□□□□□500.0
□□□□15□□□□□□□□□□>p14□□□□□□□□□$□□□□□□□□□4097024□□□□4098047□□□□□□□1024□□□□□□□□0.5
# Partition layout (not to scale) :
|10-|-0-|---1----|------2------|-----3----|---4----|----------5---------|15|
|-------------------6----------------|-----------------8-------------|
|-----------------------------------7-----------------------------------|
|----------11----------|-------12------|------13-----------|-----14-------|
```

Figure 10: Screen output of dkpart

3.5 Configuration procedure

The configuration procedure is the same for all types of virtual disk, apart from root mirror disks (see Section "Special features with root mirror disks"), mirror disks of an OLR group (see Section "Configuring OLR mirror disks"), mirror disks in a DLM/OPS environment (see Section "Configuring virtual disks in a DLM/OPS environment"), and mirror disks in an OBSERVE environment (see Section "Configuring mirror disks for OBSERVE").

The individual steps involved in the configuration procedure are described below:

- editing the */etc/dktab* configuration file
- configuring virtual disks with *dkconfig*(8)

For mirror disks only:

- setting the status of virtual mirror disks using*dkmirror*(8).

If a file system is to be created on the virtual disk:

- creating a new file system using *mkfs*(1M).
- editing the */etc/vfstab* file.
- mounting the new file system using *mount*(1M).

3.5.1 Editing the /etc/dktab configuration file

The configuration of all virtual disks in the system is defined in the */etc/dktab* file. This file is used by the *dkconfig*(8) command to manage the virtual disks.

The system administrator is responsible for maintaining this file. As with */etc/vfstab*, access to this file should be strictly controlled, i.e. write authorization should be granted to the system administrator only.

The system administrator can modify this file using a text editor. The fields within the file are separated by blanks or tabs. If the first column of a line contains a number sign (#), this line is interpreted as comment.

Blank lines are ignored. The virtual disks can be defined in any order in the */etc/dktab* file. Restrictions occur only with nested virtual disks. These must be defined beginning with the lowest nesting level. When defining mirror disks, you must ensure that the definition is preceded by the associated entry for a *statesave* device.

The path names in the */etc/dktab* file refer to block-oriented devices. All offsets and lengths are specified in blocks. A block of a virtual disk always consists of 512 bytes which is identical with the sector size of the physical disk.

The */etc/dktab* file must contain two types of configuration line for each virtual disk; a declaration line for the virtual disk as well as *n* definition lines for the individual pieces of the virtual disk. The declaration line always begins in the first column, while the definition lines for the virtual disk pieces are preceded by a blank.

The example below shows an extract from a */etc/dktab* file:

```
# Concatenated virtual disk /dev/vd/vdisk0 concat 3     /dev/ios0/sdisk001s8     /dev/ios0/sdisk002s7
/dev/ios0/sdisk003s7 # Striped virtual disk # (cluster size: 32 blocks) /dev/vd/vdisk1 stripe 2 32
/dev/ios0/sdisk003s4     /dev/ios0/sdisk004s4 # Three small simple virtual disks" # (Size: 2400 blocks each)
/dev/vd/vdisk2 simple     /dev/ios0/sdisk010s7   0       2400 /dev/vd/vdisk3 simple     /dev/ios0/sdisk010s7
2400    2400 /dev/vd/vdisk4 simple     /dev/ios0/sdisk010s7   4800    2400
```

Each declaration line for a virtual disk consists of at least two fields containing the name of the device node and the type of virtual disk. Apart from *simple* virtual disks, all virtual disks require a third field, usually used for specifying the number of pieces contained on the disk. With memory disks, the size of the memory disk is specified here in 512-byte blocks. With *striped* virtual disks, the cluster size must be specified in a fourth field in 512-byte blocks. The example below shows declaration lines for the various types of virtual disk:

```
/dev/vd/vdisk0 simple /dev/vd/vdisk1 concat     2 /dev/vd/vdisk2 stripe     2   32 /dev/vd/vdisk3 mirror     3
/dev/vd/vdisk4 statsv     2 /dev/vd/vdisk5 memory     20480 /dev/vd/vdisk6 vmemory     40960
```

Virtual disks can also be combined to form groups. This is particularly important for mirror disks. A system can contain a maximum of one *statsv* disk, which stores all the status information of non-grouped mirror disks. Alternatively, *statesave* virtual disks can be used to store the status information relating to the mirror disks of a specific group. By grouping mirror disks, the configuration and management of mirror disks is considerably more flexible.

When defining a group of mirror disks, you must ensure that the suffix *group=n*is specified in the declaration of a *statesave* virtual disk and of all mirror disks in this group. This is demonstrated in the following example of a declaration of a *statesave* virtual disk and two mirror disks in group 0.

```
/dev/vd/vdisk0   statesave,group=0   2     /dev/ios0/sdisk002s15      0     32     /dev/ios0/sdisk003s15      0
32
/dev/vd/vdisk3   mirror,group=0      2     /dev/ios0/sdisk002s7     /dev/ios0/sdisk012s7
/dev/vd/vdisk4   mirror,group=0      2     /dev/ios0/sdisk003s7     /dev/ios0/sdisk013s7
```

The example below shows the declaration of non-grouped mirror disks and of the associated *statsv* virtual disk.

```
/dev/vd/vdisk0   statsv        2     /dev/ios0/sdisk004s15      0     32     /dev/ios0/sdisk005s15      0
32
/dev/vd/vdisk3   mirror        2     /dev/ios0/sdisk004s6     /dev/ios0/sdisk014s6
/dev/vd/vdisk4   mirror        2     /dev/ios0/sdisk004s8     /dev/ios0/sdisk014s8
```

Other types of virtual disk can also be combined in groups. This is useful if you wish to configure or deconfigure groups of virtual disks simultaneously.

Please note that virtual disks belonging to a group are not configured automatically by the *brc* script when the system is booted.

If you wish group 10, for example, to be configured automatically when you switch to multiuser mode, insert the following line in the */etc/brc* file

/sbin/dkconfig -vcg10

Since */etc/brc* is supplied as part of the operating system, groups can also be configured using start scripts in the */etc/rc2.d* directory.

Except with memory disks, each declaration line is followed by one or more definition lines. A definition line consists of one or three fields in which an individual piece of the virtual disk is defined. The first field is mandatory and specifies the path name of a block-oriented physical device. The other two fields are optional and contain the block offset and the length of the physical partition in blocks of 512 bytes. If these fields are specified, the virtual disk piece consists only of a section of the physical partition beginning at the specified offset and with the specified length. Otherwise, the entire physical partition is used.

Virtual disk definitions can be nested. In this case, the definition line for a virtual disk piece can consist of the device node of a block-oriented virtual disk device. However, you must ensure that this virtual disk device is already defined in the */etc/dktab* file and configured. The maximum number of permitted nesting levels is five, although it is impractical to have more than three nesting levels. An example of effective nesting is the use of a memory disk as part of a mirror disk.

/dev/vd/vdisk10 memory 20480

/dev/vd/vdisk2 mirror 3 /dev/ios0/sdisk002s6 0 20480 /dev/ios0/sdisk003s7 40960 20480 /dev/vd/vdisk10

3.5.2 Configuring OLR mirror disks

An alternative form of mirror disk configuration has been defined for the OLR function. This configuration is also defined in the */etc/dktab* file.

3.5.2.1 Non "hot swappable" SCSI tiers

An OLR group must be defined in the case of SCSI tiers that are not "hot swappable" (see page ...). The definition of an OLR group begins with the keyword *olr*, followed by the group number and the number *N* of pieces in the defined mirror disks. The values 2 and 3 are permitted for *N* (if the disks are connected to 2 or 3 SCSI channels). The subsequent *N* lines determine the SCSI channels to which the disks are connected. You should use the device addresses output by the *autoconf -l* command for the respective host adapter port (e.g. *ios0/sport03*).

The following lines contain the definition of a virtual disk. These are always in the format:

Name of virtual disk Type of virtual disk Disk with partition

The first virtual disk must be a statesave virtual disk, and is followed by mirror virtual disks.

Example:

olr,group=1 2 ios0/sport03 ios0/sport06 vdisk10 statesave disk0s15 vdisk0 mirror disk0s7 vdisk2 mirror disk1s6 vdisk3 mirror disk1s8

Make sure that the disk name is *disk*u**s**p (and **not** *sdisk*cc**us**p). Here, u stands for the number of the disk drive (SCSI ID, hexadecimal specification from *0* to *f*), and **s** is the constant specification for the following partition number p in the range *0* to *15*. The logical controller numbers are not specified with this syntax. These numbers are already contained in the definition of the host adapter port (e.g. *sport03*).

A conventional definition is given below:

/dev/vd/vdisk10 statesave,group=1 2 /dev/ios0/sdisk030s15 /dev/ios0/sdisk060s15 /dev/vd/vdisk0 mirror,group=1 2 /dev/ios0/sdisk030s7 /dev/ios0/sdisk060s7 /dev/vd/vdisk2 mirror,group=1 2 /dev/ios0/sdisk031s6 /dev/ios0/sdisk061s6 /dev/vd/vdisk3 mirror,group=1 2 /dev/ios0/sdisk031s8 /dev/ios0/sdisk061s8

The two definitions are **not** equally valid.

If an OLR mirror disk is defined on two physical partitions on two disks attached to two different SCSI channels (see definition above), these SCSI channels are locked for other applications. All disks on the relevant channels must be mirrored. Following configuration of the first OLR mirror disk, it is no longer

possible to access a physical partition on these channels. - Any attempt to create a file system (with *mkfs*) is rejected with the error *EBUSY* (device busy).

3.5.2.2 "Hot swappable" SCSI tiers

No group needs to be defined for " hot swappable" SCSI tiers (see page ...).

Example:

/dev/vd/vdisk10 statesave,olr 2 /dev/ios0/sdisk030s15 /dev/ios0/sdisk060s15 /dev/vd/vdisk0 mirror,olr 2 /dev/ios0/sdisk030s6 /dev/ios0/sdisk060s6

Direct access to all partitions of physical disks *ios0/sdisk030* and *ios0/sdisk060* is locked after this mirror disk has been configured . This does **not** affect other physical disks of SCSI ports *ios0/sport03* and *ios0/sport06*.

3.5.3 Configuring virtual disks in a DLM/OPS environment

Distributed mirror disks are used in a DLM/ OPS environment.

The definitions of the distributed mirror disks and of the associated *statesave* devices must be identical on all systems. Their status is kept consistent on the host systems involved.

The definition is identified by the keyword *ops*, followed by the group number and the number *N* of mirror disks subsequently defined. The values 2 and 3 are permitted for *N* (if the disks are connected to 2 or 3 SCSI channels). The subsequent *N* lines determine the SCSI channels to which the disks are connected.

The first virtual disk must be a *statesave* virtual disk, and is followed by *mirror* virtual disks.

Example:

/dev/vd/vdisk0 statesave,ops,group=0 2 /dev/ios0/sdisk002s15 0 32 /dev/ios0/sdisk003s15 0 32
/dev/vd/vdisk1 mirror,ops,group=0 2 /dev/ios0/sdisk002s2 /dev/ios0/sdisk003s2

It is also possible to combine the *ops* and *olr* functions.

Example:

olr,ops,group=3 2 ios0/sport02 ios0/sport04 vdisk100 statesave disk0s15 vdisk101 mirror disk0s7 vdisk102 mirror disk1s7

Virtual disks with the *ops* attribute are configured by the startup script */etc/rc2.d/S85dlm* when the system is booted. You can also use the *ops* attribute for all other virtual disks (apart from memory disks); these virtual disks are then configured automatically when the system is started.

3.5.4 Configuring mirror disks for OBSERVE

If a system in an OBSERVE environment takes over disks from a backup system following a system crash, some of the SCSI paths to the disks are changed.

For instance, if the master system is an RM600 and the backup system an RM400, the path names of the disks are different. With the RM600, the disks were addressed via */dev/ios0/sdiskxxx*. With the RM400, the corresponding path is */dev/ios1/sdiskxxx*.

If the mirror disks are taken over from another system, the information in the *statesave* device is used to determine that the names of the physical devices have changed. In this case, the driver monitoring the status information outputs the following message:

Configuration changed since last time, previous state no longer applicable

It switches all the pieces of the mirror disk to *disabled*.The mirror disk is then no longer available to the applications.

To avoid this, the keyword *switch* has been introduced for defining mirror disks. If the mirror disk is defined with this keyword, only part of the SCSI name is checked, namely the SCSI ID and the partition number. However, specifications such as the offset and length are still evaluated.

Example:

Master system RM600:/dev/vd/vdisk27 mirror,switch 2 /dev/ios0/sdisk123s7 /dev/ios0/sdisk246s7

Backup system RM400:/dev/vd/vdisk27 mirror,switch 2 /dev/ios1/sdisk073s7 /dev/ios1/sdisk016s7

After the RM400 disks are activated, the mirror disk is configured and the old status of the RM600 is assumed. Following the switch, a catch-up process need not be executed for these mirror disks.

Devices that even change their SCSI ID during a changeover process (RAID devices, see *rdswitch*(8) command) can be defined using the keyword *nocheck*. If this keyword is specified, the entire SCSI name of the device is not checked during the changeover. However, specifications such as the offset and length are still evaluated.

Example:

Master system RM600:

```
/dev/vd/vdisk28   mirror,nocheck   2    /dev/ios0/sraid086l0s5    /dev/ios0/sraid096l0s5
```

Backup system RM600:

```
/dev/vd/vdisk28   mirror,nocheck   2    /dev/ios0/sraid073l1s5    /dev/ios0/sraid095l2s5
```

If *switch* or *nocheck* is specified, the operating system cannot determine whether the mirror disk definition has been changed (e.g. after the operating system is restarted). The system administrator must therefore take full responsibility for this. If the mirror disk definition has changed, there is a danger that all data will be lost.

3.5.5 Configuring with dkconfig(8) and dkmirror(8)

The *dkconfig*(8) command provides complete control over the configuration of virtual disks. It is used to create definitions in the */etc/dktab* configuration file. *dkconfig*(8) uses the */etc/dktab* file to obtain information on the virtual disk configurations in the same way that *mount*(1M) uses */etc/vfstab* to obtain information on the mounted file systems. *dkconfig*(8) configures the virtual disks and creates an overview of the current configurations of virtual disks. Therefore, under no circumstances should the definitions of configured virtual disks in */etc/dktab* be modified as long as the virtual disk is configured and active. The definitions of unconfigured virtual disks, however, can be changed at any time.

When configuring several virtual disks, *dkconfig*(8) checks the disks for inadvertent overlays between physical disks. However, this applies only for virtual disks configured with the current *dkconfig*(8) call. Virtual disks configured previously or subsequently (e.g. from another group) are not taken into consideration. When defining a virtual disk therefore, the system administrator should ensure that the disk pieces do not overlap with existing file systems, data areas, or other virtual disks already configured.

If a virtual disk (irrespective of its definition) is simply configured, this poses no threat. The existing file systems and data are in danger only if the disk is activated and data is actually written to it. If you are quick to locate any mistakes in the definition, you can still deactivate the virtual disk. However, a virtual disk that is already configured and currently in use cannot be deconfigured. Any attempt to deconfigure a virtual disk that is being used will result in the error *EBUSY* (device busy).

If one or more of the virtual disks defined in */etc/dktab* contain file systems, *dkconfig(8)* must be executed before *fsck*(1M) and *mount*(1M), since a virtual disk can only be used after it has been configured. This takes place automatically when switching to multiuser mode. The */sbin/brc* script calls *dkconfig*(8) in order to configure the virtual disks in accordance with */etc/dktab*. This does **not** apply for virtual disks configured in groups.

If you wish to manually check virtual file systems in single-user mode, you must execute */sbin/brc* before calling *fsck*(1M) in the command line. If there are no configured virtual disks when *fsck*(1M) is called, the error *ENXIO* (device or address not available) is returned.

Although all virtual disk path names defined in */etc/dktab* or specified as an argument of *dkconfig*(8) are block-oriented device entries, both block- and character-oriented device entries are supported.

dkconfig(8) offers an entire range of options. Further information can be found in the manual pages on *dkconfig*(8). Some examples are given below, which demonstrate how to use the command:

- Configure all virtual disks defined in */etc/dktab*:

  # dkconfig -Ac

- Configure all OPS disks defined in */etc/dktab*:

 # dkconfig -Oc

- Output a short list of the configurations of all virtual disks:

  # dkconfig -Al
- Output a detailed list of the configurations of all virtual disks:

  # dkconfig -Alv
- Output a detailed description of virtual disk *vdisk2*:

  # dkconfig -lv /dev/vd/vdisk2
- Output a short description of the first four virtual disks:

  # dkconfig -l /dev/vd/vdisk[0-3]
- Configure all virtual disks of group 1:

  # dkconfig -cg1
- Deconfigure all virtual disks:

  # dkconfig -Au
- Deconfigure all OPS disks:

  # dkconfig -Ou
- Display the current *mroot* configuration according to */mrconfig.** :

  # dkconfig -C
- Display the current *mroot* configuration according to */etc/dktab*:

  # dkconfig -L
- Configure virtual disks (*mirror,root* and *mirror,swap*) according to *dktab*:

  # dkconfig -R

Using the input/output statistics which are displayed using the *-lv* option, you can determine the distribution of input/output operations over the various physical disks. In particular with *striped* virtual disks, this information can be used to ascertain the optimum settings. In the event of uneven distribution of input/output operations over the individual pieces of the virtual disk, either the file system or the virtual disk must be reconfigured.

The configuration procedure is now complete for all virtual disks, apart from mirror disks. The initial configuration of mirror disks requires some additional management tasks (for information on special features with mirrored root disks, see Section "Special features with root mirror disks").

Following initial configuration with *dkconfig*(8), all the pieces of a mirror disk are *disabled*. The entire mirror disk is thus in *DOWN* status, and cannot be used for file systems or as a data disk. When reconfiguring mirror disks, therefore, the system administrator must manually switch the mirror disk to *MIRRORED* status. The *dkmirror*(8) command is provided for this purpose.

The mirror disk must be initialized by performing the following steps (the specified commands are examples):

1. The command

   # dkmirror -eN *virtual_disk*

   is used to set a piece of a mirror disk to *enabled*. *N* stands for the number of the mirror disk piece (the count begins at 0). For *virtual_disk*, you must specify the path name of the mirror disk. In the case of a mirror disk with two pieces, the command could look as follows:

   # dkmirror -e0 -e1 /dev/vd/vdisk8

   If several mirror disks are defined, a group of mirror disks can also be set to *enabled*

   # dkmirror -ag*N*

   Here, *N* represents the number of the mirror disk group.

   The command

   # dkmirror -aA

is used to set the status of all mirror disks of all groups to *enabled*.

2. One piece of the mirror disk must then be set to *online*. This piece then acts as the data source for the automatic catch-up process that follows. The data from this piece is written to all other *enabled* pieces. This applies even if the mirror disk does not yet contain any valid data. Only one section of the mirror disk can be manually set to *online*.

   # dkmirror -o*N* *virtual_disk*

   Once again, *N* represents the number of the disk piece and *virtual_disk* the device node of the mirror disk. In our example above, the command could look as follows:

   # dkmirror -o0 /dev/vd/vdisk8

   The pieces of a mirror disk can also be switched directly from *disabled* to *online*. Steps 1 and 2 can thus be combined as follows:

   # dkmirror -o0 -e1 /dev/vd/vdisk8

3. After one of the pieces of the mirror disk has been switched to *online*, the system starts an automatic catch-up process. Information on the status of the mirror disk and its pieces can also be obtained using the *dkmirror*(8) command.

   Once step 1 has been performed, the output looks as follows:

   # dkmirror -lv /dev/vd/vdisk8
   Mirror disk /dev/vd/vdisk8 DOWN piece 0 /dev/ios0/sdisk001s1 enabled (catch-up required) piece 1 /dev/ios0/sdisk002s1 enabled (catch-up required)

   As soon as one of the pieces of the mirror disk has been set to *online* in step 2, the catch-up process begins immediately:

   # dkmirror -lv /dev/vd/vdisk8
   Mirror disk /dev/vd/vdisk8 NOT-MIRRORED (catch-up: 35%) piece 0 /dev/ios0/sdisk001s1 online (catch-up source) piece 1 /dev/ios0/sdisk002s1 enabled (catch-up in progress)

   When the catch-up process terminates successfully, all pieces are *online* and the entire mirror disk is in *MIRRORED* status:

   # dkmirror -lv /dev/vd/vdisk8

   Mirror disk /dev/vd/vdisk8 MIRRORED piece 0 /dev/ios0/sdisk001s1 online piece 1 /dev/ios0/sdisk002s1 online

   The *-L* option can be used to display a more comprehensive table of all configured mirror disks:

   # dkmirror -L

   VDISK TYPE STAT PIECE 0 PIECE 1 ----------+----------+----------
   *** ios0/sport03 *** *** ios0/sport05 *** vdisk10 Stat ONL /ios0/sdisk032s15 onl /ios0/sdisk052s15 onl vdisk11 MIRR DOW /ios0/sdisk032s1 dis /ios0/sdisk052s1 dis vdisk12 MIRR NOT /ios0/sdisk032s2 onl /ios0/sdisk052s2 40% vdisk13 MIRR MIR /ios0/sdisk032s3 onl /ios0/sdisk052s3 onl vdisk14 MIRR DOW /ios0/sdisk032s4 Mas /ios0/sdisk052s4 ena

   The *N%* specification indicates that a catch-up process is currently running and is *N* percent complete.

   With **# dkmirror -d***N virtual_disk*, you can disable piece *N* of a mirror disk. This is effective even if this piece is the last available online piece on the mirror disk. If you are forced to disable pieces of a mirror disk using *dkmirror* during normal operation, you must specify the additional option *-N*. This prevents the last available online piece of a mirror disk from being deactivated.

   With the *-mN* option, piece *N* of a mirror disk is declared as the master piece. The next time this mirror disk is configured, a catch-up process is started automatically, which copies the data from this piece to the other piece(s).

The *autoconf-l* command returns the device names of your disks and your SCSI adapter ports in a special format. These device names are *iosx/sdiskxxx* for disks and *iosx/sportxx* for ports (where *x* represents a number). For simplicity, the device names can be used for certain *dkmirror* calls.

The alternative form of the command has the following syntax:

dkmirror -*option autoconf-device-name*

Possible values for *option* are: *d, e, m* or *o*.

If an *autoconf-device-name* is specified in the format *iosx/sdiskxxx* (disk), the operation selected with the option is performed on all pieces of all mirror disks and *statesave* devices located on the specified physical disk. If a SCSI port is entered in the format *iosx/sportxxx* for *autoconf-device-name*, the operation is performed on all physical disks on this SCSI port.

Further information on the *dkmirror*(8) command can be found in the corresponding manual pages.

3.5.6 Prioritizing the catch-up process

To control the load on the system exerted by a catch-up process, it is possible to prioritize catch-up processes using

 # dkmirror -p*N (N=1,2,3)*

A low priority *(N=1)* results in a slow catch-up process with a low system load. Conversely, a high priority *(N=3)* represents a fast catch-up process with a high system load.

In some cases, the catch-up priority cannot be changed:

- if a catch-up process is already running; the new priority is not set until all catch-up processes are complete
- if OPS mirror disks are configured

During the system startup phase, the catch-up process is initiated with a low priority so that it does not affect the file system checks.

The *-P* option allows you to query the priority of the active catch-up processes.

 # dkmirror -P
The priority is set to 2 for all mirror catchups

3.6 Sample configuration

This section contains a sample configuration showing all the steps involved in configuring virtual disks.

1. Creating all the entries in the */etc/dktab* file:

   /dev/vd/vdisk1 statsv 2 /dev/ios0/sdisk000s15 /dev/ios0/sdisk001s15

   /dev/vd/vdisk2 mirror 2 /dev/ios0/sdisk000s2 /dev/ios0/sdisk001s2

   /dev/vd/vdisk3 concat 4 /dev/ios0/sdisk002s1 /dev/ios0/sdisk002s2 /dev/ios0/sdisk003s1 /dev/ios0/sdisk003s2

   /dev/vd/vdisk4 simple /dev/ios0/sdisk003s3 0 2400

   /dev/vd/vdisk5 simple /dev/ios0/sdisk003s3 2400 2400

   /dev/vd/vdisk6 simple /dev/ios0/sdisk003s3 4800 2400

   /dev/vd/vdisk7 memory 20480

   /dev/vd/vdisk8 mirror 3 /dev/ios0/sdisk002s3 0 20480 /dev/ios0/sdisk003s3 7200 20480 /dev/vd/vdisk7

   olr,group=1 2 ios0/sport03 ios0/sport06 vdisk10 statesave disk0s15 vdisk11 mirror disk0s7 vdisk12 mirror disk1s6 vdisk13 mirror disk1s8

   /dev/vd/vdisk10 statesave,olr 2 /dev/ios0/sdisk030s15 /dev/ios0/sdisk060s15 /dev/vd/vdisk0 mirror,olr 2 /dev/ios0/sdisk030s6 /dev/ios0/sdisk060s6

2. Configuring all the virtual disks:

   # dkconfig -Ac

3. Checking the configuration data:

   # dkconfig -Al

Creating and mounting a new file system

The configuration steps described in this section need only be performed if a file system is to be created on

the virtual disk.

Since *vdisk8* is a mirror disk, you must first set one of its pieces to *online*. In this example, we will set the other two pieces to *enabled*, thus starting a catch-up process:

 # dkmirror -o0 -e1 -e2 /dev/vd/vdisk8

New *vxfs* file systems are created using the Reliant UNIX command *mkfs*(1M). A character-oriented device entry and the size of the file system must be specified as parameters. If the entire virtual disk is to be used as a file system, enter "**-**" for the file system size.

 # mkfs -Fvxfs /dev/vd/rvdisk8 -

After the file system has been initialized, you should enter the new file system in the */etc/vfstab* file. If one of the file systems */usr*, */var*, */opt*, or */home* is located on a virtual disk, make sure that 0 is specified for the level.

/dev/vd/vdisk8 /dev/vd/rvdisk8 /usr vxfs 0 yes rw

For other file systems, a declaration line in this file looks as follows:

/dev/vd/vdisk8 /dev/vd/rvdisk8 /u8 vxfs 1 yes rw

In order to use the file system, you must mount it using the *mount*(1M) command.

 # mount /u8

The directory */u8* must already exist. Otherwise, an error message is output.

If this directory does not exist, create it using

 # mkdir /u8

and repeat the mount process.

As an alternative to a *vxfs* file system, you can also create a *ufs* file system. To do this, use the *newfs*(1M) command.

Further information on the *newfs*(1M), *mkfs*(1M), *fsck*(1M), and *mount*(1M) commands can be found in the corresponding manual pages, the System Administrator's Guide, and the Reference Guide for System Administrators.

3.7 Replacing a defective OLR disk

If an OLR disk is found to be defective, it can be replaced by the system administrator. The steps to be performed are provided under the *Hardware Configuration* menu item of the *SYSADM* system administration interface or the *XCONFIG* graphical user interface. Both hardware configuration interfaces are referred to in the following as *Config*.

The procedure is described below:

- The SCSI channel is stabilized by switching all mirror pieces located on disks on this channel to *disabled*. This is necessary for SCSI tiers that are not "hot-swappable", since the use of a disk here could disrupt the SCSI bus, thereby causing data corruption on other disks on the SCSI channel.
- The disk is configured out of the kernel.
- The disk is physically replaced.
- The disk is configured back into the kernel.
- All disabled mirror pieces are reset to *enabled*. This starts several catch-up processes.

Apart from the physical replacement of the disk, all of these actions are performed by the *Config* software. The system administrator simply follows the specified instructions.

If "hot-swappable" SCSI tiers are used, this is detected by *Config*. In this case, only the disk to be replaced is disabled. This eliminates the need to perform catch-up processes for the remaining disks, which means that the system load is lower. During the replacement process, applications can still access their data (with the exception of data on the disk to be replaced if this is not mirrored). Once all catch-up processes have been completed successfully, the system is restored to full performance.

3.8 Special features with root mirror disks

When using virtual disks to mirror the root partition and the primary swap area, additional marginal conditions must be taken into consideration. This is because data stored on the root disk is required for configuration. However, Reliant UNIX also allows these partitions to be configured as mirror disks. The security and availability of the entire system is thus increased considerably. Nevertheless, the configuration procedure for the root partition and the swap area is more involved than that for other mirror disks.

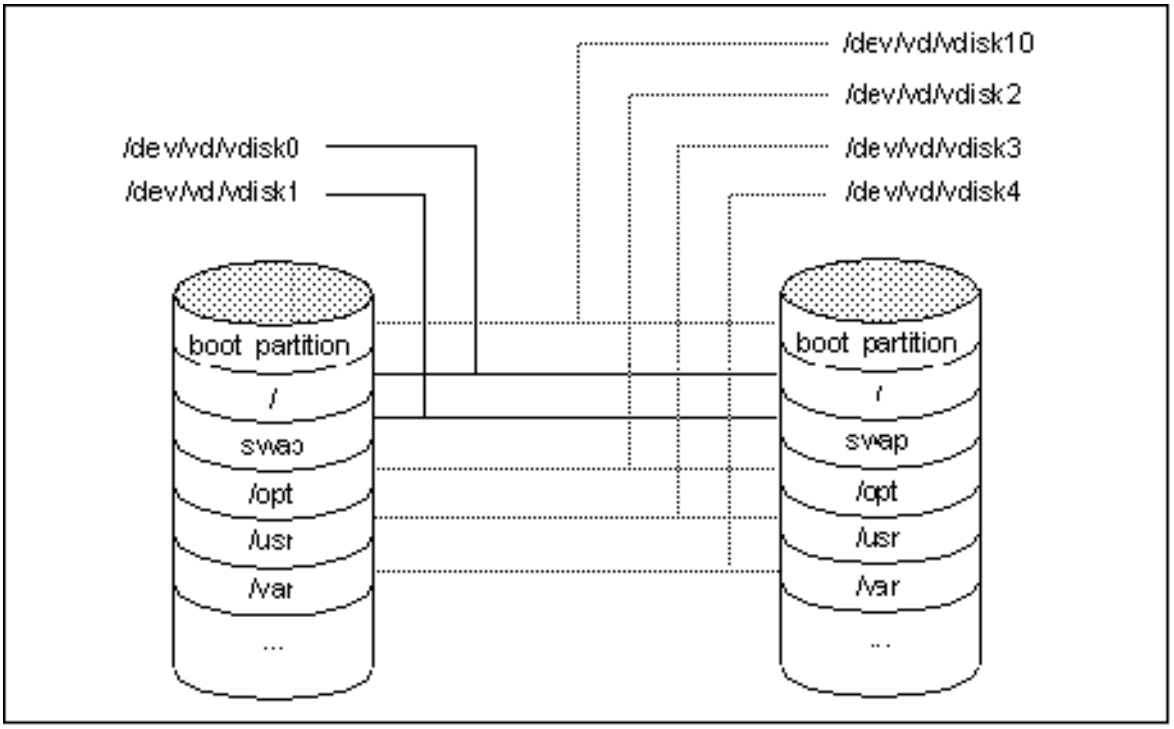

Figure 11: Sample configuration of a root mirror disk

Mirroring of the root partition and the swap area is practical only if all the pieces of the root disk are also mirrored. Only then can the system continue operating without interruption when a piece of the root disk fails. Root mirror disks are configured and managed using the same commands as with normal mirror disks, but with special parameters in some cases.

Once the operating system has been started from a root mirror disk, the disk configuration cannot be modified. Manual deactivation of all the pieces of this virtual disk is not permitted. At least one piece must be *online* at all times, as the entire system would otherwise crash.

For security reasons, the boot partition (partition 10) must also be mirrored. Only then can the system be rebooted following complete failure of the original root disk.

To ensure that mirroring of the root partition is activated, additional *boot2* or *sash* parameters must be transferred to Reliant UNIX when starting the system.

This is achieved by editing the */etc/default/boot*.

These parameters are:
BOOTPx=bootflags=0x40

or with an RM200/RM300/RM400 (the disk specifications are simply examples):
boot=ios0/sdisk000s0 root=ios0/sdisk000s0 swap=ios0/sdisk000s1 bootflags=0x40

An entry must be inserted in the */etc/dktab* file for the root partition and the primary swap area. These entries are practically identical to that of a mirror disk. However, they also contain the *root* or *swap* subtype in the type

field.

The configuration files n*/mrconfig.1* and */mrconfig.2* must then be created using the *dkconfig -R* command. These files are required when booting the system. They are based on entries in the */etc/dktab* file.

The entries for the root file system and the primary swap area must be modified in the i */etc/vfstab* file. Here, all entries for physical disks are replaced by the virtual disks.

Finally, the physical piece that now contains the correct data must be defined for the mirror disks. This piece is used as the source for a catch-up process the next time the system is booted, thereby ensuring that all pieces of the root mirror disk contain valid data. This is achieved using the *dkmirror -mN* command, where *N* represents the number of the piece containing the valid data. After the system is restarted and the automatic catch-up process is complete, the root partition and the swap area are mirrored.

Avoid mixing of OLR disks and non OLR disks.

3.8.1 Configuring a root mirror disk

The following example demonstrates the mirroring of all partitions of a root disk. The root disk is */dev/ios0/sdisk000*, and the mirror disk */dev/ios0/sdisk001*. The partition tables of both disks are identical. For status information, the partitions */dev/ios0/sdisk000s15* and */dev/ios0/sdisk001s15* are defined as a *statsv* virtual disk.

1. The */etc/vfstab* file is first edited.

   Old entries (you should save the old entries by inserting the comment character "#" in column 1)

   ```
   # /dev/ios0/sdisk000s0□□/dev/ios0/rsdisk000s0□□/□□□□□□ufs□□□0□□yes□□□□rw□
   # /dev/ios0/sdisk000s1□□/dev/ios0/rsdisk000s1□□-□□□□□□swap□□-□□-□□□□□□rw□
   # /dev/ios0/sdisk000s2□□/dev/ios0/rsdisk000s2□□/opt□□□vxfs□□0□□yes□□□□rw□
   # /dev/ios0/sdisk000s3□□/dev/ios0/rsdisk000s3□□/usr□□□vxfs□□0□□yes□□□□rw□
   # /dev/ios0/sdisk000s4□□/dev/ios0/rsdisk000s4□□/var□□□vxfs□□0□□yes□□□□rw□
   # /dev/ios0/sdisk000s5□□/dev/ios0/rsdisk000s5□□/home□□ufs□□□0□□yes□□□□rw
   ```

   New entries

   ```
   /dev/root□□□□□□□□□□□□□□/dev/rroot□□□□□□□□□□□□□□/□□□□□□ufs□□□0□□yes□□□□rw□
   /dev/swap□□□□□□□□□□□□□□/dev/rswap□□□□□□□□□□□□□□-□□□□□□swap□□-□□-□□□□□□rw□
   /dev/vd/vdisk2□□□□□□□□□/dev/vd/rvdisk2□□□□□□□□□/opt□□□vxfs□□0□□yes□□□□rw□
   /dev/vd/vdisk3□□□□□□□□□/dev/vd/rvdisk3□□□□□□□□□/usr□□□vxfs□□0□□yes□□□□rw□
   /dev/vd/vdisk4□□□□□□□□□/dev/vd/rvdisk4□□□□□□□□□/var□□□vxfs□□0□□yes□□□□rw□
   /dev/vd/vdisk5□□□□□□□□□/dev/vd/rvdisk5□□□□□□□□□/home□□ufs□□□0□□yes□□□□rw□
   /dev/vd/vdisk10□□□□□□□□/dev/vd/rvdisk10□□□□□□□□-□□□□□□raw□□□-□□-□□□□□□rw
   ```

2. The */etc/dktab* file must then be created or edited. You can append the keyword *olr* if you are using OLR disks (e.g. mirror, root,olr

   ```
   /dev/vd/vdisk15 statsv 2□
   □□□□/dev/ios0/sdisk000s15□
   □□□□/dev/ios0/sdisk001s15
   /dev/vd/vdisk0 mirror,root 2□
   □□□□/dev/ios0/sdisk000s0□
   □□□□/dev/ios0/sdisk001s0
   /dev/vd/vdisk1 mirror,swap 2    /dev/ios0/sdisk000s1    /dev/ios0/sdisk001s1
   /dev/vd/vdisk2 mirror 2    /dev/ios0/sdisk000s2    /dev/ios0/sdisk001s2
   /dev/vd/vdisk3 mirror 2    /dev/ios0/sdisk000s3    /dev/ios0/sdisk001s3
   /dev/vd/vdisk4 mirror 2    /dev/ios0/sdisk000s4    /dev/ios0/sdisk001s4
   /dev/vd/vdisk5 mirror 2    /dev/ios0/sdisk000s5    /dev/ios0/sdisk001s5
   /dev/vd/vdisk10 mirror 2    /dev/ios0/sdisk000s10    /dev/ios0/sdisk001s10
   ```

3. The configuration of the mirrored root and swap disk must be entered in the *mrconfig.1* and *mrconfig.2* files.

   # dkconfig -wR

4. All other virtual disks are then configured.

   # dkconfig -vwac

5. The status of the new mirror disks is initialized.

```
# for i in 0 1 2 3 4 5 10□
> do□
>□□□□□dkmirror□□-d0□□-d1□□/dev/vd/vdisk$i□
>□□□□□dkmirror□□-m0□□/dev/vd/vdisk$i□
> done
```

If you have specified the wrong mirror disk as the master piece, there is a danger that all data will be lost.

6. The boot parameters must be defined.

   For the RM600-xxx, the following line must be inserted in the */etc/default/boot* file:

   BOOTPx=bootflags=0x40

   Here, *x* represents a number in the range 2 to 7 which has not been used previously (example: BOOTP2=bootflags=0x40).

   For the RM200/RM300/RM400, the following lines are inserted in the */etc/default/boot* file (the disk specification are simply examples):

```
bootflags=0x40□
boot=ios0/sdisk000s0□
root=ios0/sdisk000s0□
swap=ios0/sdisk000s1
```

   If the default boot partition fails, another piece of the root mirror disk must be specified for *bootdev* or *boot* as an alternate disk.

7. Finally, the system must be restarted with *# init 6*.

   When the system is booted, automatic catch-up processes are started for all mirror disks.

Error sources:

1. Do not start any catch-up process before rebooting the system.
2. Specifying an incorrect master piece.

3.8.2 Configuring an OLR root mirror disk

If you want to convert your root mirror disk to an OLR root mirror disk, you must first deactivate the existing root mirroring including the mirroring for the file systems */usr*, */var*, */opt* and */home* (see Section "Deactivation of mirroring"). Then begin the OLR configuration completely from scratch.

An OLR root mirror disk is configured in the same way as a non-OLR root mirror disk (see Section "Configuring a root mirror disk") The only difference is that the keyword *olr* must be appended to each definition when creating the */etc/dktab* file in step 2:

```
/dev/vd/vdisk15 statsv,olr 2□
□□□□/dev/ios0/sdisk000s15□
□□□□/dev/ios0/sdisk001s15
/dev/vd/vdisk0 mirror,root,olr 2□
□□□□/dev/ios0/sdisk000s0□
□□□□/dev/ios0/sdisk001s0
/dev/vd/vdisk1 mirror,swap,olr 2□
□□□□/dev/ios0/sdisk000s1□
□□□□/dev/ios0/sdisk001s1
/dev/vd/vdisk2 mirror,olr 2□
□□□□/dev/ios0/sdisk000s2□
□□□□/dev/ios0/sdisk001s2
```

/dev/vd/vdisk3 mirror,olr 2□
□□□□/dev/ios0/sdisk000s3□
□□□□/dev/ios0/sdisk001s3
/dev/vd/vdisk4 mirror,olr 2□
□□□□/dev/ios0/sdisk000s4□
□□□□/dev/ios0/sdisk001s4
/dev/vd/vdisk5 mirror,olr 2□
□□□□/dev/ios0/sdisk000s5□
□□□□/dev/ios0/sdisk001s5
/dev/vd/vdisk10 mirror,olr 2□
□□□□/dev/ios0/sdisk000s10□
□□□□/dev/ios0/sdisk001s10

3.9 Operating mirrored root disks

You do not need to deactivate mirroring when installing the operating system.

3.9.1 Reinstalling the operating system

During a reinstallation, all relevant data on your system disk is overwritten.

Your root disk is not mirrored after installation. Follow the procedure described in Section "Configuration procedure").

You must reconfigure all virtual disks after you have reinstalled the operating system. If you have not yet reactivated root disk mirroring, then there is still a functional copy of the previous operating system on the other part of the root mirror disk. You could declare pieces of this copy as master pieces, enabling you to start the previous operating system again.

3.9.2 Installing an operating system update

Unlike previous SINIX releases, it is no longer necessary to disable the mirrored root disk during an update installation of the Reliant UNIX 5.44 operating system.

Proceed as described in the manual "Reliant UNIX Installation and Operation".

3.9.3 Deactivation of mirroring

3.9.3.1 Deactivation

If you wish to deactivate mirroring on a system running with a mirrored root disk proceed as follows:

1. Make a copy of the file */etc/vfstab*:

   # cp /etc/vfstab /etc/vfstab.bak

   Make a copy of the file */etc/dktab*:

   # cp /etc/dktab /etc/dktab.bak

2. Modify the file */etc/vfstab*:

   Old entries:

#□/dev/ios0/sdisk000s0□□/dev/ios0/rsdisk000s0□□/□□□□□□ufs□□□0□□yes□□□□rw□
/dev/ios0/sdisk000s1□□/dev/ios0/rsdisk000s1□□-□□□□□□swap□□-□□-□□□□□□rw□
#□/dev/ios0/sdisk000s2□□/dev/ios0/rsdisk000s2□□/opt□□□vxfs□□0□□yes□□□□rw□
#□/dev/ios0/sdisk000s3□□/dev/ios0/rsdisk000s3□□/usr□□□vxfs□□0□□yes□□□□rw□
#□/dev/ios0/sdisk000s4□□/dev/ios0/rsdisk000s4□□/var□□□vxfs□□0□□yes□□□□rw□
#□/dev/ios0/sdisk000s5□□/dev/ios0/rsdisk000s5□□/home□□ufs□□□0□□yes□□□□rw□
/dev/root□□□□□□□□□□□□□□□/dev/rroot□□□□□□□□□□□□□□□/□□□□□□ufs□□□0□□yes□□□□rw□
/dev/swap□□□□□□□□□□□□□□□/dev/rswap□□□□□□□□□□□□□□□-□□□□□□swap□□-□□-□□□□□□rw□
/dev/vd/vdisk2□□□□□□□□□/dev/vd/rvdisk2□□□□□□□□□/opt□□□vxfs□□0□□yes□□□□rw□
/dev/vd/vdisk3□□□□□□□□□/dev/vd/rvdisk3□□□□□□□□□/usr□□□vxfs□□0□□yes□□□□rw□
/dev/vd/vdisk4□□□□□□□□□/dev/vd/rvdisk4□□□□□□□□□/var□□□vxfs□□0□□yes□□□□rw□
/dev/vd/vdisk5□□□□□□□□□/dev/vd/rvdisk5□□□□□□□□□/home□□ufs□□□0□□yes□□□□rw□
/dev/vd/vdisk10□□□□□□□□/dev/vd/rvdisk10□□□□□□□□-□□□□□□raw□□□-□□-□□□□□□rw

New entries:

/dev/ios0/sdisk000s0□□/dev/ios0/rsdisk000s0□□/□□□□□□ufs□□□0□□yes□□□□rw□
/dev/ios0/sdisk000s1□□/dev/ios0/rsdisk000s1□□-□□□□□□swap□□-□□-□□□□□□rw□
/dev/ios0/sdisk000s2□□/dev/ios0/rsdisk000s2□□/opt□□□vxfs□□0□□yes□□□□rw□
/dev/ios0/sdisk000s3□□/dev/ios0/rsdisk000s3□□/usr□□□vxfs□□0□□yes□□□□rw□
/dev/ios0/sdisk000s4□□/dev/ios0/rsdisk000s4□□/var□□□vxfs□□0□□yes□□□□rw□
/dev/ios0/sdisk000s5□□/dev/ios0/rsdisk000s5□□/home□□ufs□□□0□□yes□□□□rw□
/dev/root□□□□□□□□□□□□□□□/dev/rroot□□□□□□□□□□□□□□□/□□□□□□ufs□□□0□□yes□□□□rw□
/dev/swap□□□□□□□□□□□□□□□/dev/rswap□□□□□□□□□□□□□□□-□□□□□□swap□□-□□-□□□□□□rw□
/dev/vd/vdisk2□□□□□□□□□/dev/vd/rvdisk2□□□□□□□□□/opt□□□vxfs□□0□□yes□□□□rw□
/dev/vd/vdisk3□□□□□□□□□/dev/vd/rvdisk3□□□□□□□□□/usr□□□vxfs□□0□□yes□□□□rw□
/dev/vd/vdisk4□□□□□□□□□/dev/vd/rvdisk4□□□□□□□□□/var□□□vxfs□□0□□yes□□□□rw□
/dev/vd/vdisk5□□□□□□□□□/dev/vd/rvdisk5□□□□□□□□□/home□□ufs□□□0□□yes□□□□rw□
/dev/vd/vdisk10□□□□□□□□/dev/vd/rvdisk10□□□□□□□□-□□□□□□raw□□□-□□-□□□□□□rw

As you see, the comment characters # in the first column of the original lines have been removed and the lines for the virtual disks are commented out instead.

3. Modify the file */etc/dktab* by inserting a comment character # in the first column of each line.

/dev/vd/vdisk15 statsv 2□
□□□□/dev/ios0/sdisk000s15□
□□□□/dev/ios0/sdisk001s15
/dev/vd/vdisk0 mirror,root 2□
□□□□/dev/ios0/sdisk000s0□
□□□□/dev/ios0/sdisk001s0
/dev/vd/vdisk1 mirror,swap 2□
□□□□/dev/ios0/sdisk000s1□
□□□□/dev/ios0/sdisk001s1
/dev/vd/vdisk2 mirror 2□
□□□□/dev/ios0/sdisk000s2□
□□□□/dev/ios0/sdisk001s2
/dev/vd/vdisk3 mirror 2□
□□□□/dev/ios0/sdisk000s3□
□□□□/dev/ios0/sdisk001s3
/dev/vd/vdisk4 mirror 2□
□□□□/dev/ios0/sdisk000s4□
□□□□/dev/ios0/sdisk001s4
/dev/vd/vdisk5 mirror 2□
□□□□/dev/ios0/sdisk000s5□
□□□□/dev/ios0/sdisk001s5
/dev/vd/vdisk10 mirror 2□
□□□□/dev/ios0/sdisk000s10□
□□□□/dev/ios0/sdisk001s10

4. Modify the bootflags in the file */etc/default/boot* to

bootflags=0x0

5. Restart Reliant UNIX by entering

 # init 6

3.9.3.2 Activation

To reactivate mirroring proceed as follows:

1. Copy back the saved files */etc/dktab.bak* and */etc/vfstab.bak*

 # cp /etc/dktab.bak /etc/dktab # cp /etc/vfstab.bak /etc/vfstab

2. Change the bootflags in the file */etc/default/boot* back to

bootflags=0x40

3. Then proceed as follows:

 # dkconfig -wR # dkconfig -vwac # for i in 0 1 2 3 4 5 10 > do > dkmirror -d0 -d1 /dev/vd/vdisk$i > dkmirror -m0 /dev/vd/vdisk$i > done

4. Reboot the Reliant UNIX (**# init 6** or **# init 5**).

After the catch-up process has updated all the mirror disks the full mirror disk functionality will be available again.

3.9.4 What to do if the system disk is mirrored and the disk containing the copy fails

The system will continue to operate.

In the example below, it is assumed that the disk containing the copy is *ios0/sdisk001*.

1. "Hot swappable" disks with OLR definition:
   - If the disk is installed in a "hot swappable" SCSI tier (only possible in the RM400-Cxx, Peripheral Cabinets BG31, BG32, and BG42, or RM600-Exx at present) and defined as an OLR disk, you can swap the disks using Config or XConfig (see the operating instructions for your system).
2. Non "hot swappable" disks:
   - If the disk is not installed in a "hot swappable" SCSI tier, deactivate all part 1 pieces of the mirror disks located on *ios0/sdisk001*:
   - **# dkmirror -d ios0/sdisk001**
   - Shut down the system at the appropriate time and replace the faulty disk with one that is identically partitioned. Restart Reliant UNIX again after the disk has been installed.

The disk must be formatted. If the new disk does not contain the same partition table as the replaced disk, or you are unsure, proceed as follows:

- Deactivate the *statesave* device on the newly installed disk (called *vdisk15*):

  # dkmirror -d1 /dev/vd/vdisk15
- Save the partitioning of the intact system disk (e.g. *sdisk000)* to a file:

  # dkpart -l /dev/ios0/rsdisk000s0 > ../FILENAME
- Copy this information to the new disk (e.g. *sdisk001*) with

  # dkpart -f ./FILENAME /dev/ios0/rsdisk001s0
- Restore the mirror disks to a consistent state:

  # dkmirror -e ios0/sdisk001

3.9.5 What to do if the system disk is mirrored and the system disk fails

The system will continue to operate.

In the example below, it is assumed that the system disk is *ios0/sdisk000* and the disk containing the copy is *ios0/sdisk001*.

1. "Hot swappable" disks with OLR definition:
   - If the disk is installed in a "hot swappable" SCSI tier (only possible in the RM400-Cxx, Peripheral Cabinets BG31, BG32, and BG42, or RM600-Exx at present) and defined as an OLR disk, you can swap the disks using Config or XConfig (see the operating instructions for your system).
2. Non "hot swappable" disks:
   - If the disk is not installed in a "hot swappable" SCSI tier, deactivate all part 0 pieces of the mirror disks located on *ios0/sdisk000*:

     # dkmirror -d ios0/sdisk000
   - The entries for *root*, *swap* and *boot* or *bootdev* in the */etc/default/boot* file must be changed from *ios0/sdisk000 to ios0/sdisk001*.
   - On the RM600 additionally enter

 # bootdisk sdisk001

to set the b oot string in the NVRAM (Non-Volatile Random Access Memory) so that booting now takes place from *sdisk001*.

- Shut down the system at the appropriate time and replace the faulty disk with one that is identically partitioned.
- After the disk has been installed, restart Reliant UNIX.
- In the case of an RM200, RM300, or RM400, the following must be specified if the system disk and the disk containing the copy are connected to SCSI channel *0*:

  PROM>dkncr(0,1,10)sash
  SASH>dkncr(0,1,0)unix

  If the disk containing the copy is connected to SCSI channel *1* (e.g. with RM400 C model *ios0/sdisk010*), the following must be specified:

  PROM>dkncr(1,0,10)sash
  SASH>dkncr(1,0,0)unix

The disk must be formatted. If this disk does not contain the same partition table as on the replaced disk (or if you are unsure) then proceed as follows:

- Deactivate the *statesave* device on the newly installed disk (called *vdisk15*):

  # dkmirror -d0 /dev/vd/vdisk15
- Save the partitioning of the non-replaced system disk (e.g. *sdisk001*) in a file:

  # dkpart -l /dev/ios0/rsdisk001s0 > ./FILENAME

  Save this information on the new disk (e.g. *sdisk000*) with

  # dkpart -f ./FILENAME /dev/ios0/rsdisk000s0
- Return the mirror disks to a consistent state:

  # dkmirror -e ios0/sdisk000
- Change the entries in */etc/default/boot* back to *ios0/sdisk000*.

  On a RM600 restore the boot string in the NVRAM to its original value with

  # bootdisk sdisk000

For RM200/RM300/RM400 please read the section "booting from Mirror Disk" in the manual "Reliant UNIX Installation and Operation".

Glossary

cache

A cache is a semiconductor memory that intercepts and speeds up the input/output requests issued to the hard disk by the system. Part of the hard disk data is stored in the cache, so that access to this data is expedited. A distinction is made between a write cache and a read cache. A write cache intercepts all write requests to a hard disk and very quickly declares access as complete. The data is actually written from the cache to the hard disk some time later. A read cache always stores the last data requested. If the user wishes to access this data again, the cache can provide it much faster than the physical disk. As a rule, the cache is fully transparent to the user.

catch-up process

A catch-up process is used to switch a *mirror disk* piece from *enabled* to *online*. Only one piece of the mirror disk can be manually switched to *online* by the system administrator. All other pieces can only be set to *enabled*. The system then starts an automatic catch-up process. This is the only way of ensuring that all the pieces of a mirror disk contain identical data.

concatenated virtual disk

A concatenated virtual disk consists of a series of two or more disks or partitions. It can comprise physical disks or further virtual disks. Concatenated virtual disks allow you to create disks whose size is restricted only by the maximum address space of the system.

cylinder

All the *tracks* of a magnetic disk which have the same radius, but which may be on different surfaces. All the tracks of a cylinder can be read without having to reposition the read/write unit.

device class number

Device classes are assigned particular device class numbers. The device class number for virtual disks is always 7. Device class numbers must be specified when creating device entries with *mknod*.

device number

Identical with the *unit number* in virtual disks.

disabled

Possible status of a piece of a *mirror disk*. A *disabled* piece is unavailable for any input/output operations or catch-up processes. All newly configured pieces of a mirror disk are *disabled*. Pieces in which input/output operations failed are automatically switched to this status by the system.

DOWN

Possible status of a *mirror disk*. If a mirror disk does not contain any *online* pieces, it can no longer be accessed. The disk is thus in *DOWN* status.

ECC error

A disk error in which the ECC (Error Correction Code) could not be determined for a *sector*, which suggests inconsistent data in that sector. Since the ECC must be recalculated for each write operation, an ECC error can sometimes be corrected by rewriting the sector. In other cases, this error may be due to defective media, which means that the track must be reformatted.

enabled

Possible status of a piece of a *mirror disk*. An *enabled* piece is unavailable for any input/output operations issued by users or programs. However, it is available as the target of a *catch-up process*. Following a successful catch-up process, the piece is switched to *online*. A piece can only be switched manually to *enabled* by the system administrator using the *dkmirror*(8) command.

header mismatch

A disk error in which the *sector* cannot be identified. This error is normally due to a damaged sector head caused by a power failure or a hardware defect. This defect can be rectified by reformatting the track. This may involve mapping defective tracks to alternate tracks.

"hot swappable"
Disks can be replaced while the system is running in "hot swappable" SCSI tiers (in RM600-E, RM400-Cxx and auxiliary cabinets BG70/BG71 and BG31, BG32 and BG42) even if other disks are still active.

length
Number of blocks (of 512 bytes) which determines the size of a virtual disk.

major number
See *device class number*.

master piece
The piece of a virtual mirror disk which is defined as containing valid data. With a catch-up process, the master piece is used as the data source. Under Reliant UNIX, a piece of a mirror disk can be declared as the master piece using the *dkmirror* utility.

memory disk (RAM disk)
Memory disks are areas in the system memory which provide users and application programs with the same interfaces as disk storage systems via corresponding device drivers. As far as the user is concerned, therefore, a memory disk is a high-speed virtual disk implemented as a memory or vmemory virtual disk. When using memory disks, you must remember that its contents are volatile. Following a system shutdown or crash, all data stored on the memory disk is lost. Memory disks are thus ideally suitable for storing temporary data. They can also be used as pieces of a *mirror disk*, thereby combining the security of mirror disks with the high-speed access capability of memory disks.

minor number
See *device number*.

mirror disk
Mirror disks are virtual disks that simultaneously perform all input/output operations on two or more disks. They can consist of physical disks or further virtual disks. When writing to mirror disks, the new data is written to all pieces. The write speed of a mirror disk is thus the same as that of its slowest piece. When reading from mirror disks, data is read from the piece that can provided the fastest access. The read speed of a mirror disk is thus generally higher than that of an individual disk. Mirror disks significantly increase the security and availability of disks, and thus of the entire system. As long as at least one piece of the mirror disk is operating without errors, input/output operations can be performed on the disk without restriction. In the event of defective mirror disk pieces, these can usually be repaired without having to shut down the entire system. Mirror disks are the only disk type that can also be used for the root partition and the swap area.

MIRRORED
Possible status of a *mirror disk*. A mirror disk is in *MIRRORED status if at least two of its pieces are online*.

mounting a file system
Mounting a file system involves inserting a newly created file system into any position in the directory tree using the *mount* command. Only then can the user access these areas.

NOT-MIRRORED
Possible status of a *mirror disk*. A mirror disk is in NOT-MIRRORED status if only **one** of its pieces is *online*. The mirror disk can still be used without any problems, but enhanced data security is no longer provided.

offset
An offset is a relative pointer. With virtual disks, an offset can be specified in order to determine the start block for the virtual disk within a physical *partition*. It specifies the first block of the virtual disk in the physical partition.

online

Possible status of a piece of a *mirror disk*. All write operations on the mirror disk are performed in an *online* piece. Read operations are performed only in the *online* piece that can provide the fastest access to data.

online replacement

Replacing disks while the system is .

Partition

A partition is a piece of a physical disk, which is defined in the system kernel.

piece of a virtual disk

The components defined in */etc/dktab* are known as pieces of a virtual disk. Depending on the disk type, a virtual disk can consist of one, two, or more pieces. For instance, simple virtual disks only have one piece. *Mirror disks*, however, must contain at least two pieces. Since the original data and the copies are identical, we refer only to the pieces of a mirror disk.

raw disk partition

A raw disk partition is an area that can be accessed directly by one or more applications without using a file system. It can be a physical disk partition or a virtual disk.

root mirror disk

The only type of virtual disk that can also be used for the root partition and primary swap area of the system. Root *mirror disks* are created and managed using the same commands as normal mirror disks. However, since normal virtual disks require information stored in the root area, the procedure for creating a root mirror disk is more involved.

sector

Each disk is logically divided into concentric circles known as *tracks*. The tracks are in turn logically divided into "slices", just like a pie. These "slices" are known as sectors.

simple virtual disk

A virtual disk that consists of one piece of a disk drive *partition*.

statesave and statsv virtual disk (statesave device)

Statesave and statsv virtual disks are required only when using *mirror disks*. The are used for nonvolatile storage of the status of mirror disks. Since this data occupies relatively little space, statesave and statsv virtual disks can be kept very small. This type of virtual disk cannot be used by users or application programs.

striped virtual disk

A striped virtual disk is one where the sequential block groups are distributed over two or more physical disks or partitions in accordance with a particular algorithm. Optional access to a striped virtual disk is thus distributed over several physical disks and/or controllers. Once the parameters are set accordingly, the read/write speed of a striped virtual disk is much higher than that of a physical disk. The individual pieces of a striped virtual disk must be the same size.

track

Each disk is logically divided into concentric circles known as tracks. The tracks are in turn logically divided into "slices", just like a pie. These "slices" are known as *sectors*.

unit number

To ensure that a system can support several virtual disks, each virtual disk is assigned a unit number. The unit number is derived from the device entry of the virtual disk (*/dev/vd/vdisk8*). Numbering begins at zero and is limited to 50 by default on RM200/RM300/RM400 systems, although this limit value can be increased without any problems. With RM600 systems, the limit value is defined as 1024 and cannot be changed.

Abbreviations

BBU
: Battery Backup Unit

DLM
: Distributed Lock Manager

Ebyte
: Exabyte

ECC
: Error Correction Code

Gbyte
: Gigabyte

ID
: Identifier

Kbyte
: Kilobyte

Mbyte
: Megabyte

MTBF
: Mean Time Between Failure

NFS
: Network File System

NVRAM
: Non-Volatile Random Access Memory

OLR
: Online Replacement

OPS
: Oracle Parallel Server

Pbyte
: Petabyte

RAID
: Redundant Array of Independent Disks

RFS
: Remote File System

SASH
: Stand-Alone Shell

SCSI
: Small Computer System Interface

Tbyte
: Terabyte

TCP/IP
: Transport Control Protocol / Internet Protocol

UFS
: UNIX File System

VXFS
: Veritas Extended File System

Related publications

SINIX 5.44
User Guide

Target Group
Users

Contents
Overview of the Reliant UNIX operating system and an introduction into the general basics for users.

SINIX 5.44
Commands, Users' Reference Manual
Description

Target Group
Reliant UNIX shell users

Contents
Description of Reliant UNIX commands in alphabetical order.

SINIX 5.44
System Administrator's Guide
User's guide

Target Group
System administrators

Contents
Introduction to the administration of Reliant UNIX systems.

SINIX 5.44
System Administrator's Reference Manual
Reference manual

Target Group
System administrators

Contents
Commands and application programs for system maintenance, file formats, and special files for system administration, diagnostic information.

SINIX 5.44
System Administration and Hardware Configuration Using the SYSADM User Interface
System Administrator's Guide

Target Group
System administrators

Contents
Operating SYSADM, system administration with SYSADM (administer file systems and network services, install software, system configuration, user administration), Config.

SINIX 5.44
Hardware Configuration with Config under SINIX/windows
Product Manual

Target Group
System administrators, service engineers

Contents
Description of the Config tool for hardware configuration of RM systems under the graphical user interface SINIX/windows.

SINIX
OBSERVE V1.2

User Guide

Target Group
System administrators, programmers

Contents
This manual describes OBSERVE V1.2. It is intended for system administrators who configure and support OBSERVE, and for programmers who wish to use the C interfaces offered with OBSERVE.

Reliant UNIX
SIDLM V1.2

Target Group
System administrators, Service engineers

Contents
This manual describes the software installation procedure for the SINIX Distributed Lock Manager (SIDLM) software package and how to administer cluster configurations.

Ordering manuals

Please apply to your local office for ordering the manuals.