Logical Volume Manager Basics

Contents

What is LVM Really?

- High Level Commands
- Intermediate Commands
- Library Commands
- LVM Device Driver
- Disk Device Driver
- SCSI Device Driver
- Actual Disk

High Level Commands

- Commands used by SMIT
- Commands used by users in command line or shell scripts
- High levels of error checking

Notes: High Level Commands

Notes: Most of the LVM commands reside in /usr/sbin. There are exceptions such as the library commands which reside in /usr/lib/liblvm.a and /usr/lib/libsm.a and /etc/cfgvg. The High Level commands are a combination of shell scripts and C binaries:

chlv, chpv, chvg, cplv, exporting, extendlv**, extendvg, importvg, lslv*, lsvg*, lsvgfs*, migratepv, mklv, mkvg, reducevg, reorgvg, rmlv, rmlvcopy, syncvg, synclvodm, varyonvg, varyonvg*

* indicates that this command is a binary

** this command also is used for the mklvcopy command

These commands are those that SMIT and experienced users are meant to execute directly.

Intermediate Commands

- Commands used by High Level commands
- Commands provide usage statements - unfortunately
- Some level of error checking still provided, no documentation on output

Notes: Intermediate Commands

Notes: The Intermediate commands are all C binaries:

getlvcb, getlvname, getlvodm, getvgname, lvgenmajor, lvgenminor, lvrelmajor, lvrelminor, putlvcb, putlvodm, lchangelv, lcreatelv, ldeletelv, lextendlv, lquerylv, lreducelv, lresynclv, lchangepv, ldeletepv, linstallpv, lquerypv, lresyncpv, lcreatevg, lqueryvg, lqueryvgs, lvaryonvg, lvaryoffvg, lresynclp, lmigratepp.

The commands are called by the High Level command shell scripts. These commands unfortunately give usage statements when queried from the command line, which has given users the incorrect impression that these commands are for general use in LVM. This is not true. These commands have been “tuned” for use by the High Level shell scripts. Because it is known that only high level shell scripts will be calling these intermediate commands, certain error checking is not performed by the intermediate commands. The High Level commands are responsible for screening potentially dangerous errors or conditions out.

The removal of the unintended usage statements has been considered, but users have asked LVM to leave them, even if they aren’t documented.

Library Calls - API

- Library calls to be used as Application Program Interface
- Some error checking provided
- Calls documented in Infoexplorer or books
- Very rarely used

Notes: Library Calls - API

Notes: The LVM library calls are part of the LVM Applications Programmer Interface (API) which lets programmers access the LVM layer directly through the library layer provided by /usr/lib/liblvm.a. In our experience, very few people use this API because of the complexity of issues within LVM. Users mostly rely on shell scripts that call the High Level commands. The API library functions begin with **lvm**_ and are documented within Infoexplorer. These commands are only called in situations where the user is writing C code and does not want to access the High Level shell scripts from within a C program. Or, they require a faster response from LVM than they can achieve with shell scripts. Writing an LVM specific program using the API certainly does give a performance improvement to some applications. Note, the LVM library API is not a thread-safe library.

LVM Device Driver

- Access same as those for typical IBM device driver
- Checking provided, but assumes user has basic device driver knowledge
- Rarely used. If used wrong, disastrous results!

Notes: LVM Device Driver

Notes: The LVM device driver comes in two portions, the pinned portion and the non-pinned portion. The pinned portion is called /usr/lib/drivers/hd_pin and /usr/lib/drivers/hd_pin_bot. Before AIX 4.1, the driver was just called hd_pin and the entire driver was pinned into memory (not pageable). With AIX 4.1, the driver's true non-pageable portion is in hd_pin_bot and the hd_pin is now pageable. The LVM device driver is either called by the jfs filesystem or the lvm library routines. When a request is received by the LVM device driver, it calls the disk device driver.

Disk Device Driver

- Access same as those for typical IBM device driver
- Sometimes other vendors use the disk device driver as pseudo driver for their products

Notes: Disk Device Driver

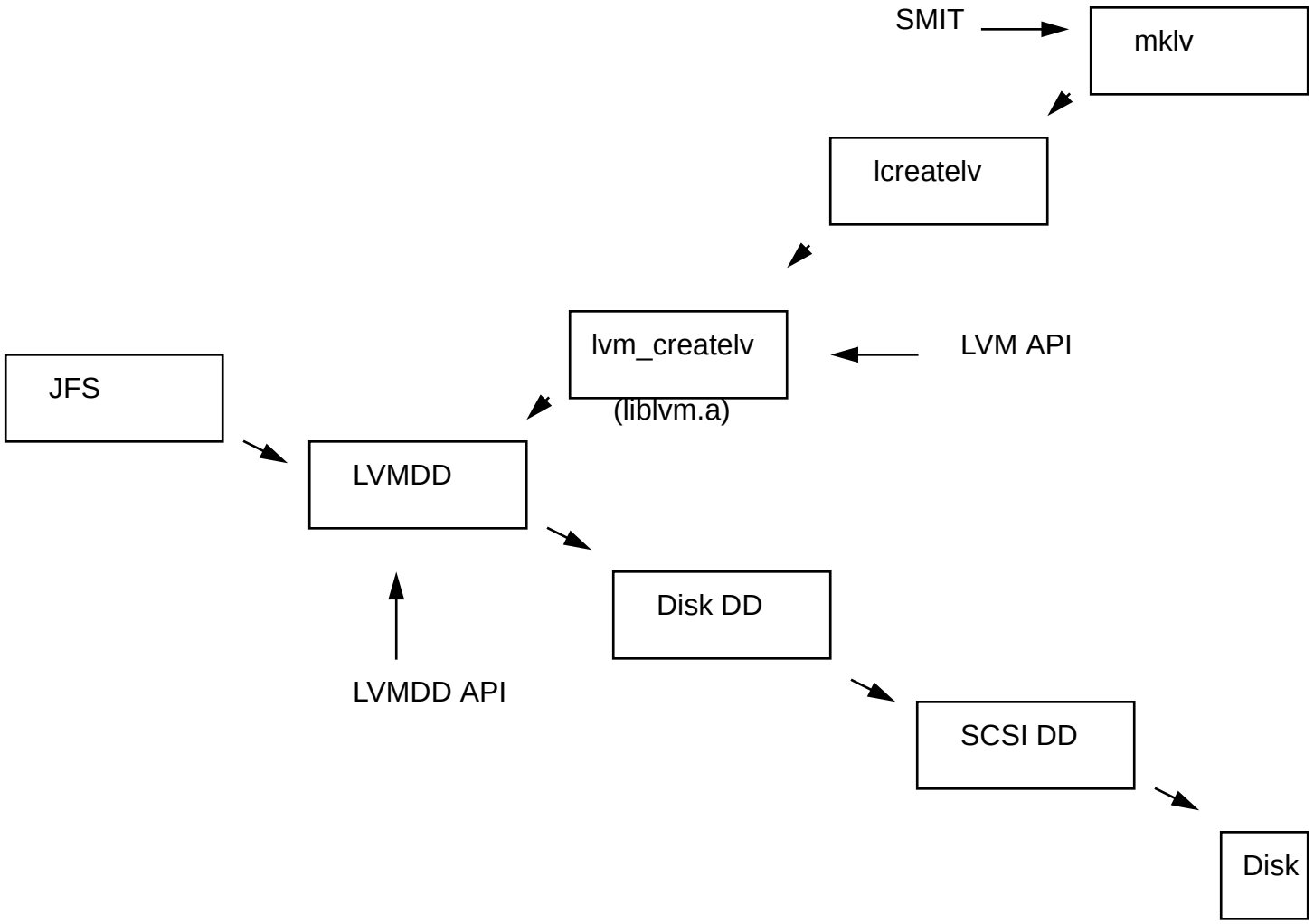
Notes: There are various disk device drivers in AIX. The most common disk device driver is the one for scsi device drivers, /usr/lib/drivers/scdisk and /usr/lib/drivers/scdiskpin. The second most common disk device driver is probably the one for the 9333 serial dasd device. This is found in the binaries /usr/lib/drivers/sd and /usr/lib/drivers/sdpin. In both device drivers, the portion called “pin” is that which is pinned into memory and cannot be paged out of memory. When a request from the LVM device driver is received by the disk device driver, the request is packaged and then transferred down to the scsi device driver.

SCSI Device Driver

- Access same as those for typical IBM device driver
- Very useful to access devices directly through this interface, although people prefer not to use this access point

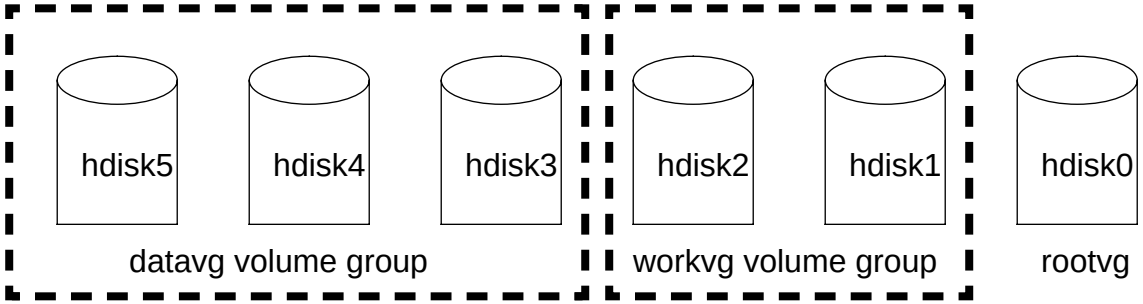
Notes: SCSI Device Driver

Notes: There are several scsi device drivers for AIX and they have the common name string of “scsi” residing as part of their name. The most common of them is the original SCSI-1 device driver /usr/lib/drivers/hscsidd. The scsi device driver takes a command from a scsi device such as tape, disk, scanner, etc...and processes it to be sent to the scsi device connected onto the scsi bus. The scsi device is device neutral, it does not know or care which device sent it a command or even what that command is. It treats all requests the same and puts them into the same type of scsi command packaging required by all scsi devices.

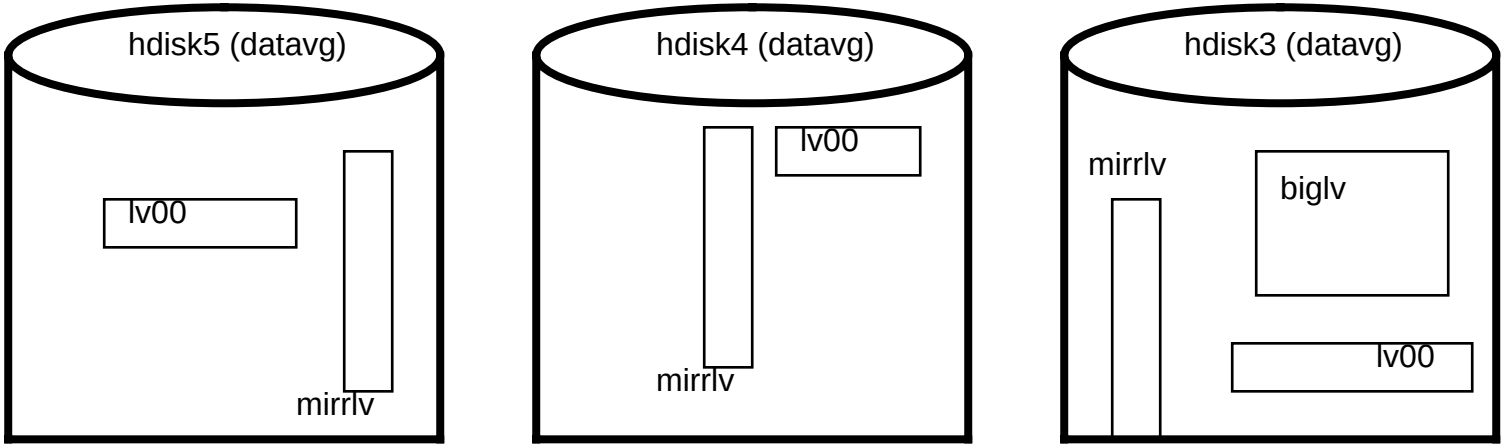

SMIT
mklv
lcreatelv
lvm_createlv
(liblvm.a)
LVM API
JFS
LVMDD
Disk DD
LVMDD API
SCSI DD
Disk

Terms

- Physical Volumes
- Volume Groups
- Logical Volumes
- Mirrors



On one system, many disks can make up a volume group. Disks cannot be shared between volume groups. The entire disk is dedicated to being part of one volume group.



Logical volumes reside only on volume groups. A logical volume can span multiple disks within a volume group (but cannot span multiple volume groups): lv00. Or, a logical volume can just reside on one disk: biglv. Finally, a logical volume can have multiple copies of itself: mirrlv.

Physical Volumes

- Disk Independence
- PVID’s and how they configure

Notes: Physical Volumes

Notes:

Disk Independence

The basic design of the interaction between LVM and the disk device driver has always assured that LVM's use of the disk would behave the same regardless of the type of disk being used in LVM. Thus, a disk drive such as the serial dasd drive (9333 drives) behave the same in LVM as the standard SCSI drive, although they use different disk and adapter device drivers. LVM just concentrates on the special file that represents the disk drive. Although it is not a requirement by LVM, almost all the disks are configured with the name hdiskX. The only exception to this transparent use of disks by LVM are the read/write optical disk drives. There is special code in the high-level LVM commands which checks for the existence of optical drives. Because of the portability of these type drives, special handling must take effect to guarantee that LVM knows the true identity of the optical disk drives.

PVID's and how they configure

When a disk is configured to a system for the first time, it shouldn't have a PVID if it is a brand new disk. When it is used in a volume group, the user sees a message to the effect of:

Making this disk a physical volume

Which is another way of saying that a PVID is being placed onto the disk. The PVID is an amalgamation of the machine's serial number (from the systems EPROMs) and the date that the PVID is being generated. This combination insures the extremely low chance of two disks being created with the same PVID. Finally, when a system is booted, the disk configurator goes and looks at the PVID sitting on each disk platter and then compares that to an entry in ODM. If the entry is found, then the disk is given the hdiskX name that is associated with the ODM entry for the PVID. If there is no PVID, the configuration routines will automatically assign the next free hdisk name from the pool of "free" hdisk names. Note that if a hdisk has been removed with the "rmdev -l hdiskX -d" command, then this hdisk name will be available for reuse by a later disk.

Warning: Some users use the “dd” command to copy the entire contents of one disk to another disk. This is an unsupported method of “quick install”. What the user forgets in this action is that they are literally copying over the PVID of the first disk onto the platter of the second disk. The extremely rare consequence of this “dd” is that the user may have, in the distant future, two disks attached to the same RS/6000 with the same PVID. When this occurs, EXTREME confusion will occur.

Volume Groups

- Portability of volume groups
- Mix and Match disks
- Volume Group Descriptor Area
- Volume Group Status Area
- Quorum
- VGID
- A Graphical View of VGDA expansion

Notes: Volume Groups

Notes:

Portability of volume groups

The one great attribute of LVM is the ability of the user to take a disk or sets of disks that make up a volume group and take it to another RS/6000 system and introduce the information created on another machine onto the second machine. This ability is provided through the Volume Group Descriptor Area (VGDA) and the logical volume control block (lvcb). The design of LVM also allows for accidental duplication of volume group and logical volume names. If on the new machine, the volume group or logical volume names being imported already exist, then LVM will generate a distinct volume group or logical volume name.

Mix and Match disks

As mentioned before, LVM allows the user to attach disks to volume group, regardless of what type of physical device it true is or what type of device driver is used to operate the device. Thus, RAID systems, serial dasd drives, and the plain SCSI drives can all make up one volume group that may reside across several adapters. The physical location of each drive and the true nature of the drive doesn't matter to LVM as long as the disk device drivers follow a certain format required by LVM in order to create logical volumes on those drives.

Volume Group Descriptor Area

The VGDA is an area at the front of each disk which contains information about the volume group, the logical volumes that reside on the volume group and disks that make up the volume group. For each disk in a volume group, there exists a VGDA concerning that volume group. This VGDA area is also used in quorum voting. The VGDA contains information about what other disks make up the volume group. This information is what allows the user to just specify one of the disks in the volume group when they are using the "importvg" command to import a volume group into an AIX system. The importvg will go to that disk, read the VGDA and find out what other disks (by PVID) make up the volume group and automatically import those disks into the system (and its) ODM as well. The information about neighboring disks can sometimes be useful in data recovery. For the logical volumes that exist on that disk, the VGDA gives information about that logical volume so anytime some change is done to the status of the logical volume (creation, extension, or deletion), then the VGDA on that disk and the others in the volume group must be updated.

Volume Group Status Area

The Volume Group Status Area (VGSA) is comprised of 127 bytes, where each bit in the bytes represents up to 1016 Physical Partitions that reside on each disk. The bits of the VGSA are used as a quick bit-mask to determine which Physical Partitions, if any, have become stale. This is only important in the case of mirroring where there exists more than one copy of the Physical Partition. Stale partitions are flagged by the VGSA. Unlike the VGDA, the VGSA's are specific only to the drives which they exist. They do not contain information about the status of partitions on other drives in the same volume group. The VGSA is also the bit masked used to determine which physical partitions must undergo data resyncing when mirror copy resolution is performed.

Quorum

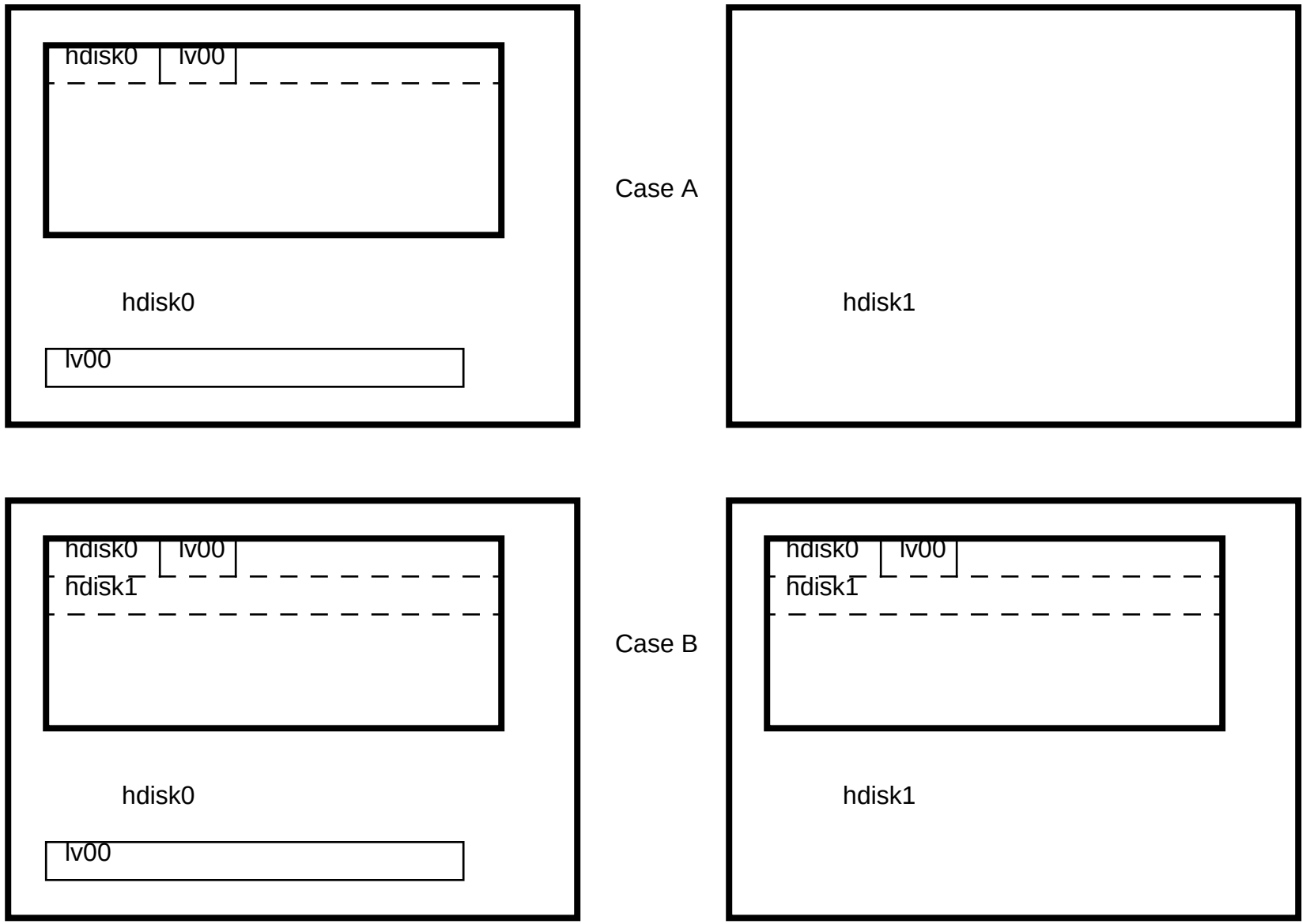
Quorum is a sort of "sanity" check that LVM uses to resolve possible data confliction and prevent data corruption. Quorum is a method by which 51% or more quorum votes must be available to a volume group before LVM actions can continue. Quorum is issued to a disk in a volume group according to how the disk was created within the volume group. When a volume group consists of one disk, there are two VGDA's on that disk. Thus, this single disk volume group has a quorum vote of 2. When another disk is added to the volume group with an "extendvg", then this new disk gets one VGDA, but the original, first disk still retains the two VGDA's. When the volume group has been extended to three disks, the third disk gets the spare VGDA sitting on the first disk and then each disk has a quorum vote of 1. Every disk after the third disk is automatically given one VGDA, and thus one vote.

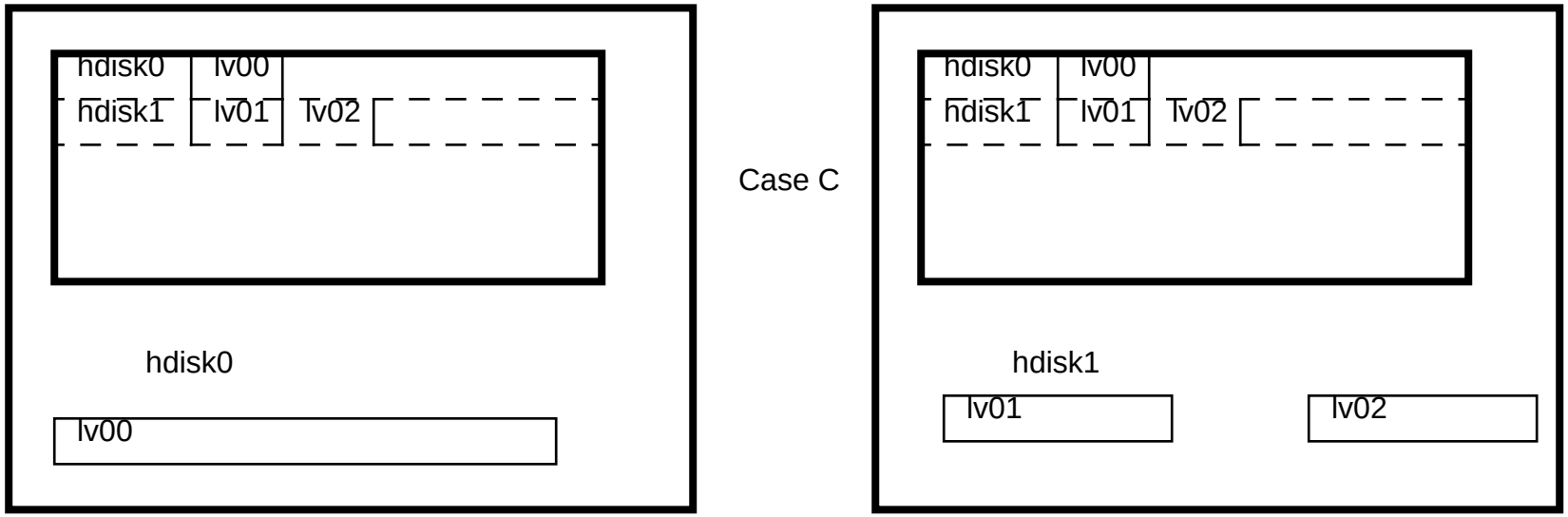
VGID

Just as the PVID is a soft serial number for a disk, the VGID is the soft serial number for the volume group. It is this serial number, not the volume group's ascii name, which all low level LVM commands reference. Additionally, it is the basis for the LVIDs created on that VGID.

A Graphical View of VGDA Expansion

This shows how the VGDA grows from one disk to another and how information is carried in the VGDA which cross-references information among the disks in a volume group. The exact VGDA structure is not defined in this section because it is quite complex and contains IBM confidential information.


hdisk0
lv00
hdisk0
lv00
Case A
hdisk1
hdisk0
lv00
hdisk1
hdisk0
lv00
Case B
hdisk1
hdisk0
lv00
hdisk1



In these examples, you see how the VGDA is used to monitor disks and logical volumes that belong to a volume group. In Case A, the volume group, samplevg, contains only hdisk0. The example also shows how the VGDA holds the map for the logical volume lv00. There is no VGDA represented on hdisk1 since it is not part of the samplevg. However, it may be the case that hdisk1 contains some long-forgotten VGDA from a previous volume group. But since the VGDA for samplevg has no record of hdisk1 belonging to the volume group, hdisk1 is ignored in samplevg's eyes.

Case B is an example of the aftermath of an extendvg of samplevg into hdisk1. A copy of the VGDA from hdisk0 is placed onto hdisk1. But, notice that the VGDA has also been updated on hdisk0 to reflect the inclusion of a new disk into the volume group. Although lv00 does not exist on hdisk1, the VGDA on hdisk1 will have a record of the lv00 mapping sitting on another disk since it's part of the same volume group.

Case C shows the result of the creation of two logical volumes, lv01 and lv02, onto hdisk1. This is to show how the VGDA sitting on hdisk0 is updated with the activity that is being conducted on another disk in the volume group.

One critical point from this information is that since the VGDA on each disk in a volume group “knows its neighbors’ business”, then sometimes this information about neighboring disks can be used to recover logical volumes residing on other the other drives. This usually happens works if the VGDA on a disk is missing or corrupted. However, this is invalid in the cases where the disk is truly dead or if the volume group is only made up of one disk.

Logical Volumes

- Members of specific volume groups
- Can control location of logical volumes
- Can view location of logical volumes
- Non-typical logical volumes
- Logical Volume Control Block
- LVID

Notes: Logical Volumes

Notes:

Members of specific volume groups

Logical volumes that reside on a volume group and only exist in that volume group. Logical volumes are not allowed to be mirrored to other volume groups and cannot be spanned from one volume group to another volume group. This is in part to the information held on a logical volume by the VGDA. The VGDA can't be expected to not only track a logical volume in its own volume group and then additionally track the logical volume (and more important its status) on another volume group.

Can control location of logical volumes

The creation of the logical volume allows many levels of control, from no control to disk specific control to partition location control. The main binary which determines how the partitions are allocated to the user's request is /usr/sbin/allocp. The binary takes into account user specific requirements at the command line and the default values, such as the intra-policy and the inter-policy. With no specific arguments, such as the disk(s) where the logical volumes "should" be created on, allocp will just use the available defaults as the blueprint to determine how the logical volume is allocated. If the user gives a list of disks where they would like the logical volume to be allocated, allocp will create the logical volume with those disks as its parameters. If it cannot fulfill these disk requirements, then the "mklv" command will fail. Finally, with map files, the user can tell the logical volume which exact physical partition on the disk they wish the logical volume to be created. Implicit in this fact is the ability to control the ORDER which these physical partitions are allocated. People with their own theories of optimal data access tend to try to control the logical volume formation at the physical partition level.

Can view location of logical volumes

The LVM commands provide many ways of letting the user view the same logical volume. They can usually just be simply deduced, after some experience in using LVM. For instance, if I want to look at how a logical volume is laid out onto one disk, I simply type: "lspv -M hdiskA | grep lvX". Or, I can type "lslv -p lvX". Or, I can use "lslv -m lvX". Or, I can use "lsvg -l vgname | grep lvX". The point is that there is more than one way the logical volume's location can be viewed by the many LVM high level commands.

Non-typical logical volumes

There are a few logical volumes that exist on the system that are used and accessed in the typical manner. Besides the log logical volume (used by jfs), there are the dump device logical volume, the boot logical volume, and the paging device logical volume. In AIX 4.1, note that the paging device and the dump device are the same logical volume. This was done as part of an effort to “lighten” the system and makes sense. Not only do you free up disk space formally used by the dump logical volume (which you hoped never had to be used), but you’ve also now guaranteed that the dump device is large enough to capture the dump image if you do have a system failure. However, there is a side-effect to this change. When the user tries moving their original paging device (hd6), they cannot do so even though they have correctly turned off their paging device and rebooted the machine. The reason is that the dump device has a lock on the logical volume that prevents the removal of the dormant paging device. This is easily fixed by resetting the dump device with the “sysdumpdev” command.

Logical Volume Control Block

The logical volume control block (lvcb) consists of the first 512 bytes of a logical volume. This area holds important information such as the creation date of the logical volume, information about mirrored copies, and possible mount points in a journaled filesystem.

LVID

The LVID is the soft serial number used to represent the logical volume to the LVM libraries and low level commands. The LVID is created from the VGID of the volume group, a decimal point, and a number which represents the order which the logical volume was created on the volume group.

Mirrors

- Every one is a copy
- Sequential vs. Parallel
- What good is Sequential Mirroring?
- Staleness of Mirrors

Notes: Mirrors

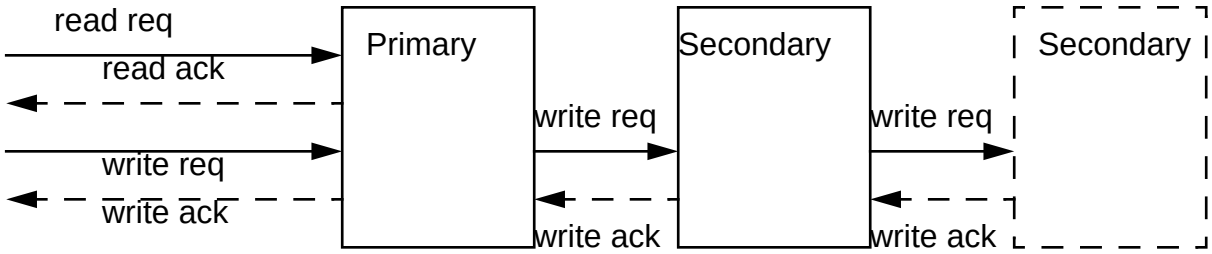
Notes:

Every one is a copy

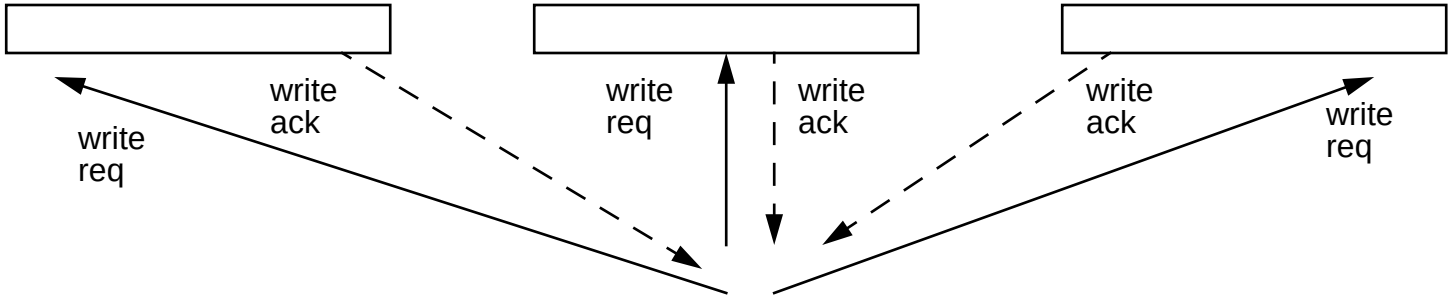
When discussing mirrors in LVM, it is easier to refer to each copy, regardless of when it was created, as a copy. the exception to this is when one discusses Sequential mirroring. In Sequential mirroring, there is a distinct PRIMARY copy and SECONDARY copies. However, the majority of mirrors created on AIX systems are of the Parallel type. In Parallel mode, there is no PRIMARY or SECONDARY mirror. All copies in a mirrored set are just referred to as copy, regardless of which one was created first. Since the user can remove any copy from any disk, at any time, there can be no ordering of copies.

Sequential vs. Parallel

Sequential mirroring is based on the concept of an order within mirrors. All read and write requests first go through a PRIMARY copy which services the request. If the request is a write, then the write request is propagated sequentially to the SECONDARY drives. Once the secondary drives have serviced the same write request, then the LVM device driver will consider the write request complete.



In Parallel mirroring, all copies are of equal ordering. Thus, when a read request arrives to the LVM, there is no first or favorite copy that is accessed for the read. A search is done on the request queues for the drives which contain the mirror physical partition that is required. The drive that has the fewest requests is picked as the disk drive which will service the read request. On write requests, the LVM driver will broadcast to all drives which have a copy of the physical partition that needs updating. Only when all write requests return will the write be considered complete and the write-complete message will be returned to the calling program.



What Good is Sequential Mirroring?

It becomes obvious that there is a definite performance advantage to use parallel mirroring over sequential mirroring. The question arises why anyone would be interested in using Sequential Mirroring. The answer is that since there is a definite and defined Primary copy, this will be the first disk always read during a reboot. Thus, it will be the “source” mirror copy used when mirror resync occurs. This implication will become clear in the section on Mirror Write Consistency Check.

Staleness of Mirrors

The idea of a mirror is to provide an alternate, physical copy of information. If one of the copies has become unavailable, usually due to disk failure, then we refer to that copy of the mirror as going “stale”. Staleness is determined by the LVM device driver when a request to the disk device driver returns with a certain type of error. When this occurs, the LVM device driver notifies the VGSA of a disk that a particular physical partition on that disk is stale. This information will prevent further read or writes from being issued to physical partitions defined as stale by the VGSA of that disk. Additionally, when the disk once again becomes available (suppose it had been turned off accidentally), the

synchronization code knows which exact physical partitions must be updated, instead of defaulting to the update of the entire disk. Certain High Level commands will display the physical partitions and their stale condition so that the user can realize which disks may be experiencing a physical failure.

What Does LVM Give You?

- Flexibility
- Data Integrity
- Speed

Flexibility

- Real-time Volume Group and Logical Volume expansion/ deletion
- Use of Logical Volume under filesystem
- Use of Logical Volume as raw data storage
- User customized logical volumes
- Ability to customize data integrity check

Notes: Flexibility

Notes:

Real-time Volume Group and Logical Volume expansion/deletion

Typical UNIX operating systems have static filesystems that require the archiving, deletion, and recreation of larger filesystems in order for an existing filesystem to expand. LVM allows the user to add disks to the system without bringing the system down and allows the real-time expansion of the filesystem through the use of the logical volume. All filesystems exist on top of logical volumes. However, logical volumes can exist without the presence of a filesystem. When a filesystem is created, the system first creates a logical volume, then places the journaled filesystem (jfs) “layer” on top of that logical volume. When a filesystem is expanded, the logical volume associated with that filesystem is first “grown”, then the jfs is ”stretched” to match the grown filesystem.

Use of Logical Volume under a filesystem

The logical volume is a logical to physical entity which allows the mapping of data. The jfs maps files defined in its filesystem in its own logical way and then translates file actions to a logical request. This logical request is sent to the LVM device driver which converts this logical request into a physical request. When the LVM device driver sends this physical request to the disk device driver, it is further translated into another physical mapping. At this level, LVM does not care about where the data is truly located on the disk platter. But with this logical to physical abstraction, LVM provides for the easy expansion of a filesystem, ease in mirroring data for a filesystem, and the performance improvement of file access in certain LVM configurations.

Use of Logical Volumes as raw data storage

As stated before, the logical volume can run without the existence of the jfs filesystem to hold data. Typically, database programs use the “raw” logical volume as a data “device” or “disk”. They use the LVM logical volumes (rather than the raw disk itself) because LVM allows them to control which disks the data resides, allows the flexibility to add disks and “grow” the logical volume, and gives data integrity with the mirroring of the data via the logical volume mirroring capability.

User customized logical volumes

The user can create logical volumes, using a map file, that will allow them to specify the exact disk(s) the logical volume will inhabit and the exact order on the disk(s) that the logical volume will be created in. This ability allows the user to tune the creation of their logical volumes for performance cases.

Ability to customize data integrity checks

The user has the ability to control which levels of data integrity checks are placed in the LVM code in order to tune the system performance. The user can change the mirror write consistency check, create mirroring, and change the requirement for quorum in a volume group.

Data Integrity

- Mirrors
- Mirror Write Consistency Check
- Ability to detect staleness and to correct
- Quorum checking
- Data relocation
- Write Verify

Notes: Integrity

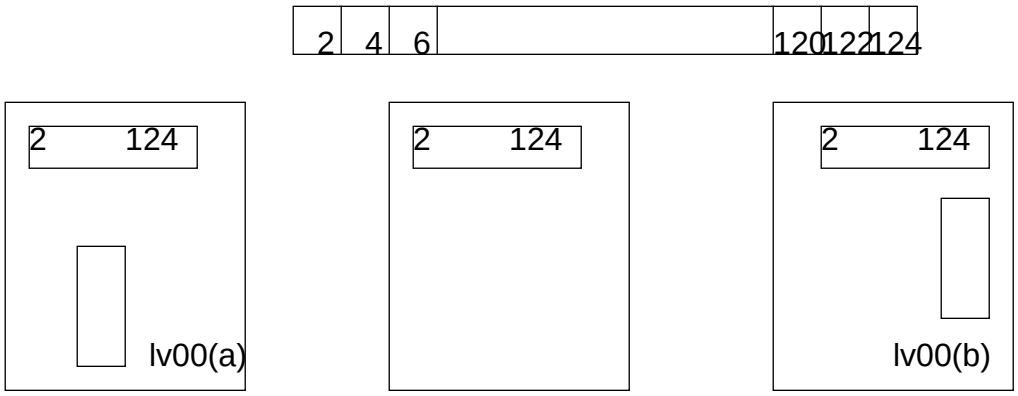
Notes:

Mirrors

AIX allows up to three copies of a logical volume and the copies may be in sequential or parallel arrangements. Mirrors improve the data integrity of a system by providing more than one source of identical data. With multiple copies of a logical volume, if one copy cannot provide the data, one or two secondary copies may be accessed to provided the desired data.

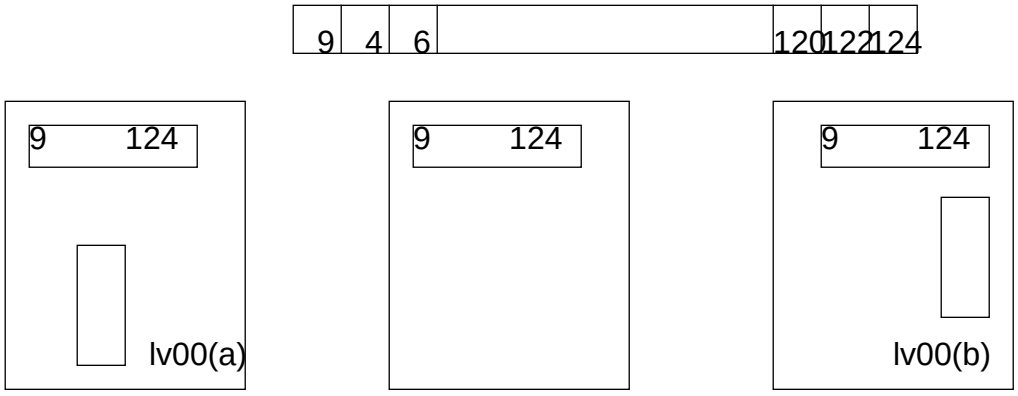
Mirror Write Consistency Check

Mirror Write Consistency Check (MWCC) is a method of tracking the last 62 writes to a mirrored logical volume. If the AIX system crashes, upon reboot the last 62 Logical Track Group (LTG) writes to mirrors are examined and one of the mirrors is used as a “source” to synchronize the mirrors (based on the last 62 disk locations that were written). The LTG is always of size 128 KBytes. This “source” is of importance to parallel mirrored systems. In sequentially mirrored systems, the “source” is always picked to be the Primary disk. If that disk fails to respond, the next disk in the sequential ordering will be picked as the “source” copy. There is a chance that the mirror picked as “source” to correct the other mirrors was not the one that received the latest write before the system crashed. Thus, the write that may have completed on one copy and incomplete on another mirror would be lost. AIX does not guarantee that the absolute, latest write request completed before a crash will be there after the system reboots. But, AIX will guarantee that the parallel mirrors will be consistent with each other. If the mirrors are consistent with each other, then the user will be able to realize which writes were considered successful before the system crashed and which writes will be retried. The point here is not data accuracy, but data consistency. The use of the Primary mirror copy as the source disk is the basic reason that sequential mirroring is offered. Not only is data consistency guaranteed with MWCC, but the use of the Primary mirror as the source disk increases the chance that all the copies have the latest write that occured before the mirrored system crashed. Some usrs turn off MWCC for increased write performance or the software product they use requires that MWCC be turned off. In this case, where MWCC can’t guarantee mirror consistency the user should run a forced volume group sync (syncvg -f <vgname>) immediately after coming up from a system crash.



Before write to mirrored logical volume

In this situation, only even numbered Logical Track Groups have been the last 62 writes to the mirrored logical volume. All disks in the volume group track this MWCC table, regardless of if they contain the mirrored logical volume or not.



After write to mirrored logical volume

A write to an odd Logical Track Group (9) causes the oldest entry in the MWCC (2) to be deleted and the entry to be entered (in RAM). Then, the MWCC table must be written to all disks in the volume group.

Note: This replacement and update only occurs if the new LTG number is not already one of the last 62 entries into the MWCC table.

Ability to detect stale mirror copies and correct

The Volume Group Status Area (VGSA) tracks the status of 1016 physical partitions per disk per volume group. During a read or write, if the LVM device driver detects that there was a failure in fulfilling a request, the VGSA will note the physical partition(s) that failed and mark that partition(s) “stale”. When a partition is marked stale, this is logged by AIX error logging and the LVM device driver will know not to send further partition data requests to that stale partition. This saves wasted time in sending i/o requests to a partition that most likely will not respond. And when this physical problem is corrected, the VGSA will tell the mirror synchronization code which partitions need to be updated to have the mirrors contain the same data.

Note: There is a distinct difference between a MWCC sync and a stale partition sync. If the operating system crashes in the middle of many writes, it has no time to mark a partition as “stale”. This is where MWCC comes in; it assumes that last 62 LTG writes were “stale” and performs a sync, even though the partitions where the LTGs exist are not marked stale by the VGSA.

Quorum checking

Quorum checking is the voting that goes on between disks in a volume group to see if a majority of disks exist to form a quorum that will allow the disks in a volume group to become and stay activated. LVM runs many of its commands and strategies based on the having the most current copy of some data. Thus, it needs a method to compare data on two or more disks and figure out which one contains the most current information. This need gives rise to the need of a quorum. If not enough quorums can be found during a varyonvg command, the volume group will not varyon. Additionally, if a disk dies during normal operation and the loss of the disk causes volume group quorum to be lost, then the volume group will notify the user that it is ceasing to allow any more disk i/o to the remaining disks and enforces this by performing a self varyoffvg. The reasoning behind this action is that if the disks have an unreliable state, then the sooner less data is allowed to be written to the disk, the fewer recovery steps must be performed once the error is detected. In the case where a logical volume may cross disks in multiple points, the filesystem would be put into a partially active state, which could lead to data corruption. However, the user can turn off this quorum check and its actions by telling LVM that it always wants to varyon or stay up regardless of the dependability of the system. Or, the user can force the varyon of a volume group that doesn’t have quorum. At this point, the user is responsible for any strange behavior from that volume group.

Data Relocation

There are three types of data relocation: internal to the disk, hardware relocate ordered by LVM, and software relocation. Relocation typically occurs when the system fails to perform a read or a write due to physical problems with the disk platter. In some cases, the data i/o request completes, but with warnings. Depending on the type of recovered error, we may be so wary of the success of the next request to that physical location that LVM orders a relocation to be on the safe side. The lowest logical layer of relocation is the one that is internal to the disk. These types of relocations are typical private to the disk and there is no notification to the user that a relocation occurred. The next level up in terms of relocation complexity is a hardware relocation called for by the LVM device driver. This type of relocation will instruct the disk to relocate the data on one physical partition to another portion (reserved) of the disk. The disk takes the data in physical location A and copies it to a reserved portion of the disk, location B. However, after this is complete the LVM device driver will continue to reference physical location A, with the understanding that the disk itself will handle the true i/o to the real location B. The top layer of data relocation is the “soft” relocation handled by the LVM device driver. In this case, the LVM device driver maintains a bad block directory and whenever it receives a request to access a logical location A, the LVM device driver will look up the bad block table and translate it to actually send the request to the disk drive at physical location B. As a default, AIX will not allow the performance of the intermediate level hardware relocation on non-IBM drives. On IBM drives, LVM knows that we’re guaranteed the existence of reserved areas for possible hardware relocation (typically 256 blocks per disk). However, on oem drives, IBM cannot predict what spare blocks may or may not exist for possible relocation. So assuming the worst, LVM only allows the internal disk relocation and the top layer software relocation to occur for oem drives.

Write Verify

There is a capability in LVM to specify that you wish an extra level of data integrity is assured every time you write data to the disk. This is the ability known as write verify. This capability is given to each logical volume in a volume group. When you have write verify enabled, every write to a physical portion of a disk that’s part of a logical volume causes the disk device driver to issue the Write and Verify scsi command to the disk. This means that after each write, the disk will reread the data and do an IOCC parity check on the data to see if what the platter wrote exactly matched what the write request buffer contained. This type of extra check understandably adds more time to the completion length of a write request, but it adds to the integrity of the system.

Speed

- Mirrors
- Striping

Notes: Speed

Notes:

Mirrors

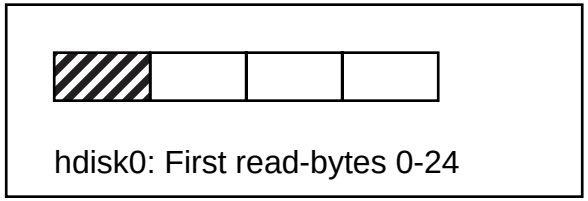
Disk mirroring can improve the read performance of a system, but at a cost to the write performance. Of the two mirroring strategies, parallel and sequential, parallel is the better of the two in terms of disk i/o. In parallel mirroring, when a read request is received, the lvm device driver looks at the queued requests (read and write) and finds the disk with the least number of requests waiting to execute. This is a change from AIX 3.2, where a complex algorithm tried to approximate the disk that would be “closest” to the required data (regardless of how many jobs it had queued up). In AIX 4.1, it was decided that this complex algorithm did not significantly improve the i/o behavior of mirroring and so the complex logic was scrapped. The user can see how this new strategy of finding the shortest wait line would improve the read time. And with mirroring, two independent requests to two different locations can be issued at the same time without causing disk contention, because the requests will be issued to two independent disks. However, with the improvement to the read request as a result of disk mirroring and the multiple identical sources of reads, the LVM disk driver must now perform more writes in order to complete the write request. With mirroring, all disks that make up a mirror are issued write commands which each disk must complete before the LVM device driver considers a write request as complete.

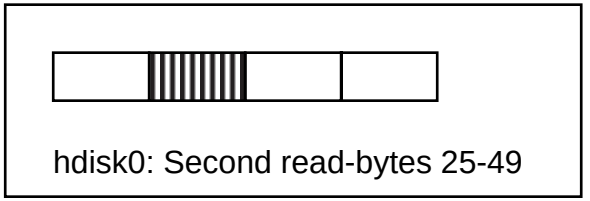
Disk Striping

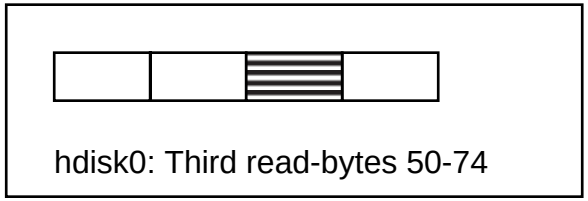
Disk striping is the concept of spreading sequential data across more than one disk to improve disk i/o. The theory is that if you have data that is close to each other, and if you can divide the request into more than one disk i/o, you will reduce the time it takes to get the entire piece of data. This request must be done so it is transparent to the user. The user doesn’t know which pieces of the data reside on which disk and does not see the data until all the disk i/o has completed (in the case of a read) and the data has been reassembled for the user. Since LVM has the concept of a logical to physical mapping already built into its design, the concept of disk striping is an easy evolution. Striping is broken down into the “width” of a stripe and the “stripe length”. The width is how many disks the sequential data should lay across. The stripe length is how many sequential bytes reside on one disk before the data jumps to another disk to continue the sequential information path.

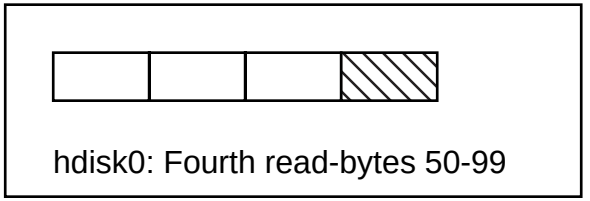
We present an example to show the benefit of striping: A piece of data that is stored of the disk is 100 bytes. The physical cache of the system is only 25 bytes. Thus, it takes 4 read requests to the same disk to complete the reading of 100 bytes:

As you can see, since the data is on the same disk, four sequential reads must be required. If this logical volume were









created with a stripe width of 4 (how many disks) and a stripe size of 25 (how many consecutive bytes before going to the next disk), then you would see:

Logical Volume Manager Limitations

Contents

1. 1016 Physical Partitions Per Disk in a Volume Group
2. Volume Group Descriptor Area
3. Logical Volume Control Block
4. Maximum Size of Logical Volume and Mirrors

1016 physical partitions per disk in a Volume Group

In the design of LVM, each Logical Partition maps to one Physical Partition. And, each Physical Partition maps to a number of disk sectors. The design of LVM limits the number of Physical Partitions that LVM can track per disk to 1016. In most cases, not all the possible 1016 tracking partitions are used by a disk. The default size of each Physical Partition during a “mkvg” command is 4 MB, which implies that individual disks up to 4 GB can be included in a volume group (using the default values).

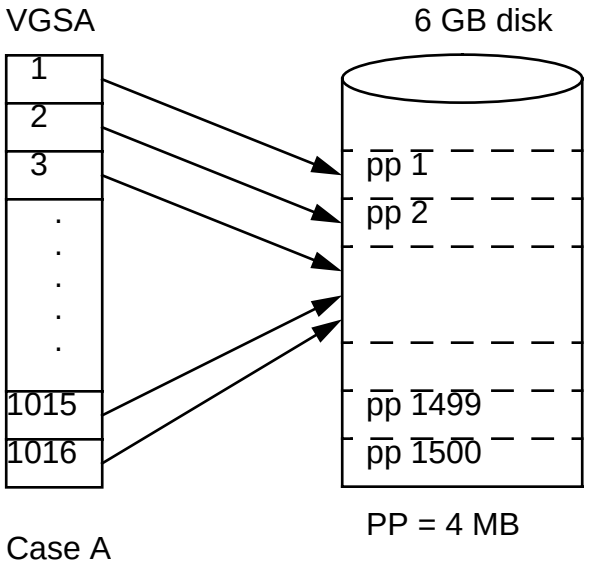
If a disk larger than 4 Gb is added to a volume group (based on usage of the 4 MB size for Physical Partition) the disk addition will fail with a warning message that the Physical Partition size needs to be increased. There are two instances where this limitation will be enforced. The first case is when the user tries to use “mkvg” to create a volume group where the number of physical partitions on one of the disks in the volume group would exceed 1016. In this case, the user must pick from the available Physical Partition ranges of:

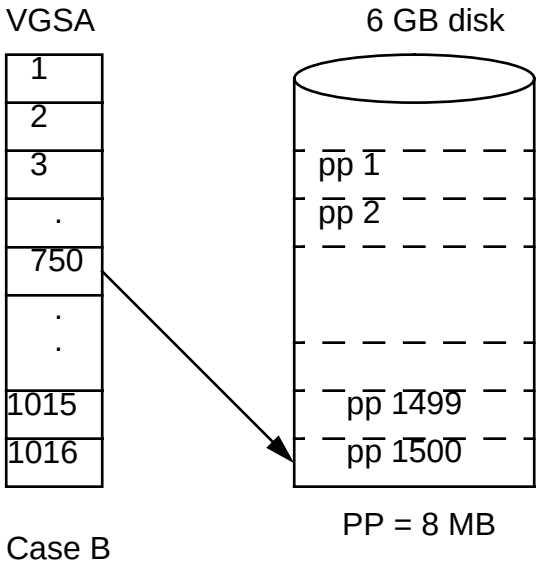
1, 2, (4), 8, 16, 32, 64, 128, 256

Megabytes and use the “-s” option to “mkvg”. The second case is where the disk which violates the 1016 limitation is attempting to join a pre-existing volume group with the “extendvg” command. The user can either recreate the volume group with a larger Physical Partition size (which will allow the new disk to work with the 1016 limitation) or the user can create a stand-alone volume group (consisting of a larger Physical Partition size) for the new disk.

In AIX 4.1 and 3.2.5, if the install code detects that the rootvg drive is larger than 4 GB, it will change the “mkvg -s” value until the entire disk capacity can be mapped to the available 1016 tracks. This install change also implies that all other disks added to rootvg, regardless of size, will also be defined at that new Physical Partition size.

For RAID systems, the /dev/hdiskX name used by LVM in AIX may really consist of many non-4GB disks. In this case, the 1016 limitation still exists. LVM is unaware of the size of the indivicual disks that may really make up /dev/hdiskX. LVM bases the 1016 limitation on the AIX recognized size of /dev/hdiskX, not the real, independent physical drives that make up /dev/hdiskX.





These two cases show the problems and remedies of the 1016 PP case:

Case A shows what would happen if the 1016 PP limitation were not enforced. In the 6 GB disk, all physical partitions starting with partition 1017 would no longer be tracked by the VGSA. This is rather important because it implies that at least one-third of the disks would not be correctly monitored by LVM logic.

Case B shows the corrected case and how, by changing the PP size to 8 MB one can have more than enough VGSA bits to monitor all partitions in the 6 GB disk.

The fixes for this problem are:

Fixes for attempts to install and create/re-install mksysb images: ix46862 and ix46863

Fixes to prevent mkvg and extendvg from working if in violation of 1016: ix48926 and ix51754

Note: In these fixes, if this error is discovered the error messages will not be output to a SMIT panel if the mkvg and extendvg commands are executed from within SMIT.

Common Questions and Concerns about this 1016 Limitation

A) What are the symptoms of problems seen when in violation of 1016?

If you are in violation of 1016, you may possibly get a false report of a non-mirrored logical volume being “stale” (which is an oxymoron). Or worse, you may get a false indication that one of your mirrored copies has gone stale. Next, migratepv may fail because migratepv briefly uses mirroring to move a logical volume partition from one disk to another. Basic to LVM is philosophy that LVM will never allow a user to remove a mirror if the mirror left behind is “stale”. Thus in migratepv,

if the target logical partition is “stale” (due to misinformation from a 1016 violation), then the removal of the original logical partition will be prevented and the migratepv will fail. This will be also true for the reorgvg and the reducelv comands.

B) How safe is my data? What if I never use mirroring or migratepv or reorgvg?

The data is as safe (in your mind) as the day before you were informed about the 1016 problem. The only case where data may be lost is if someone is mirroring a logical volume and ALL copies go bad (whether true hardware error or imagined 1016 error) at the same time. If you never mirror or use migratepv or use reorgvg, then this issue shouldn’t concern you. But, it might be unwise to state that you’ll never use either of those options.

C) Can I move this volume group between RS/6000 systems and various versions of AIX?

Yes and No. You can move a volume group, that is in violation of 1016, to another machine or level of AIX and still use that volume group. However, the ability to extendvg the volume group depends on the software loaded on that machine.

D) Will this limitation be changed?

No. First, the ability to current ability for the user to create a volume group of different PP sizes assures that regardless of the size of future disks, one will always be able to accommodate any disks within the 1016 limitation. Second, if a change to the limitation were made, then a huge incompatability problem would be introduced to LVM.

E) Will I have to rebuild the volume group?

Yes, the fixes mentioned in this chapter only PREVENT the future creation and/or extensions of disks into a volume group. Disks that are already in violation of the 1016 limit must be recreated at larger Physical Partition sizes.

Volume Group Descriptor Area

Notes: Volume Groups

Volume Group Descriptor Area

The VGDA is an area at the front of each disk which contains information about the volume group, the logical volumes that reside on the volume group and disks that make up the volume group. For each disk in a volume group, there exists a VGDA concerning that volume group. This VGDA area is also used in quorum voting. The VGDA contains information about what other disks make up the volume group. This information is what allows the user to just specify one of the disks in the volume group when they are using the “importvg” command to import a volume group into an AIX system. The importvg will go to that disk, read the VGDA and find out what other disks (by PVID) make up the volume group and automatically import those disks into the system (and its) ODM as well. The information about neighboring disks can sometimes be useful in data recovery. For the logical volumes that exist on that disk, the VGDA gives information about that logical volume so anytime some change is done to the status of the logical volume (creation, extension, or deletion), then the VGDA on that disk and the others in the volume group must be updated.

In some instances, the user will experience a problem adding a new disk to an existing volume group or in the creation of a new volume group. The warning message provided by LVM will be:

Not enough descriptor space left in this volume group. Either try adding a smaller PV or use another volume group

When the user creates a volume group, the “mkvg” command defaults to allowing the new volume group to have a maximum of 32 disks in a volume group. However, as bigger disks have become more prevalent, this 32 disk limit is usually not achieved because the space in the VGDA is used up faster, as it accounts for the capacity of the bigger disks. This maximum VGDA space, for 32 disks, is a fixed size which is part of the LVM design. Large disks require more management mapping space in the VGDA, which causes the number and size of available disks to be added to the existing volume group to shrink. When a disk is added to a volume group, not only does the new disk get a copy of the updated VGDA, but as mentioned before, all existing drives in the volume group must be able to accept the new, updated VGDA.

The exception to this description of the maximum VGDA is rootvg. There are two reasons that a special rule is provided for rootvg. First in order to provide AIX users more free disk space, when rootvg is created, “mkvg” does not use the maximum limit of 32 disks that are allowed into a volume group. Instead in AIX 3.2, the number of disks picked in the install menu of AIX is used as the reference number by “mkvg -d” during the creation of rootvg. Second, the VGDA structure is kept in memory during the operation of a volume group. By keeping the VGDA size small for rootvg, then the

AIX operating system can install on a machine with a small amount of RAM. Less of the RAM is used up tracking a large VGDA space that may never all be used. For AIX 4.1, this “-d” number is 7 for one disk and one more for each additional disk picked. i.e. you pick two disks, the number is 8. You pick three disks, the number is 9, and so on....This limit does not mean the user cannot add more disks to rootvg in the post-install phase. The amount of free space left in a VGDA, and thus the number of size of the disks added to a volume group, depends on the size and number of disks already defined for a volume group. However, this smaller size during rootvg creation implies the user will be able to add fewer disks to rootvg than compared to a non-rootvg volume group.

Logical Volume Control Block

Logical Volume Control Block

The logical volume control block (lvcb) consists of the first 512 bytes of a logical volume. This area holds important information such as the creation date of the logical volume, information about mirrored copies, and possible mount points in a journaled filesystem. Certain LVM commands are required to update the lvcb, as part of completeness algorithms in LVM. The old lvcb area is first read and analyzed to see if it is a valid lvcb. If the information is verified as valid lvcb information, then the lvcb is updated. If the information is not completely valid, then the lvcb update is not performed and the user is given the warning message:

Warning, cannot write lv control block data

Most of the time, this is a result of database programs accessing the raw logical volumes (and thus bypassing the journaled filesystem) as storage media. When this occurs, the information for the database is literally written over the lvcb. Although this may seem fatal, it is not the case. Once the lvcb has been overwritten, the user can still:

1) Extend a logical volume

2) Create mirrored copies of a logical volume

3) Remove the logical volume

4) Create a journaled filesystem with which to mount the logical volume (note that this will destroy any data sitting in the lvcb area)

However, there is a limitation caused by this deletion of the lvcb. The logical volumes with deleted lvcb's face possible, incomplete importation into other AIX systems. During an "importvg", the LVM command will scan the lvcb's of all defined logical volumes in the volume group to be imported. Surprisingly, if the lvcb is deleted, the imported volume group will still define the logical volume to the new AIX system which is accessing this volume group, and the user can still access the raw logical volume. However, any journaled filesystem information is lost and the logical volume and its associated mount point won't be imported into the new AIX system. The user must create new mount points and the availability of previous data stored in the filesystem is NOT assured. Additionally, all copies of the logical volume (mirrors), except the original

copy, are put into an unreliable state. The mirrors are physically there, but some of the lvm commands which depend on their information from the lvcb will not realize that the mirrors still exist. Finally, with an erased lvcb, the output from the “lslv” command might be misleading or unreliable.

Maximum Size of Logical Volumes and Mirrors

- Theoretical Limits
- Practical Limits
- Mirroring Limit

Notes: Theoretical Limits

The theoretical limits for a logical volume can be calculated by the design limits in LVM:

Number of disks in a volume group: 32

Number of physical partitions per disk in a volume group: 1016

Largest sized Physical Partition in a volume group: 256 MB

Largest logical volume: 32 (disks) * 1016 (max physical partitions per disk) * 256 MB = 8.3 Terabytes (approx)

Notes: Practical Limits

The theoretical limit is not reachble because, although the logical volume could be created, there would be no practical way to access all portions of the logical volume.

There are two ways to access data in a logical volume: direct to the raw device or Journaled FileSystem (jfs).

The limits for these two access methods are listed:

AIX 3.2

jfs: 2 GB

raw logical volume: 1.3 Terabytes

AIX 4.1

jfs: 1.3 Terabytes

raw logical volume: 1.3 Terabytes

The reason that the access is greater, in AIX 3.2, in the raw logical volume rather than jfs is the method of addressing. By going directly to the raw logical volume, the user is using a different block interface method than jfs. The direct block addressing method is 40 bits in the lvm device driver. Thus, a program using the lvm device driver can access 2 to the 40th bits via the device driver, which translates to 1.3 Terabytes.

Logical Volume Manager - Odds and Ends

Things That Might Be Useful to Understand

- How importvg really works
- How exportvg really works
- Difference between exportvg and reducevg
- How LVM works with ODM
- Known problems with LVM and ODM
- What to check for when things don’t work
- Official mirroring of rootvg statement
- Why mirroring of rootvg only supported on 4.1/beyond

How importvg really works

The significant advantage of AIX is the ability to move a set of data disks among AIX systems. By running importvg, the user allows the new AIX system to learn:

1) which disks constitute the data disks

2) what logical volumes exists on the data disks

3) what mount points, if any, exists for the logical volumes

The system will prevent name collisions between the LVM entities existing AIX system and new data disks members attached to the system. Additionally, LVM may create the appropriate filesystem entries and special files to allow the user to mount and use the jfs filesystems immediately. How is this all done? The key to this ability is the VGDA that resides on each disk in a volume group. The VGDA is essentially a “suitcase” of information. You can pack this suitcase up, take it to another AIX system, open it up and let that system learn about the information in that volume group. For this discussion, the key features of the VGDA are:

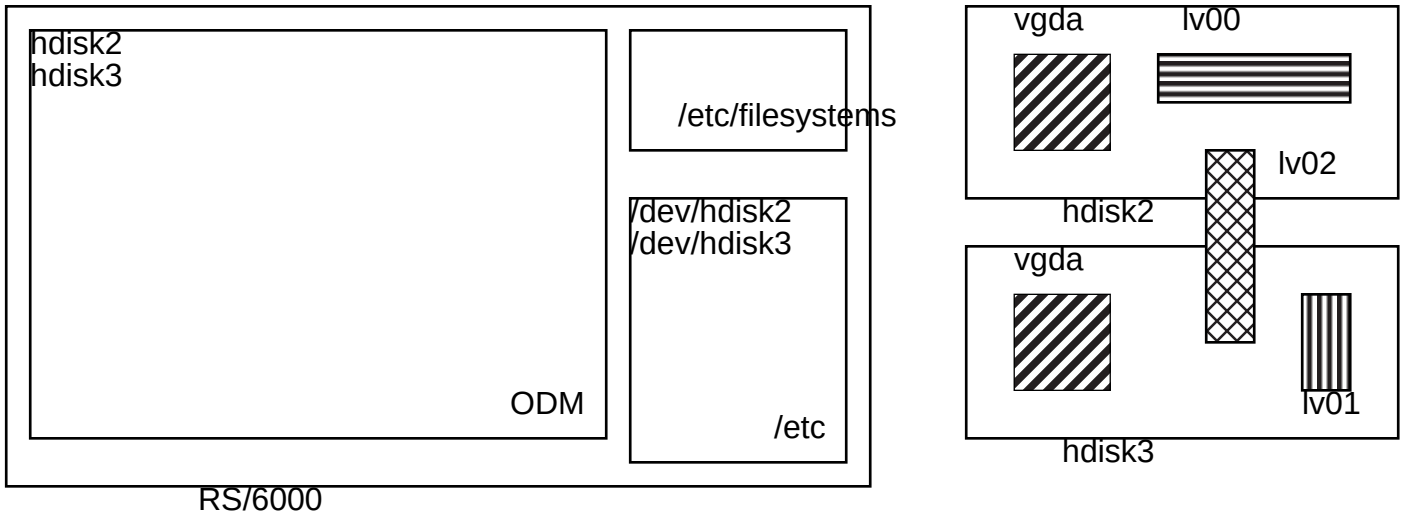
a) lists of all the PVID’s of disks in the volume group

b) list of all the LVIDs in the volume group

c) list of the *preferred* logical volume names associated with the LVIDs in the volume group

d) mount points for the jfs logical volumes in the volume group

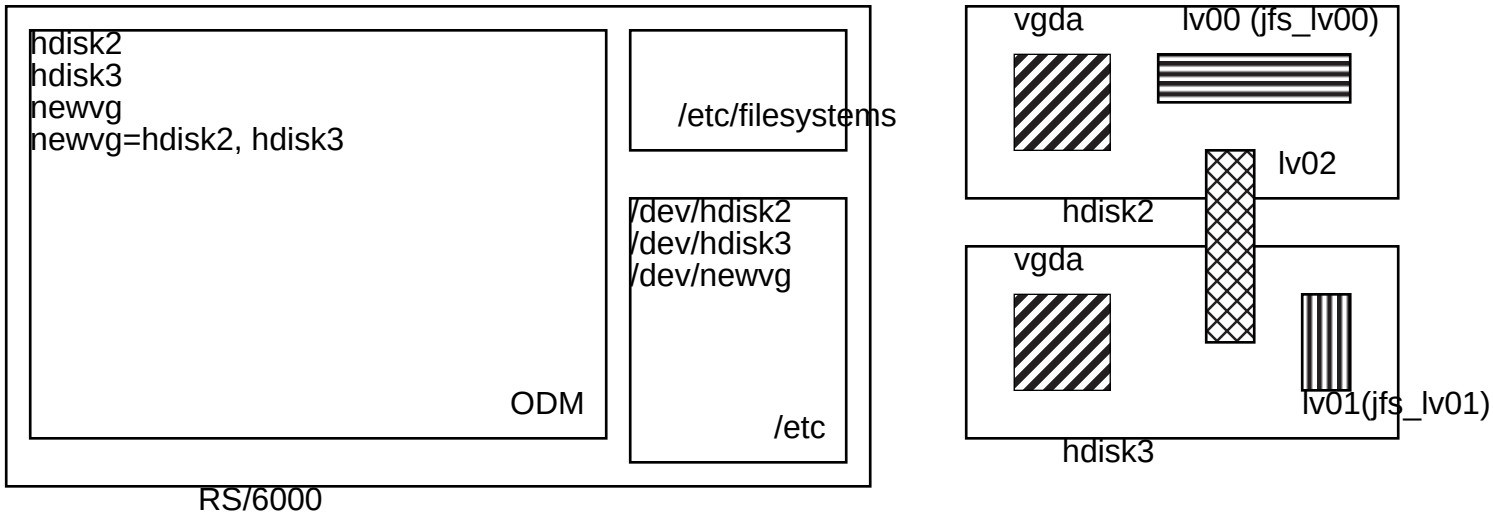
Let’s take a graphical view of an importvg of a volume group of two disks into a new AIX system. Here, we have two disks, hdisk2 and hdisk3 that were part of a volume group on another system. We attach these two disks to an RS/6000.



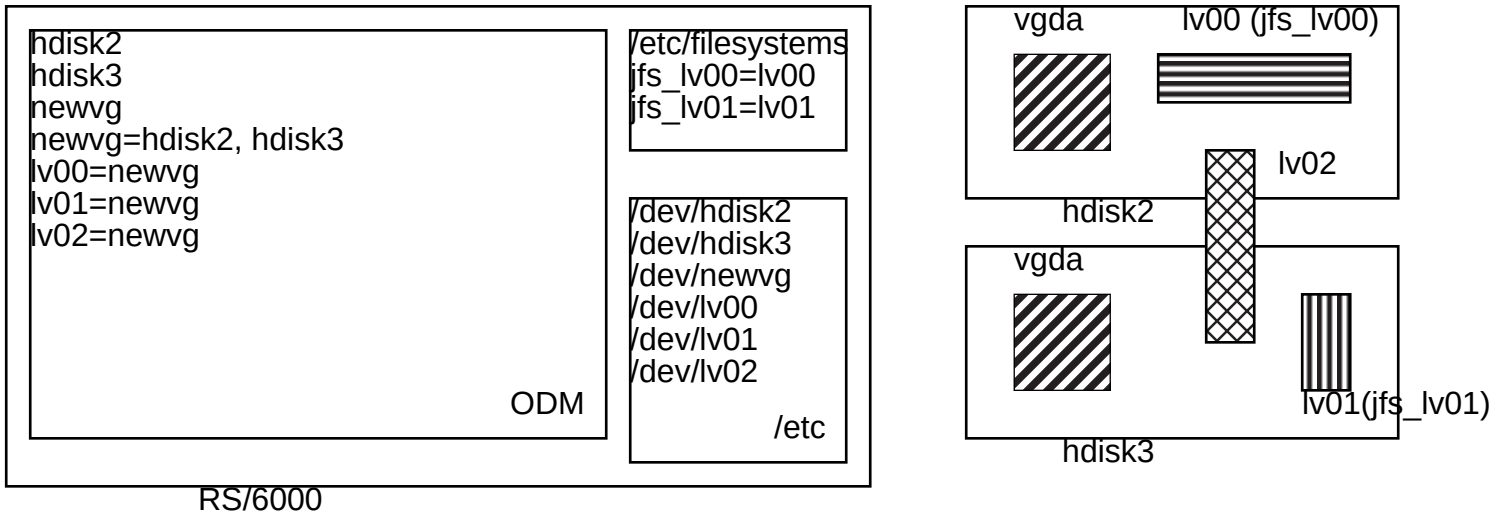
Before anything is done on the system, this is what it looks like graphically. The RS/6000 is aware that there are two new disk drives attached to it, but it doesn't know about the internal contents of these drives. The user then types the command:

importvg -y newvg hdisk2

Notice that the second disk drive, hdisk3, was not included in the command string for importvg. This is okay because a critical part of LVM's VGDA is the ability of each disk in a volume group to know about the existence of all drives that are part of the volume group. The LVM code goes and queries the VGDA of hdisk2 and it discovers that hdisk3 is part of the volume group. Then, importvg will consider any data that is only found on hdisk3 as well as the original disk (hdisk2). When all the information about what disks make up the volume group is collected, LVM enters this information into ODM and /etc. In ODM, the key names of the volume group and disk associations are built. In /etc, the special files for the volume group are built:



After the disk information is read into the ODM and the corresponding special files are created, the same action is performed on the logical volumes associated with the volume group. First, the existence of each logical volume in the volume group is read into the ODM and entry points created in /dev. For each logical volume read into the system, the logical volume is scanned to see if there are mount points associated with a jfs filesystem. If this information is valid, the mount points are created and the entries are placed into /etc/filesystems.



In the following action, there is no filesystems entry for lv02 because lv02 does not have an associated jfs filesystem. Note that there is an important safeguard in this process. If importvg detects that the importing RS/6000 already has a logical volume with the same name as the logical volume it is trying to import into the system, importvg will rename the logical volume without user approval. After LVM has renamed the logical volume in the ODM, it will also update the VGDA and the LVCB on the disks where the renamed logical volume resides. This is an important point as users might be surprised to find out that their logical volumes have been renamed without their express knowledge.

Another case that might occur is when the user imports a raw logical volume that has had its LVCB overwritten with data. This typically occurs in cases where the raw logical volume is used by a database program. When this occurs, the importvg will not be able to fully populate the ODM with information about the logical volume. Typically, the user will see a message in the importvg about it unable to update the Logical Volume Control Block. When the LVCB gets read into a system, a timestamp is placed into the LVCB by the importvg process. But, the logic of LVM first checks to see if the LVCB is valid before updating the LVCB structure. Obviously, if the LVCB is overwritten by data, then the checks for a valid LVCB will fail and the LVCB will not be updated by importvg (and thus saving having importvg corrupt user data). Many users are concerned that having the LVCB unreadable will cause the logical volume to be unusable. That will not be the case. For instance, the VGDA will tell the AIX system that a logical volume is truly mirrored and the lvm device driver will correctly handle mirroring functions. However, the COPIES value within the “lslv” command is derived from the ODM. And since the

ODM couldn’t be updated from a correct read of the LVCB, then the default value of COPIES: 1 is used in the ODM. This does not imply that mirroring isn’t working, it just causes user confusion. At this point, there is no good compromise for this confusion. It is considered better to have confusing output rather than having LVM corrupt potential user data in the raw logical volumes. This limitation has been documented in the AIX 4.2 System Management Guide.

How exportvg really works

The most confusing aspect of a volume group and how it's related to ODM is usually seen with the exportvg command. The key point to the exportvg command is that is mostly a pure ODM related command. It does not affect the physical VGDA that sits on a disk platter. When running exportvg, it first checks to make sure the volume group has been varied off. If the volume group is not varied off first, the exportvg will fail. LVM does not want to "pull the rug out" from under users that could be potentially using the volume group and the volume group's associated ODM entries. When an exportvg is run, the reverse of an importvg is performed. First, exportvg finds all the disks associated with a volume group. Next, it finds all the logical volumes associated with that volume group. After the logical volumes are discovered, the potential mount points associated with the logical volumes are noted. Then in a backward sweep of what importvg performed, exportvg first deletes /etc/filesystems entries associated with logical volumes with filesystems. After this is done, the /dev special files are removed, and then the ODM has all entries associated with the volume group deleted from the ODM database. Note that during this time, there was no action upon the volume group and the disk platters that make up the volume group. Many people mistakenly believe that the physical disks have to be there or that when exportvg is run, the disk information on the disks becomes erased.

One of the most common questions from users is what to do if all the disks that make up a volume group have been removed from an RS/6000, without first running exportvg on the system. The problem is that LVM cannot be made aware if:

a) the disks have died

b) the disks have been moved to another RS/6000

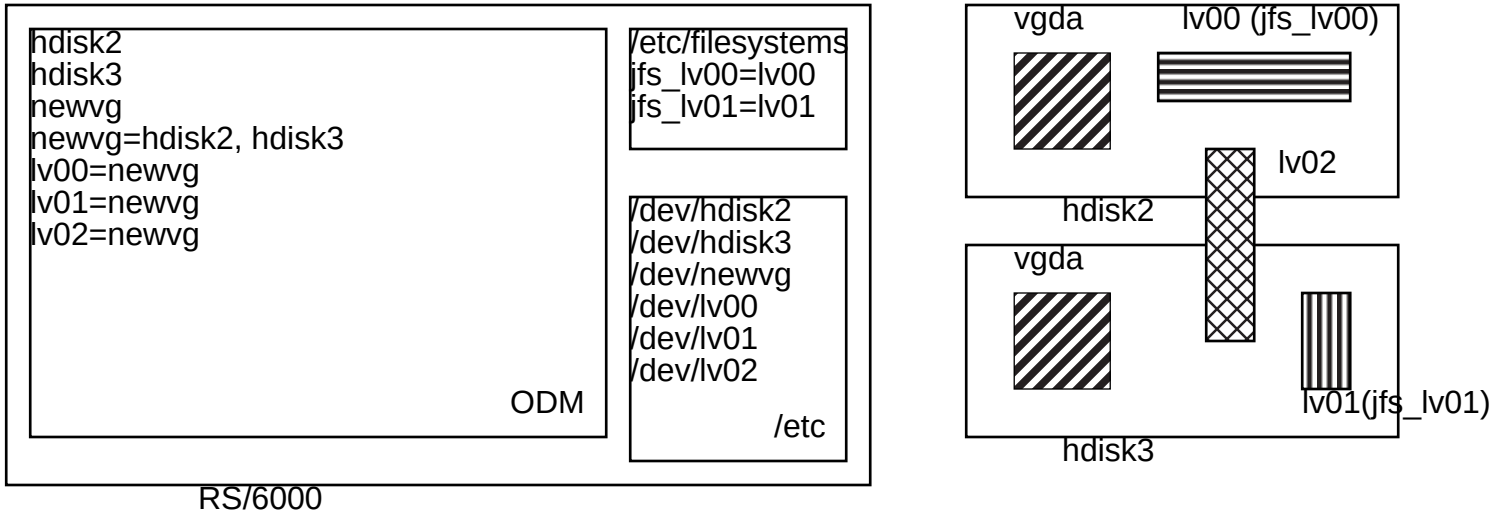
c) the disks have just been temporarily powered off

So the user will still see the volume group listed and maybe during boot, LVM will complain that it couldn't varyonvg the volume group that has been removed. People complain that they don't know how to "shake off" this ghost reference to a volume group the user no longer cares about. As stated previously, LVM doesn't know what has really happened to the volume group. In the case of (a) or (b), the user simply tells LVM that it wants all references to that volume group erased from the system. A simple exportvg should take care of this problem. However, if only one disk in a multi-disk volume

group has been replaced, this is a completely different set of circumstances. In this case, reducevg should be used, but reducevg is very different from exportvg.

Difference between exportvg and reducevg

This section first covers how reducevg works and then explicitly points out the differences between the way exportvg and reducevg behave. reducevg should be considered a type of ODM/VGDA modifier. reducevg is used to edit a disk out of a volume group. Taking a look at the previous example:



Let's assume that you want to take away hdisk3 because it has died or that it is giving you indications that it is going to fail in the near future. You would want the volume group to stay up and provide the existence of lv00 (via hdisk2), while you replace hdisk3. You run the command:

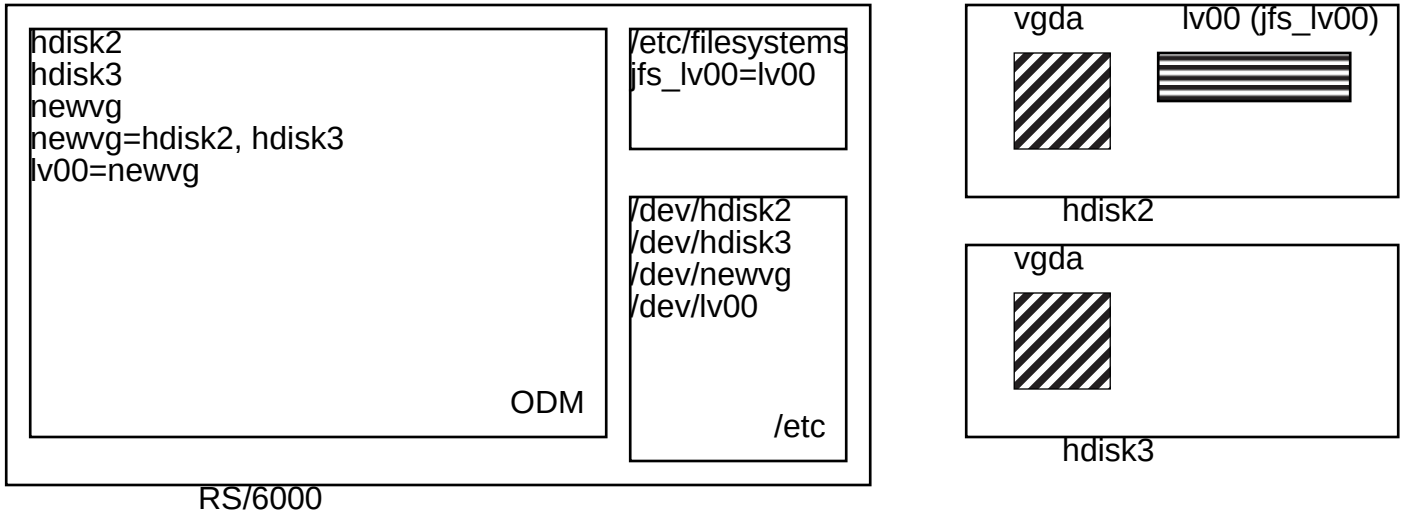
reducevg newvg hdisk3

Which says, "from newvg, take away the memory of hdisk3". The first thing that's going to happen is that reducevg is going to detect the existence of lv01 and lv02 via the ODM and the VGDA of the volume group on hdisk3. Since these logical volumes to potentially hold important user data, reducevg will prompt the user to see if they know that logical

volumes exist on the disk which is to be removed and if they want to remove those logical volumes. If the user says yes, reducevg will go ahead and remove the logical volumes one at a time.

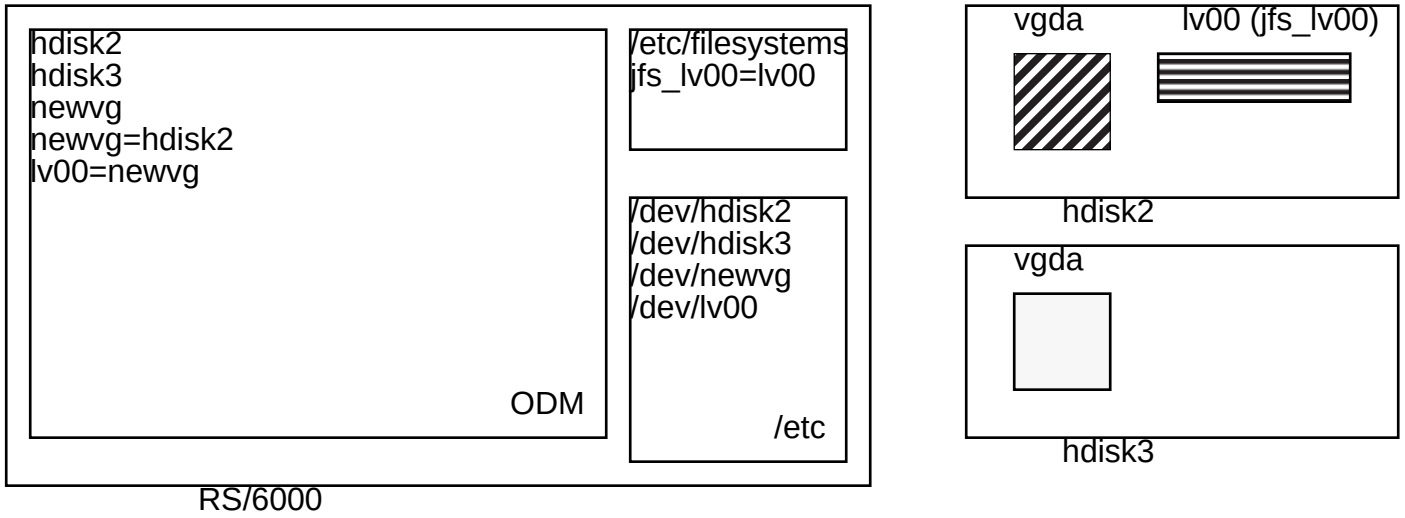
A very important point must be made at this point. Sometimes, people are unaware that a particular logical volume straddles a disk. Or, they don't realize the consequences of removing a logical volume that spans more than one physical disk. In the case where hdisk3 had gone bad, the use would rightly conclude that the data on the portion of lv02 that sits on hdisk2 is not recoverable and that portion can be removed. However, some users think that maybe reducevg will just "cut off" lv02 at the point where it crosses from hdisk2 to hdisk3. This is not the case; the ENTIRE logical volume is removed. There is no point of saving off the "good" portion of the logical volume. With the ability of data to be placed in any order on the logical volume, saving just a portion of the logical volume (that sitting on hdisk2) could lead to data corruption or crashes in the filesystem due to references to addresses that are no longer valid.

So after the logical volumes from hdisk3 have been removed, the example looks like:



Notice that the logical volume information is removed from ODM and that the /etc/filesystems and /dev entries for the corresponding logical volumes have also been removed. And although it's not apparent, the VGDA on both hdisk2 and hdisk3 have been changed to delete any reference or knowledge of lv01 and lv02. After the logical volume level deletions

have been done, the disk level information is changed. First, the ODM entries that tie hdisk3 to newvg are removed. Then, the VGDA on both hdisk2 and hdisk3 are changed. hdisk2's VGDA is edited so it no longer has a knowledge of hdisk3's existence. hdisk3's VGDA is essentially zeroed out so that it has no memory of being part of any volume group. Finally, any type of quorum maintenance is performed to "right" or move the majority vote of quorum from hdisk3 to hdisk2 if it is appropriate to the case. Now, the example is represented:



From this example, notice that the references to hdisk3 as an independent entity still exists. The concept of a disk called hdisk3 is still recognized in the ODM and /dev directories. LVM does not get rid of these references because LVM does not know the intentions of the user who has called reducevg. The user could be detaching hdisk3 from newvg to be used in another volume group or the user may be replacing a bad disk or the user could be using reducevg to remove many logical volumes that happened to be sitting on hdisk3 all at once, without wanting to resort to hand-typing multiple "rmlv" commands. As a side issue, to completely get rid of the hdisk3 references in ODM and /dev, one would type:

rmdev -l hdisk3 -d

Now, an important part of this discussion is what happens if the user were to next run the command:

reducevg newvg hdisk2

When this happens, the same deletion of the logical volume and editing of the remaining VGDA occurs for hdisk2 as it occurred in the example presented for hdisk3. And, the essential effect of this command is that the systems knowledge concerning newvg is erased. But the main point to be made is that the VGDA's on these disks have been zeroed out as part of the reducevg. exportvg does not do this type of VGDA editing. Some people have mistakenly used reducevg to perform what they thought was equivalent to exportvg. And when they then tried to re-importvg to the system or on another system, importvg will fail because there is no valid VGDA to read in. To put it simply, the data is lost and unrecoverable.

The two most often asked questions in LVM are:

"I have this volume group I don't care about, even the disks are gone, but LVM keeps giving me a message about not being able to find the volume group. How do I get rid of this bogus volume group".

"I replaced a disk, but I keep getting this reference to the disk I replaced. How do I get rid of it?"

In the first case, the reason LVM keeps trying to varyonvg a volume group that isn't physically there is that the volume group still exists in the ODM. The default entry for that volume group, in the ODM, tells LVM to automatically try to varyonvg the volume group at system boot. As mentioned before, LVM can't know if the disk are simply powered off, if the disks are involuntarily powered off due to power loss or physical problems, if the disks have failed and can't be read, or the disks were moved to another RS/6000 system. Thus, the user has to expunge the volume group information from ODM so LVM doesn't need to concern itself with this bogus volume group. On the other hand, sometimes these error messages about unable to varyonvg a volume group are the first indication that something may be wrong with their disk systems. This case is simply cleaned up in the ODM and filesystems with the command:

exportvg <vg_name>

The second case involves the fact that the VGDA on each disk in a volume group keeps track not only of itself but the other disks that are members of the volume group. Typically, this problem occurs when users swap out a disk drive before first informing LVM and the VGDA's of the remaining disks. So when your run varyonvg, LVM will inform you that a member of your volume group family is missing and maybe this should be investigated. You can force the memory and references of this missing disk to be edited from the VGDA a force action provided in reducevg:

reducevg -f <vg_name> <PVID of missing disk>

If you can’t remember the PVID of the missing disk, varyonvg will always provide the PVID for you. Simply run varyonvg on the already varied on volume group to recapture the missing PVID.

This command was purposely built for the user to reference the 16 digit hexadecimal PVID of the missing disk. The reason is that if the user replaces the disk improperly, then there might be multiple, duplicate hdisk names floating around in the ODM. However, there should be only one distinct PVID for each disk (regardless of hdisk name) sitting in the ODM.

How LVM Works with ODM

The main LVM commands, such as lsvg, lspv, and lslv work with both the ODM and the VGDA to gather their information. For older versions of AIX, these commands don't work directly with the VGDA, but work with what are commonly called the vg special files. These special files are located in /etc/vg and are distinguished by the vg<VGID> naming method.

In the original 3.1 version of LVM, the idea was that a file copy of the VGDA would be kept in the /etc/vg/vg* files. The commands that are used to describe LVM would then consist of information gathered from ODM and the /etc/vg files. This was thought as a quick way of accessing data without disk i/o to the system. However, two main problems occurred during the use of this design in 3.1 and 3.2. The first problem was that every so often, the information held in /etc/vg/vg* would be out of sync with what was truly held in the VGDA. The second, more common problem, was when /etc would fill up during the creation of a new volume group. Every new non-rootvg volume group caused a file of roughly 1 MB to be created. However, sometimes the /etc directory would be filled up before the entire /etc/vg special file was completely written. This caused an ambiguous state within the volume group. It could be accessed, but certain commands would fail because the vg special file was "chopped off". The only solution to this problem was to exportvg the volume, expand the root directory, and then remake the volume group. But by the time this limitation is realized as the full root directory problem off some other LVM problems, the user already has data sitting in the volume group.

In AIX 4.1, the solution to these two problems was to have the lvm commands reference both the ODM and the VGDA, with the /etc/vg/vg* file used purely for locking/synchronization purposes during configuration. So the lsvg, lspv, and lslv commands now go to the actual VGDA on the platter to gather their information. This design frees up /etc space and guarantees the user the absolute latest data about the volume group and its attributes.

Another aspect of LVM's involvement with the ODM is the ODM object known as the vg-lock. Whenever an LVM modification command is started, the LVM command will go to the vg's entry in ODM and put a vg-lock. This tells other LVM processes that may want to access and change the same volume group. However in some cases, the vg-lock gets left behind and jobs are inadvertently being turned away. There are two solutions to this, the user can unlock the vg with the simple command:

putlvodm -K <vgid>

(this vg was locked in the first place with putlvodm -k <vgid>)

Or, the user can simply varyonvg the volume group. varyonvg always clears any vg locks, which might explain why leftover locks are never seen after a system reboot. And to many people's surprise, varyonvg can be run on a volume group that is already varied on. No harm done here, as long as the volume group is not part of some twin-tailed configuration (then you'd have to take locking conditions from the varyon into account).

Known problems with LVM and ODM

Slow behavior of lsvg, lslv, and lspv

One result of the change in which the ls* commands are fulfilled in AIX 4.1 is that commands themselves take longer. This is an unfortunate side-effect of the switch from reading a database file (/etc/vg/vg*) to reading the disk platter (VGDA). Currently, this can not be easily modified unless some further design and testing is done on the system.

ls* commands breaking the vg lock

One of the original designs of LVM stated that the ls* commands (lsvg, lslv, and lspv) should never be denied access to volume group information. It should list information, even if it is in the middle of a transaction. Thus, the first thing these ls* commands do is break any vg-lock so it may read the information from the ODM and the VGDA. The problem with this lock breaking is that sometimes results of an output have partial or confusing data. But the worse side-effect of this design is that you can allow corruption of ODM volume group information. What could happen is that process A starts a big mklv. Then process B starts an lsvg, which breaks the vg-lock placed by process A. After this, process C starts a mklv and then you have two processes modifying the ODM for the same volume group. This window has been closed in AIX 4.2. When the ls* commands are run, they check to see if the lock is in place. If the vg-lock is in place, the following error message will be displayed:

Warning: Volume group <vgname> is locked. This command will continue retries until lock is free. If lock is inadvertent and needs to be removed, execute varyonvg on active volume group to clear the volume group lock.

So the ls* commands will spin in a loop waiting for the lock to be freed. If the user is tired of waiting, then killing the ls* commands is another option.

getlvodm with multiple processes

Another issue with LVM/ODM contention is when parallel processes are accessing the ODM to consult with the database on creating new names or attributes for LVM. The culprits in this problem in LVM are “getlvodm”, “genmajor” and “genminor”. These commands can be used to verify that a name or number is not already used by some other LVM entity. If it is, then the command provides an alternative. This would most likely occur in the mkvg or mklv command where the user does not provide a preferred name via the “-y” option. Because of timing windows, two independent mklv commands for two distinct volume groups would ask LVM to provide it a LVM/ODM value. Then, two ODM retrieval/generation

commands would be following each other in a visit to the ODM and think each process has found a distinct, new ODM value for the LVM object. The LVM command scripts can be modified so that more synchronization and high-level locking is provided. But the problem is that LVM commands such as getlvodm or the LVM libraries can be used by user programs or scripts that would encounter into the same problem. Until a design change is made for a higher level of locking is provided, users are warned to be careful of self-corrupting the ODM.

What to check for when things don't work

When strange things start happening or things aren't working in LVM, one of the first things people say is, "Oh, ODM is corrupted". This could be true, but it can't be the blame-all for everything. Here are several methods or considerations a user may take into account when trying to diagnose why something is failing:

Where is it failing?

If the user is "lucky" the failing LVM routine is one of the many shell commands. Then, the user can "shell out" the LVM command and see what portion of the command it didn't like. First, this assumes that the user is knowledgeable about shell programming. There is a neat little command called "script" (look in infoexplorer for more details) which can be quite useful in debugging. Here's what one should type on the command line (all entries represent commands, unless in parenthesis):

root# script

root# /bin/sh -x /usr/sbin/<lvm_command_to_debug and associated command options/data>

root# <control-D>

You will now have the failed script output saved into the file "typescript". Examine this file for the failure point to see what the LVM command didn't like. At the very least, this information will most likely be asked by you from the IBM software support personnel, so it would be helpful later on.

Check ODM Entries

The two main databases to check are CuAt and CuDvDr. CuAt will handle the names, lvids, pvids, and vgids of the volume groups and logical volumes to be debugged. The main interest of CuDvDr is this is the database which controls the major/ minor number generation in LVM. The CuAt entries are common sense:

odmget -q "name=lv00" CuAt

or

odmget -q “name=workvg” CuAt

Which will give you the entries for the logical volumes or volume groups. Check to see if the disks associated with these volume groups are valid or if there are duplicate entries. The entries for CuDvDr are a bit trickier. The entries come in the form:

value1 - major number

value2 - minor number

value3 - the object associated with the major/minor number

So to look at all the entries used by rootvg, one could type:

odmget -q “value1=10” CuDvDr (due to the order that rootvg is created, it will almost always consist of major number 10)

Sometimes, the ODM gets “corrupted” where maybe two items have the same major/minor number, then you would check to see if there are duplicate entries. If the user gets confused as to what is what, the best thing is to always go to a system or a volume group that is considered reliable and run the same commands onto these systems. The output will show you what you might expect from typical system.

Is what I’m looking at really there?

Sometimes when one looks at a disk, it difficult to say which disks they are viewing. Typical confusion occurs when one is dealing with an n-tailed configuration where each rootvg has a different hdisk name for a set of shared drives. The important thing to remember here is that although the hdisk names may be different, the pvid’s on those shared drives have to be the same, regardless of the hdisk names. Now, the next thing to check must be to verify that the disk’s pvid matches that in ODM. Always pull the pvid of the disk from ODM via:

odmget -q “name=hdiskX” CuAt

Then, run either command which gives you results:

hdf /dev/hdiskX

or

lquerypv -h /dev/hdiskX

This will print out in hex format the first 128 bytes from very beginning of the disk platter. This information will be laid out in hexadecimal rows, with four columns. The pvid is held in the first two columns of hex row 0x80. Verify that this matches the pvid found in the ODM database. What people don't realize is that if some direct action is performed to the disk platter, such as dd'ing zeroes onto the beginning of the platter (and thus erasing the physical pvid), then ODM has no way of knowing this has happened. Additionally, the third and forth column of row 0x80 MUST be all zeroes.

When is Available really Available?

Disks do die, that's a fact of life. The tricky thing is that sometimes they have "strokes" so that they're partially dead and partially alive. An occasional occurrence is to have a drive where enough of it's hardware will allow the configuration of the disk into the "Available" state, but nothing else. Users are under the mistaken impression that there is something wrong with LVM when commands fail on these drives. The easiest way to test this is to run the "hdf" or "lquerypv" commands on these disks. If no data is displayed from these commands, then it's obvious that the disk is the root point of failure and must be replaced. If LVM can't even read data from the disk, then it won't be able to configure and write new parameters to the disk.

Official mirroring of rootvg statement

The support of rootvg mirroring has confused many people because of the hardcopy documentation and the answers provided at various times via phonecalls to IBM. The following explanation covers rootvg mirroring for 3.2.5 and 4.1 BEFORE the official 4.1 rootvg support was finally announced. It is included to explain the confusion prior to official 4.1 support announce (May, 1996).

Is the mirroring of rootvg officially supported? The information to mirror rootvg started out as a tip FAX. The tip FAXes typically have warnings by IBM detailing how the set of instructions are provided as a courtesy or suggestion, but not guaranteed to be officially supported as a product. These tips are usually the result of some innovative support personnel or customer who found a way to utilize the AIX system in which it hadn't originally designed to be used.

Along the way, it seems the FAX to mirror rootvg was inadvertently inserted into the official IBM publications:

AIX Version 4.1 System Management Guide - SC23-2525

AIX Storage Management Guide - GG24-4484-00

The Storage Management Guide is a "redbook" with the appropriate warnings about users being responsible for using the information in the book. The article has been removed from future editions of the System Management Guide.

However, inclusion in the System Management Guide implied that this mirroring of rootvg is a totally supported command. What is meant by totally supported? Well, it means that IBM will fix their system if it stops working or it behaves strangely and the root cause of the problem is determined to be IBM code (in this case, the operating system). In the case of mirroring rootvg, this will not be true for AIX 3.2. For AIX 4.1, support will be given if the user follows the officially supported steps for mirroring rootvg listed later in this section.

Customers may continue using mirrored rootvg on 3.2, but a their own risk. In the FAX sent to customers, the following warning was prominent in the document:

SPECIAL NOTICES

Please use this information with care. IBM will not be responsible for damages of any kind resulting from its use. The use of this information is the sole responsibility of the customer and depends on the customer's ability to evaluate and integrate this information into the customer's operational environment.

This statement is what makes the information on rootvg mirroring invalid for official documentation.

The official documentation for rootvg:

MIRRORING OF THE ROOTVG VOLUME GROUP:

1/15/97

This document is to specify the supported method for mirroring of the rootvg volume group for high availability access of the AIX operating system. Because the rootvg is special in that it contains the AIX operating system, several restrictions exist for this configuration that do not exist for the non-rootvg volume group mirroring.

This document supersedes previous IBM documentation/instructions on mirroring the rootvg volume group. Updates to this document may exist, refer to the document date on the second line to determine which is the latest set of instructions for this subject.

FUNCTION:

1) Provides the continuous operation of the AIX operating system in the event that a disk that is part of the operating system experiences failure and there exists an active mirrored copy of the operating system on some other disk.

2) Provide the ability to boot more than one disk of the rootvg in the event that another boot disk has failed. In some cases, the ability to boot from an alternate disk may require some user intervention.

RESTRICTIONS:

1) This is only valid for AIX 4.1 and later releases.

2) The user should contact their local IBM Branch office to check on the supportability of this functionality on versions of AIX older than AIX 4.1.

3) This function is not supported on /usr client, diskless client, or dataless client systems.

4) APAR ix56564 must be applied on the system for mksysb to correctly restore a mirrored volume group back into a system in mirrored format. If this apar fix is not applied, then the mksysb will still restore the rootvg onto a system, however the base mirrors of the logical volumes will not be automatically created and must be recreated by the user. To determine if this fix is installed on a machine, type:

instfix -i -k IX56564

5) In some instances, the user will not be able to capture the dump image from a crash or the dump image may be corrupted. The design of LVM prevents mirrored writes of the dump device. Depending on the boot sequence and disk availability after a crash, the dump may be in three states:

a) not available

b) available and not corrupted

c) available and corrupted

State (a) will always be a possibility. If the user prefers to prevent the risk of encountering State (c), then they can create a non-mirrored logical volume (that is not hd6) and set the dump device to that non-mirrored logical volume.

6) Parallel mirroring policy (the default) must be the policy used on the base logical volumes that constitute AIX (hd1, hd2, ...hd9var). This restriction is for performance reasons only.

7) Mirror strictness (the default) must be maintained on all the base logical volumes that constitute AIX (hd1, hd2, ...hd9var). If two copies of a logical volume reside on the same disk, this would defeat and invalidate the original goal of mirroring a rootvg volume group against possible disk failure.

8) The base logical volumes that constitute AIX:

hd1, hd2, hd3, hd4, hd5, hd6, hd8, and hd9var

must all exist on the same disk. Their equivalent mirrors must also reside on a common disk (another disk). In other words, these logical volumes and their mirrors cannot span multiple disks.

9) hd6 must exist in rootvg and must be an active paging device. Other paging devices on other volume groups are permitted, but hd6 (and it's mirrors) must exist on the disks that make up rootvg.

10) The disks that constitute the boot drives for AIX, must be disks that are supported to boot on RS/6000 platforms. A machine and disk configuration may be queried to determine if that combination of machine and disk supports booting from disk:

bootinfo -B hdiskX

If the command returns a "1", then it will be bootable by AIX. Any other value indicates that this disk is not a candidate for rootvg mirroring.

11) In the case of disk failure and then replacement, the user should consult the AIX System Management Guide appropriate to their level of AIX.

In terms of replacement of a mirrored volume group, the steps are identical (with the exception of Restriction [13]) for rootvg and non-rootvg volume groups:

a) mirror copies referenced on the bad disk are removed

b) the bad disk reference is removed (via reducevg) from the volume group.

c) follow the procedures to remirror a volume group.

12) Physical and design limitations (see Procedure 1) may exist that prevent the mirroring of rootvg.

13) When the user runs the commands:

migratepv, reorgvg, or reducevg

And the logical volume that constitutes the dump device (typically hd6) OR its mirror resides on the disk, the user must first set the dump device to /dev/sysdumpnull before executing the command. After the command is complete, the user may reset the dump device to the original dump logical volume.

14) Some models of RS/6000 do not support the bootlist command (see Procedure 6). The rootvg can still be mirrored on these systems but the possible reboot following a failure of the original boot disk will require user intervention to reselect alternate boot disks for subsequent boots.

15) The syncvg command is not available to rootvg at the time of the varyon of the rootvg volume group. Thus if after a mirrored rootvg has booted and there exists stale logical volumes in rootvg, then the user must run syncvg on rootvg to synchronize any stale logical volumes.

16) Apar ix58121 is required to be applied on UP and SMP systems before mirroring of rootvg can be initiated. Apar ix58267 is only required on SMP systems, for performance reasons. To determine if either fix is installed on a machine, type:

instfix -i -k <apar number>

Apar ix61186 is required for bosboot to work correctly in some cases.

17) In AIX 4.1, the release notes document that the simultaneous creation of mirrored logical volumes greater than 2 GB might fail and should not be attempted. Instead, the user should create the large logical volume with the mklv command. Then, the user should create a mirror for the logical volume with the mklvcopy command. This restriction also applies to the restoration of a mirrored rootvg from a mksysb image, if the mirrored rootvg contains a logical volume greater than 2 GB. To work around this, the user should set the

Create Map Files

option in mksysb to YES. This restriction does not apply to AIX 4.2 systems.

PROCEDURE:

The following steps assume the user has rootvg contained on hdisk0 and is attempting to mirror the rootvg to a new hdisk1.

1) extend rootvg to hdisk1:

extendvg rootvg hdisk1

If the user encounters the error message:

0516-050 Not enough descriptor space left in this volume group, Either try adding a smaller PV or use another volume group.

Then the user may not add hdisk1 to rootvg for mirroring. The user may attempt to mirror rootvg's logical volumes to another disk that already exists in rootvg and meets the criteria and restrictions listed in the RESTRICTIONS section. Or, the user may attempt to add a smaller disk to the rootvg. If neither option is possible, then mirroring rootvg cannot be performed on this system.

2. Disable QUORUM, by running the following:

chvg -Q rootvg

4) Mirror the logical volumes that make up the AIX operating system:

mklvcopy hd1 2 hdisk1

mklvcopy hd2 2 hdisk1

mklvcopy hd3 2 hdisk1

mklvcopy hd4 2 hdisk1

mklvcopy hd5 2 hdisk1

mklvcopy hd6 2 hdisk1 (if the user has other paging devices, rootvg and non-rootvg, they must also mirror those logical volumes in addition to hd6)

mklvcopy hd8 2 hdisk1

mklvcopy hd9var 2 hdisk1

If hd5 consists of more than one logical partition, then after mirroring hd5, the user must verify that the mirrored copy of hd5 resides on contiguous physical partitions. This can be verified with the command:

lslv -m hd5

If the mirrored hd5 partitions are not contiguous, then the user must delete the mirror copy of hd5 (on hdisk1) and rerun the mklvcopy for hd5 using the “-m” option. The user should consult documentation on the usage of the “-m” option for mklvcopy.

4. Synchronize the newly created mirrors:

syncvg -v rootvg

5. Bosboot to initialize all boot records and devices:

bosboot -a

6. Initialize the boot list:

bootlist -m normal hdisk0 hdisk1

WARNING: Even though this command identifies the list of possible boot disks it does not guarantee that the system will boot from the alternate disk in all cases involving failures of the first disk. In such situations, it may be necessary for the user to boot from the installation/maintenance media, select maintenance, reissue the bootlist command leaving out the failing disk, and then reboot. On some models, firmware provides a utility for selecting the boot device at boot time. This may also be used to force the system to boot from the alternate disk.

7. Shutdown and reboot the system:

Shutdown -Fr

This is so that the “Quorum OFF” functionality takes effect.

Why mirroring of rootvg is supported on 4.1 and beyond

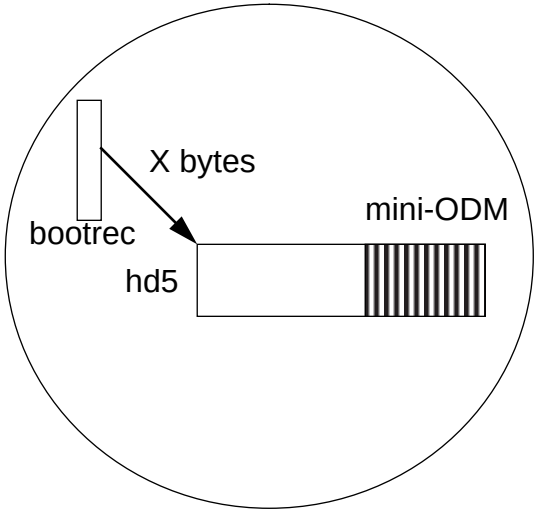
This section will attempt to show graphically why the code in 3.2.5 precludes reliable use of a mirrored rootvg. Several key phrases or definitions are explained here:

1) boot logical volume - a.k.a. blv - a.k.a. hd5 - This contains the minimal filesystem needed to begin the boot of a system. An important portion of this minimal filesystem is the “mini-ODM” held in the mini-filesystem.

2) bootrec - this is the small amount of data sitting at the beginning of the disk that the IPL ROS reads to find the reference to the blv.

3) savebase - the command that will update the mini-ODM that resides on the same disk as /dev/ipldevice.

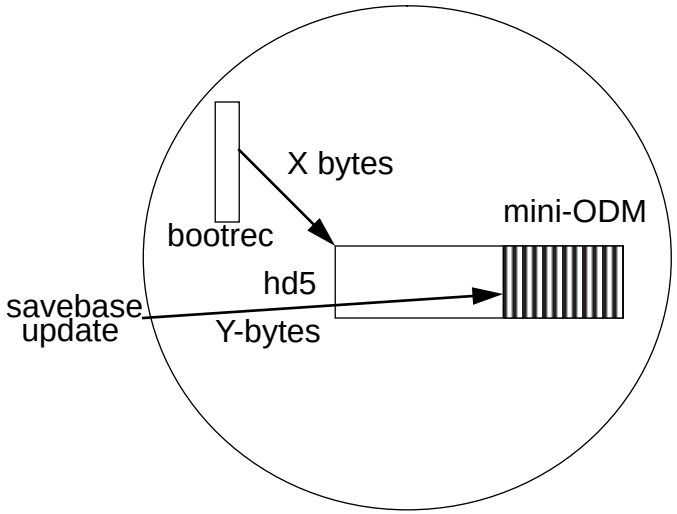
The relationship between the bootrec, the blv, and the mini-ODM is illustrated:



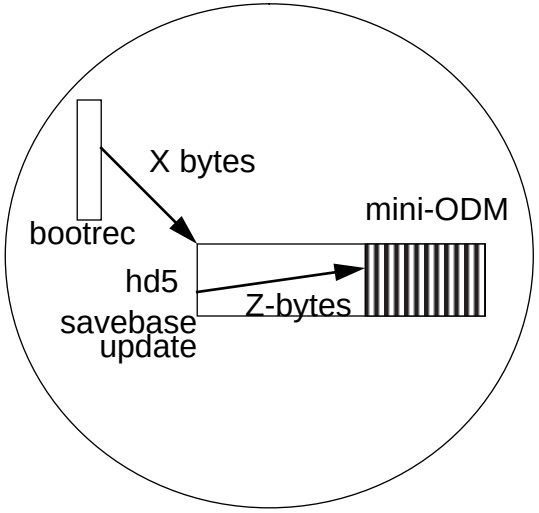
The bootrec is read by the IPL ROS, it tells the ROS that it needs to jump X bytes into the disk platter to read the hd5, the boot logical volume. The the processor starts reading in the mini-filesystem to start the AIX system on its way to boot (note that this is a VERY simplified view of the boot process, enough for basic understanding of upcoming conflicts). As part of processing the mini-AIX filesystem, there is a mini-ODM read into RAM. When the true rootvg filesystem comes online, AIX will merge the data held in the mini-ODM with the true ODM held in /etc/objrepos on the true rootvg filesystem.

Whenever an LVM command is executed that will change the “main” ODM, there is a command called “savebase” which is run at the successful completion of the LVM command. Savebase takes a snapshot of the main ODM and compresses it and picks out only what is needed for boot, such as LVM information concerning logical volumes in rootvg, etc. It takes this compressed boot image and looks at /dev/ipldevice and finds out what disk this represents. Typically on most systems, /dev/ipldevice is a link to /dev/hdisk0 (the install rootvg). /dev/ipldevice is really needed to tell us which disk holds hd5. Here the first big difference between 3.2 and 4.1 code becomes apparent.

3.2 savebase - consults the bootrec and finds out how many Y bytes the mini-ODM exists from the beginning of the disk and goes to that Y-byte location and updates the mini-ODM. It skips past the entire concept of hd5 and updates a physical disk location from the beginning of /dev/ipldevice. Shown here:



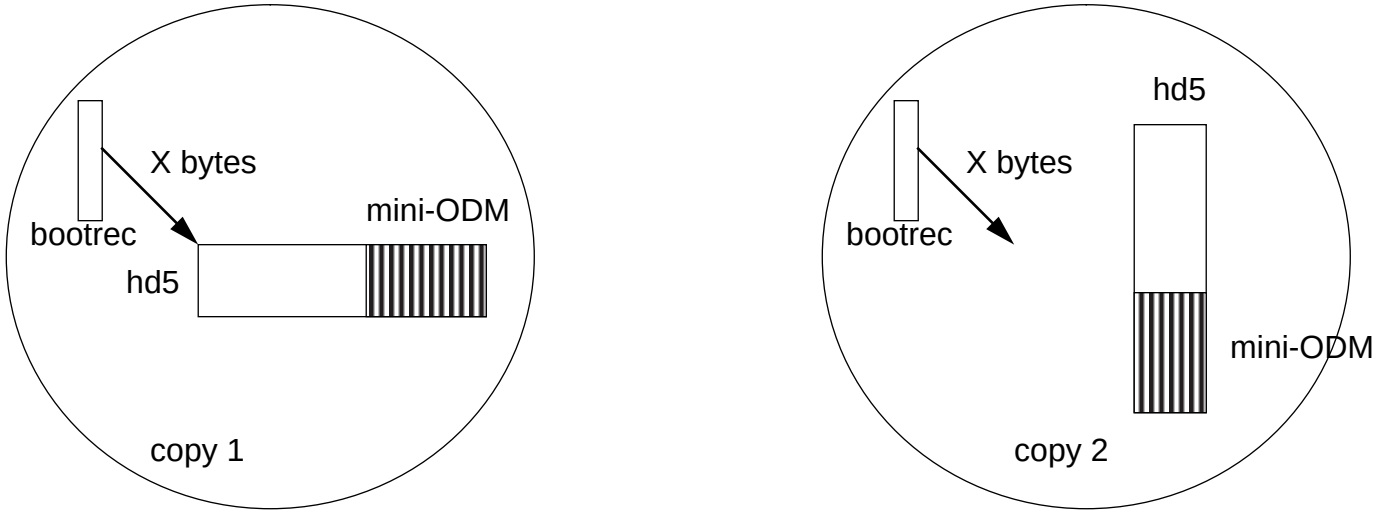
4.1 savebase - consults the bootrec and finds out how many Z bytes the mini-ODM exists from the beginning of hd5 and goes to that Z-byte location (from the beginning of hd5) and updates the mini-ODM. In this case, the data is written in reference to the logical volume hd5, not a raw physical disk location.



This is one of the first reasons why rootvg mirroring on 3.2 has problems. Mirroring is not done on the disk-to-disk level, but on the logical volume level. When the hd5 is updated in 3.2, the device to update the mini-ODM is /dev/hdisk0. Thus, a mirrored hd5 on the rootvg disk, /dev/hdisk0 doesn't know it must update it's copy of hd5 on /dev/hdisk1. However in 4.1, the mini-ODM is updated via the logical volume hd5. Thus, any mirrors of hd5 sitting on another disk automatically gets updated.

Another major difference between the 3.2 and 4.1 versions of the boot information process is the creation of bootrec and its relationship with hd5. As mentioned earlier, the bootrec has a reference point that tells the IPL ROS how many X bytes into the disk platter hd5 begins. First, one cannot mirror the bootrec because it is literally a set amount of bytes that are laid down onto the beginning of the boot disk by bosboot. Since the bootrec doesn't exist in logical volume format, it can't be mirrored for automatic updates. However, one wouldn't want an automatic update anyway. The user must consider that a mirrored set of hd5s will most likely NOT exist on the same physical partition number on both disk drives. If the user

were to “dd” the bootrec from the original rootvg drive to the mirrored disk (and had a mirrored hd5), one would get something like:



Although the bootrec is now identical in copy 2 as it is in copy 1, it is also pointing to the identical location of hd5 that copy 1 is pointing to. However, as the example shows, the mirrored hd5 in copy 2 is in a different physical location. The instructions (that are not supported) for 3.2 rootvg mirroring works around this by creating a second boot logical volume, hd5x and causes a second bootrec to be created that points directly to hd5x via a second bosboot done to the system. This will allow the system to boot from the second copy.

AIX 4.1 solves this problem by having a smart bosboot. When bosboot is run, the logic checks to see if hd5 is mirrored. If it is mirrored it does a query to find out what disk the copy of hd5 is mirrored onto. Then, bosboot writes a customized bootrec onto that mirroring disk. This customized bootrec will correctly point to the physical location of the mirrored hd5!

The main concern of users still using rootvg mirroring on AIX 3.2 should be the ODM out-of-sync issue with savebase. Here is typically what will happen: the user will mirror their rootvg, create the hd5x and perform the second bosboot. Then, during the next few months they will create, delete, modify volume groups and logical volumes. During this time, the true mini-ODM is being updated on the original rootvg disk. The mirrored copy of hd5 will have an old stale image of the mini-

Hello outside world,

This is an extra notice being sent to those asking for the LVM ODDS AND ENDS notes. This is essentially the rootvg mirroring whitepaper, but with the bonus of the mandatory apar fixes required listed at the end of the document. Please note the notice about prpq.clvm.

Many people don't read the document close enough and they order fixes for the prpq.clvm fileset even though they don't have it installed. Three key things we would like you to be aware of:

1) If you have an MP machine and you do mirroring (rootvg or not), we would strongly suggest you pick up the mirroring fixes listed below.

2) These are NOT the latest filesets. Fileset numbers slowly churn upward with accumulated fixes, thus there are fixes even later than this. Please consult your service provider for the latest lvm commands/libs/driver fixes. Note that in AIX 4.1, the commands and libraries are in the fileset bos.rte.lvm, but the lvm device driver is in the fileset bos.rte.devices. However, in AIX 4.2, all LVM components are in bos.rte.lvm.

3) This whitepaper was written with a "conservative" point of view. Thus some restrictions are there because there was not enough test time given to testing all possible combinations of RS/6000 hardware and logical volume placements. Deviations from these notes are at your own risk.

But while I have your attention, let me add a blurb about mirror reads. One of the fixes listed below fixes a problem on MP machines where the least busy mirror is not always being chosen for parallel reads. Once you put that fix on your system and reboot (an lvm device driver fix), then you should truly get the least busy drive, as known by the lvm device driver. I get notices from people in pmrs that say:

"I'm doing triple mirroring, but the reads (from my iostat program) seem to always go to the same disk. I don't think the most efficient mirror is being chosen."

If you have the fix, apar ix58267/apar ix59519, the algorithm goes like this:

number_of_outstanding_ios = i/o count of first mirror in mirror "array"

```
mirror_to_read = first copy

for (loop = second copy; loop <= third copy; loop++)
{
        if (this loop copy's outstanding i/o's < number_of_outstandin
g_ios)
        {
                number_of_outstanding_ios = this loop copy's outstand
ing i/o's
                mirror_to_read = this mirror (loop)
        }
}
```

From the algorithm above, you can see why many times the first mirror in the array will most likely be chosen. The misleading stat from io stat can result from the simple fact that when you run a testcase, by the time the next read reqest comes down, the first mirror is chosen again because it already finished its previous read. Another problem that occurs is that people try to make their disks "busy" by doing something like:

```
dd if=big_test_file of=/dev/hdiskX count=100000
```

They do something like that, yet LVM still picks hdiskX instead of hdiskY. The key point here is that by doing dd directly to /dev/hdiskX, you are bypassing the LVM device driver completely. Thus, the number of i/o counts that should increase with this dd test aren't being taken into consideration by the bypassed LVM device driver. The correct way to stress a disk if you want to test out mirror efficientcy is to dd the file directly into a logical volume that exists on a disk(s). Then, the lvm device driver becomes involved.

Hopefully, this section above will be included in the next set of LVM - SHARE notes.

MIRRORING OF THE ROOTVG VOLUME GROUP
01/15/97

This document is to specify the supported method for mirroring of the

rootvg volume group for high availability access of the AIX operating system. Because the rootvg is special in that it contains the AIX operating system, several restrictions exist for this configuration that do not exist for the non-rootvg volume group mirroring.

This documentation supersedes previous IBM documentation/instructions on mirroring the rootvg volume group. Updates to this document may exist, refer to the document date on the second line to determine which is the latest set of instructions for this subject.

FUNCTION:

1) Provides continuous operation of the AIX operating system in the event that a disk that is part of the operating system experiences failure and there exists an active mirrored copy of the operating system on some other disk.

2) Provide the ability to boot more than one disk of the rootvg in the event that another boot disk has failed. In some cases, the ability to boot from alternate disks may require some user intervention.

RESTRICTIONS:

1) This is only valid for the AIX 4.1 and AIX 4.2. In reference to a pars, two numbers will be provided - one for 4.1 fixes and one for the equivalent 4.2 fix. A table at the end of this document will summarize the fixes.

2) The user should contact their local IBM Branch office to check on the supportability of this functionality on versions of AIX older than AIX 4.1.

3) This function is not supported on /usr client, diskless client, or dataless client systems.

4) APAR ix56564 (AIX 4.1 only)must be applied on the system for mksysb to correctly restore a mirrored volume group back into a system in mirrored format. If this apar fix is not applied, then the mksysb will still restore the rootvg onto a system, however the base mirrors of the logical volumes will not be automatically created and must

be recreated by the user. To determine if this fix is installed on a machine, type:

```
instfix -i -k IX56564
```

This fix is default in the AIX 4.2.0 code.

5) In some instances, the user will not be able to capture the dump image from a crash or the dump image may be corrupted. The design of LVM prevents mirrored writes of the dump device. Depending on the boot sequence and disk availability after a crash, the dump may be in three states:

a) not available
b) available and not corrupted
c) available and corrupted

State (a) will always be a possibility. If the user prefers to prevent the risk of encountering State (c), then they can create a non-mirrored logical volume (that is not hd6) and set the dump device to that non-mirrored logical volume.

6) Parallel mirroring policy (the default) must be the policy used on the base logical volumes that constitute AIX (hd1, hd2, ...hd9var). This restriction is for performance reasons only.

7) Mirror strictness (the default) must be maintained on all the base logical volumes that constitute AIX (hd1, hd2, ...hd9var). If two copies of a logical volume reside on the same disk, this would defeat and invalidate the original goal of mirroring a rootvg volume group against possible disk failure.

8) The base logical volumes that constitute AIX:

hd1, hd2, hd3, hd4, hd5, hd6, hd8, and hd9var

must all exist on the same disk. Their equivalent mirrors must also reside on a common disk (another disk). In other words, thes

e logical
volumes and their mirrors cannot span multiple disks.

9) hd6 must exist in rootvg and must be an active paging device. Oth
er paging
devices on other volume groups are permitted, but hd6 (and it
’s mirror)
must exist on the disks that make up rootvg.

10) The disks that constitute the boot drives for AIX, must be disks
that are
supported to boot on RS/6000 platforms. A machine and disk
configuration may be queried to determine if that combination
of
machine and disk supports booting from disk:

bootinfo -B hdiskX

If the command returns a "1" then it will be bootable by AIX.
Any other value indicates that this disk is not a candidate f
or
rootvg mirroring.

11) In case of disk failure and then replacement, the user should con
sult the
AIX System Management Guide appropriate to their level of AIX
.

In terms of replacement of a mirrored volume group, the steps
are identical (with the exception of Restriction [13]) for
rootvg and non-rootvg volume groups:

a) mirror copies referenced on the bad disk are removed.
b) the bad disk reference is removed (via reducevg) from the
volume
group.
c) follow the procedures to remirror a volume group.

12) Physical and design limitations (see Procedure 1) may exist that
prevent
the mirroring of the rootvg.

13) When the user runs the commands:

migratepv, reorgvg, or reducevg

And the logical volume that constitutes the dump device (typi
cally
hd6) OR its mirror resides on the disk, the user must first s

et
the dump device to /dev/sysdumpnull before executing the comm
and.

After the command is complete, the user may reset the dump de
vice
to the original dump logical volume.

14) Some models of RS/6000 do not support the bootlist command (see
Procedure 6). The rootvg can still be mirrored on these syst
ems
but the possible reboot following a failure of the original b
oot disk
will require user intervention to reslect alternate boot disk
s for
subsequent boots.

15) The syncvg command is not available to rootvg at the time of the
varyon of the rootvg volume group. Thus if after a mirrored
rootvg has booted and there exists stale logical volumes in
rootvg, then the user must run syncvg on rootvg to synchroniz
e
any stale logical volumes.

16) Apar IX58121 (AIX 4.1) is required to be applied on UP and SMP sy
stems
before mirroring of rootvg can be initiated. Apar IX58267 (A
IX 4.1)
is only required on SMP systems, for performance reasons. To
determine if either fix is installed on a machine, type:

instfix -i -k <apar number>

Apar IX61186 is required for bosboot to work correctly in som
e
cases.

The equivalent AIX 4.2 apar fixes are listed in table at the
end
of this document

17) At the end of this document is a table of apar fixes for AIX 4.1
and AIX 4.2. In addition to the madatory fixes listed in RES
TRICTIONS
(4) and (16), other apars relevant to LVM mirroring are liste
d
as highly recommended fixes a user should place on their syst
em.
These fixes are for those users who are running their system
in
a non-rootvg configuration as well as those who are.

18) In AIX 4.1, the release notes document that the simultaneous creation
of mirrored logical volumes greater than 2 GB might fail and should not be attempted. Instead, the user should create a logical volume with the mklv command. Then, the user should create a mirror for the logical volume with the mklvcopy command. This restriction also applies to the restoration of a mirrored rootvg from a mksysb image, if the mirrored rootvg contains a logical volume greater than 2 GB. To work around this, the user should set the

Create Map Files

option in mksysb to YES. This restriction does not apply to AIX 4.2 systems.

PROCEDURE:

The following steps assume the user has rootvg contained on hdisk0 and is attempting to mirror the rootvg to a new disk hdisk1.

1. Extend rootvg to hdisk1:

```
extendvg rootvg hdisk1
```

If the user encounters the error message:

```
0516-050 Not enough descriptor space left in this volume group.
Either try adding a smaller PV or use another volume group.
```

Then the user may not add hdisk1 to rootvg for mirroring. The user may attempt to mirror rootvg’s logical volumes to another disk that already exists in rootvg and meets the criteria and restrictions listed in the RESTRICTIONS section above. Or, the user may attempt to add a smaller disk to the rootvg. If neither option is possible, then mirroring rootvg cannot be performed on this system.

2. Disable QUORUM, by running the following:

```
chvg -Qn rootvg
```

3. Mirror the logical volumes that make up the AIX operating system:

```
        mklvcopy hd1 2 hdisk1            #  /home file system
        mklvcopy hd2 2 hdisk1            #  /usr file system
        mklvcopy hd3 2 hdisk1            #  /tmp file system
        mklvcopy hd4 2 hdisk1            #  / (root) file system
        mklvcopy hd5 2 hdisk1            #  blv, boot logical volume

        mklvcopy hd6 2 hdisk1            #  paging space
        (if the user has other paging devices, rootvg and non-rootvg,
 they

        must also mirror those logical volumes in addition to hd6)
        mklvcopy hd8 2 hdisk1            #  file system log
        mklvcopy hd9var 2 hdisk1         #  /var file system


        If hd5 consists of more than one logical partition, then afte
r
        mirroring hd5 the user must verify that the mirrored copy of
        hd5 resides on contiguous physical partitions.  This can be v
erified
        with the command:

                lslv -m hd5

        If the mirrored hd5 partitions are not contiguous, then the
        user must delete the mirror copy of hd5 (on hdisk1) and
        rerun the mklvcopy for hd5 using the "-m" option.  The user
        should consult documentation on the usage of the "-m" option
        for mklvcopy.
```

4. Synchronize the newly created mirrors:

```
        syncvg -v rootvg
```

5. Bosboot to initialize all boot records and devices:

```
        bosboot -a -d /dev/hdisk?

        Where hdisk? is the first hdisk listed under the "PV" heading

        after the command:

        lslv -l hd5

        is executed.
```

6. Initialize the boot list:

bootlist -m normal hdisk0 hdisk1

WARNING: Even though this command identifies the list of possible boot disks it does not guarantee that the system will boot from the alternate disk in all cases involving failures of the first disk. In such situations, it may be necessary for the user to boot from the installation/maintenance media, select maintenance, reissue the bootlist command leaving out the failing disk, and then reboot. On some models, firmware provides a utility for selecting the boot device at boot time. This may also be used to force the system to boot from the alternate disk.

7. Shutdown and reboot the system:

shutdown -Fr

This is so that the "Quorum OFF" functionality takes effect.

Description	UP & MP	MP only	Apar Level
mksysb fix	ix56564		4.1
	------	-------	
	In 4.2.0 code		4.2
mirror poweroff crash	ix58121		4.1
	------	-------	
	ix59517		4.2
mirror read algorithm		ix58267	4.1
	------	-------	
		ix59519	4.2
mirror sync hang		ix58889	4.1
	------	-------	
		In 4.2.0 code	4.2
mirror sync crash		ix58919	4.1

```
                          extra.txt
                        ------          -------
                                        ix59520         4.2
-------------------------------------------------------------------------
-----
mklvcopy failure        not needed                      4.1
                        ------          -------
                        ix59414                         4.2
-------------------------------------------------------------------------
-----
bosboot failure
                        ix61186                         4.1
                        ------          -------
                        ix62417                         4.2
-------------------------------------------------------------------------
-----
```

The fixes listed above are packaged for the clvm prpq under the cumulative apar IX58707, except the bosboot apars. To determine if you have the clvm prpq installed, type:

```
lslpp -h prpq.clvm
```

If it is not installed onto your system, you do not need to order IX58707.