75

AIX

January 2002

In this issue

update

AIX Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: trevore@xephon.com

North American office

Xephon
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

Subscriptions and back-issues

A year's subscription to *AIX Update*, comprising twelve monthly issues, costs £180.00 in the UK; $275.00 in the USA and Canada; £186.00 in Europe; £192.00 in Australasia and Japan; and £190.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the November 1998 issue, are available separately to subscribers for £16.00 ($24.00) each including postage.

AIX Update on-line

Code from *AIX Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/aix; you will need to supply a word from the printed issue.

Editors

Trevor Eddolls and Richard Watson

Disclaimer

Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, scripts, and other contents of this journal before making any use of it.

Contributions

When Xephon is given copyright, articles published in *AIX Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

Two savers – space and time

The following are a couple of quick ‘saver’ tips. One saves (typing) time and one saves space and maybe even a little back-up time.

SAVING SPACE WITH PIPES

Named pipes is a very powerful Unix mechanism to re-route command output and input. One simple example is that of creating a compressed file directly rather than having to first create the file and then compress it. A practical example would be the need to create a large *tar* file, for back-up or other purpose, that exceeds either the 2GB file limit, if you are faced with it, or any available space you have.

Here are the steps illustrated as an actual example:

```
- mknod /tmp/thepipe p                  < make the pipe >
- compress < /tmp/thepipe > t05.tar.Z & < start the compress "sucker">
- tar -cf /tmp/thepipe /t05             < issue the tar command >
```

This results in the file: 50918897 Nov 09 08:57 t05.tar.Z; while a ‘normal’ tar results in: 141680640 Nov 09 08:53 t05.tar

This is a relatively small example, size wise; however, as you can see, the savings are significant.

To continue the example, the steps to read or use the compressed *tar* file, without actually uncompressing it, are:

```
- mknod /tmp/thepipe p                     < make the pipe >
- uncompress < t05.tar.Z > /tmp/thepipe &  < start the "sucker" >
- tar -tvf /tmp/thepipe | pg               < issue the tar >
```

This results in:

```
tar: The block size is 8 bytes.
drwxr-sr-x 203 203       0 Oct 23 06:58:53 2001 /t05/
drwxrwx--- 203 203       0 Oct 23 06:25:14 2001 /t05/lost+found/
drwxr-sr-x 303 303       0 Oct 23 06:58:53 2001 /t05/archlogs/
drwxr-sr-x 303 303       0 Nov 09 05:25:04 2001 /t05/archlogs/ohub/
```

and so forth, without ever having to actually uncompress the *tar* file.

While *tar* was the choice in these examples, the pipe methodology used here has a myriad other uses and can be integrated with most Unix commands, and many other utilities (Oracle Export for example).

SAVING SOME TYPING TIME WITH AWK

Often I've found myself typing up scripts, with many repetitive lines of code, in order to do quick changes or fixes. One recent example is a server that had a large number of empty hdisks on it, which needed to be removed from the ODM to have some reconfiguration work on them, then to allow **cfgmgr** to do its thing. Rather than type **rmdev –dl hdisk*x*** a couple of hundred times, the following is a simple, practical example of using **awk** to save some typing:

```
-  lspv | awk {'print "rmdev -dl " $1'} > /tmp/becareful_running_this.sh
```

Then, after careful reviewing and editing, the /tmp/becareful_running_this.sh script will looks like this:

```
rmdev -dl hdisk0
rmdev -dl hdisk1
rmdev -dl hdisk2
rmdev -dl hdisk3
rmdev -dl hdisk4
rmdev -dl hdisk5
rmdev -dl hdisk6
…….
rmdev -dl hdisk177
```

Simply run:

```
sh /tmp/becareful_running_this.sh
```

While this is a very simple example, and admittedly there are plenty of other approaches to accomplish the same thing, I like this example because it illustrates the simplicity of using **awk** (which many shy away from), and it also allows you to document and review the potentially hazardous commands you are about to run.

David Miller
Database Architect
Baystate Health Systems (USA)

Tivoli Storage Manager

TSM or Tivoli Storage Manager (ADSM as was) is, on the face of it, a simple application. What could be more simple than backing up one computer to another? But to run a competent TSM installation takes much time and planning. I hope the following notes will be of use to anyone administering a TSM server (or clients), both experienced and novice.

IT'S A FULL-TIME JOB

What may start out life as a back-up solution for three or four AIX servers can easily become a corporate solution for hundreds of heterogeneous servers and workstations. Either way, just allocating an hour or so a day to the administering of the TSM server will almost never suffice. For 'small' installations (by which I mean an installation with only one TSM server and fewer than 20 servers to back-up) five half-days may suffice, but a larger set-up will require more time devoted to it.

There is one thing that can save you much time and heartache – getting the planning and set-up correct from the start.

PLANNING

Planning is always a good idea! However, what is the difference between a good plan and a bad one? When will you discover which plan yours is? Can you change the set-up after a month or year?

Not surprisingly, there are no easy answers to the above questions. Can you predict where your company will be in five years? Ah, if only we had a crystal ball. So, what are the important things to incorporate into your plan? Below are three headings to work with:

- Budget
- Growth
- Monitoring.

Budget

As with all things, money plays an overriding part. Whether you are simply trying to fund an administrator or to purchase a replacement TSM solution, it is the (someone's) budget that will dictate what can be afforded.

With respect to the hardware and software, it is advantageous to get the best solution you can afford. My experience has always been that a company will try to get as much (and more) from any one purchase. TSM is no exception, because all those important managers will suddenly want their PCs backed up over the network and the data kept for several years. The larger the tape silo, the more capacity of storage pools, the more network adapters in the server etc, will all help days/ months/years down the line.

Growth

Growth goes with budget really, but can be looked at in terms of future planning. Can your current solution handle all the servers and workstations in your company (do you have that many client licences?). What would happen if the requirement for keeping data went from, say, a month to one to five years? Can your current solution handle that amount of data? Some companies have a legal requirement to keep data for 25+ years (for things like copyright). If money is available in your budget for TSM and you are not expanding to cope with new traffic, look at expanding the capacity of the service or possibly the network throughput.

Monitoring

So to the day-to-day stuff. What follows can be used for capacity planning (growth) and trend analysis. I will concentrate on the monitoring aspect, but I have included some set-up information. The list that follows is not exhaustive. It is, however, in some kind of order of importance, the top being most important.

- TSM database size – probably the most important aspect to monitor. If the database fills up, TSM will stop! But what is a safe operating level and how much space do you need to allocate to it?

What recovery features are best employed?

The overall size of the database will depend on: the total number of files backed up; the number of versions of each file; the amount of activity log you keep online; the size of the database queries you do; and a few other things. You cannot guess the size of database queries, and making a good guess at the number of files in a large organization is unrealistic. So, when starting from scratch, allocate about 1GB and add one client, then grow the database as required. There is no maximum size so think carefully about where the storage space for the database is. Do you have enough space in rootvg for one copy of the database? See the *TSM Administrators Guide* for a better explanation of estimating the initial size of the database.

I would advise mirroring each dbvol (and logvol) at the TSM level rather than at the AIX level. This allows each mirror to be on separate volume groups and disk packs, ie internal SCSI and external SSA.

For a safe operating level I would suggest no greater than 70%, no matter how big the database. This margin should allow you to cope with the data most (normal) clients throw at you. Motto: always allow plenty of room, the unexpected does and will happen. Use the command **query db f=d** to check the size and utilization. Keep an eye on the 'max. pct util' statistic. This tells you the maximum usage of database space at any one time. It can be zeroed using the command **reset dbmaxutilization**. This is the main size statistic to trend.

Over time, the database will become fragmented, causing some space to become 'dead'. To reclaim this space, a **dsmserv unload** and **dsmserv load** must be performed. Once every six months is a reasonable interval for this operation. Note: the unload and load require that the server is down and the service is unavailable. Remember the operation may take some time to complete.

It is a matter of preference what form of database back-ups are to be performed. If you have fast tape devices, then a daily back-up is worth doing (as well as sending one offsite!). Back-ups to disk

are fine (devtype=FILE) but you must have the spare disk capacity. This back-up may well be a db_snap rather than a full back-up. You should try to back up the database once a day for disaster recovery reasons, if only by doing an incremental back-up.

- TSM log – not quite as important as the database, but can cause problems if not monitored. Checking is easy. Use the **query log f=d** command to monitor the usage. And on a daily basis, just after recording the output from the previous command, reset all the statistics using **reset logconsumption** and **reset logmaxutilization**. Over a period of time, this should give you some idea of how good your estimates of space are for the logs and whether you need to allocate more log volumes.

- TSM database back-ups (recovery) – TSM will work without taking a database back-up but you will get plenty of warning messages about it. This topic really comes under the DR planning. However, one fact to be aware of is that, if you dump database back-ups to file (online disk), you must watch that there is enough space in the filesystem, otherwise TSM will crash.

- Error message checking – TSM logs many messages. It is all too easy to miss something. A simple AIX command will enable you to get all the warning/error messages TSM throws out (for the dsmcmd script, see later):

```
grep AN[SR][0-9]*[WED] `/usr/local/bin/dsmcmd 'q actlog'
```

  I suggest putting this into a script that will mail you the output and put the script into a crontab entry. You might also want to store the last few days/weeks/months of activity logs offline. Simply capture the output for the **query actlog begintime=now-24** and store it away.

- Schedule checking – depending on how you schedule your TSM activity, you may want to track the success or failure of the schedules (both client and administrative). Look for anything with the completion code of ‘missed’ (/usr/local/bin/dsmcmd –comma "q sched").

- Free tapes – if you are using a managed tape library (eg 358*x*,

357*x*, etc), you need to monitor the number of scratch tapes you have available. The command **select count(*) from libvolumes where status='SCRATCH'** will help. What is a safe number of scratch tapes to keep depends on the capacity of the library and tapes, along with the amount of data that is dumped to the library each day. Obviously this number should never be zero, but even the most well-managed of systems should keep at least three days' worth of scratch tapes in a library with many spare blank tapes outside the system in case the number of scratch tapes gets too low (this presumes that you do have some spare slots in the library).

- AIX disk space – this should be something to be aware of rather than monitor. You should always have space for new TSM disk pool volumes or be able to expand the TSM database/logs. Hopefully, you will be monitoring and maintaining the database, which will prevent the need to allocate new db/log volumes. It is difficult to suggest how much free AIX system disk space to maintain because it depends on your TSM set-up and strategy. However, providing you are able to expand the disk capacity of the system, for a system which is not expanding (adding more clients), you should be OK.

- Network performance – this may be out of your control, but obviously you need to monitor the throughput of all the clients. You can check the activity log for the statistics for each client session as well as turning session accounting on (**set accounting on**), remembering to set the environment variable DSM_ACCOUNTING_DIR. From the dsmaccnt.log file, you can work out the throughput for each client/server session. A good idea is to benchmark each client by running an incremental back-up when there is no other activity on the server. This will give you something to compare normal activity with. For help with tuning the TSM server network options, try looking at the IBM Redbooks and also the postings on the ADSM bulletin board (www.adsm.org). TSM TCP/IP parameters to use include: TCPWINDOWSIZE, TCPNODELAY, and USELARGEBUFFERS. AIX parameters to look at include

thewall, sb_max, tcp_sendspace, and tcp_recvspace. Also, for adapters such as ATM, check the send and receive packet sizes.

- Storage pools – (**query stgp**) it depends on your strategy but monitor such things as the amount of space you have in both tape and disk storage pools. You want to keep track of tapes available in tape storage pools so that TSM can still allocate scratch volumes to it. The calculation is:

  'Max Scratch Volumes Allowed' value from the q stgp xxxx f=d minus the output from the command select count(*) from volumes where stgpool_name='xxxx'.

  Another parameter to tune is the 'Reclaim Threshold' value in the storage pool definition. Too low and your tape drives will be very busy. Too high and, when a reclaim is performed, it will take hours.

  For disk storage pools, tracking space against the amount of data sent in one session is a good idea. One idea is to flush the disk pools to tape before the overnight back-up window starts.

- Tapes – tape errors must be watched for. TSM has a habit of putting tapes into an 'unavailable' state. Try the simple command **select volume_name from libvolumes where access='UNAVAILABLE'**.

- Amount of data per client – it is worth keeping an eye on which clients are taking up most space on your TSM server. The **query auditoccupancy** command will tell you how much, in KB, each client is taking. You must run the **audit licence** command before the statistics for the query are updated.

When writing AIX scripts to manipulate or process output from TSM commands I have found it useful to define an administrator to use specifically from the AIX command line, eg batch. Then use this administrator within a script eg:

```
if [[ $1 = "-comma" ]]
then
  /usr/bin/dsmadmc -id=batch -pass=<passwd> $1 "$2"
else
  /usr/bin/dsmadmc -id=batch -pass=<passwd> "$1"
```

```
fi
                    (dsmcmd script)
```

The ‘comma’ option gives the output of any command in comma separated form. Very handy for **awk** scripts or PERL.

By now, you should have rather a lot of things to monitor and watch for. The above list is by no means exhaustive but hopefully will give you some pointers.

There are plenty of places to get more information, help, and advice. The Tivoli Internet site has copies of all the manuals and Redbooks. The manuals are shipped with the product as an installable image, and there is a very good bulletin board site at www.adsm.org where most things to do with all flavours of TSM have been or are being discussed.

Phil Pollard
Unix and TSM Administrator (UK)

Why not share your expertise and earn money at the same time? *AIX Update* is looking for technical articles and hints and tips about AIX performance, as well as example scripts that experienced AIX users have written to make their life, or the lives of their users, easier.

Articles can be e-mailed to Trevor Eddolls at trevore@xephon.com or sent to any of the addresses shown on page 2. A copy of our *Notes for contributors* is available from www.xephon.com/nfc.

Design considerations for SSA disks and data availability in an HACMP or high availability environment

This article discusses some of the issues involved in designing and configuring SSA disk installations in an HACMP environment – HACMP is IBM's High Availability Cluster Multi-Processing software. Some of the concepts discussed are not specifically related to HACMP, nor to SSA disks alone, but are standard AIX features which can be used in any situation requiring high availability of data.

A simple HACMP cluster consists of two nodes attached to one or more external SSA disk subsystems on which application data is stored. In the event of a node failure, the cluster logical and physical design must be such that the standby node (ie the node that has not failed) must be capable of taking over critical applications running on the failed node, which means that the external application data must also be accessible on the standby node after a takeover.

The article will not discuss RAID configurations, only standard AIX mirrored configurations and quorum issues.

AVAILABILITY OF SSA DISKS

The location of mirrored copies of logical volumes on disks and whether or not the quorum should be turned on or off are important design considerations in any HACMP environment. Poor planning and bad design can mean that disk failures become single points of failure, resulting in application failures, data corruption, and in some cases system crashes. When a system crashes, normally a node takeover will occur, but, if data has already been corrupted, applications may still be unusable even after a takeover.

In the type of HACMP environment we are considering, all SSA disk configurations should have mirrored logical volume copies to ensure the highest level of availability. Although this is the most desirable configuration, it may not be possible if there are cost constraints. If mirroring is not used, then, should disks fail or disk subsystems

become inaccessible for whatever reason, it will not be possible to replace failed disks and reintegrate them into their respective volume groups without considerable downtime. This may be acceptable in some installations, but generally speaking is not desirable since it defeats the objective of trying to attain high availability for an application and its data.

The location of disks and mirrored copies in and across subsystems, and their connections within SSA loops, must be carefully considered in order to achieve the highest possible level of availability of data within a cluster, and also to reduce downtime to a minimum when the reintegration of failed components is required.

SSA disk, adapter, and cabling configurations should be such that, no matter which node has control of cluster resources, high availability of disks and data is considered to have been achieved only if one of the following criteria is valid when an SSA loop is broken:

- A cluster node is still able to access all disks in the resource group it currently controls – this occurs when all disks are still accessible by using alternative routes around the loop and in this situation no logical volume partitions become stale.
- A cluster node can access a complete set of mirrored copies of the data within the resource group, although there may inevitably be stale partitions – in this case either single disks may have failed, or multiple disks have become inaccessible because of SSA adapter or subsystem failure, or some cable break has occurred. The most ideal high-availability designs include sufficient redundancy to allow all disks to be accessible through alternative loop routes when SSA adapters fail, or when loop cabling fails, but cost restraints or bad design may prevent this ideal from being achieved.

A resource group is a logical configuration within HACMP, which can contain combinations of IP addresses, volume groups, logical volumes, filesystems, and applications, although all may not necessarily be configured as part of a single resource group. Each of these possible components may be considered to be a resource which can be acquired (taken over) by another cluster node in the event of a forced takeover or actual node failure.

SSA LOOP FAILURE

SSA loop failures will occur in clusters connected to one or more SSA subsystems if any of the following failures occur within the loop:

- A node fails.
- An SSA adapter fails.
- An SSA cable fails. The location of the cable and what it is connected to is critical. In certain circumstances the cable failure may leave the loop intact, particularly when connected to bypass cards (see below), which may be configured to close the loop in the event of the failure of a cable connected to one of the jumpers on the card.
- A disk subsystem fails.
- A subsystem disk fails.
- A signal/bypass card fails. The bypass card sits in SSA subsystems and contains either the jumpers that connect two 4-pack disks, for example the card containing jumpers 4 and 5 (which are internally connected to disk 4 at the end of the first pack and disk 5 at the start of the second pack), or jumpers used to connect to SSA adapters or other subsystems, such as the card containing jumpers 1 and 16.

The resulting disk access (or lack of it) will be determined by what has failed, and where the failure occurred; the configuration must be designed to minimize the effects of any of these failures.

In clusters where each node contains a single SSA adapter (a not very desirable configuration), a forced node takeover may be required in order to ensure that clients can continue to access their application data. An SSA adapter failure in this type of configuration will almost always be promoted to a node failure, since you can never be certain where the loop has been broken without some serious problem determination, which would inevitably result in an unacceptable period of downtime. The easiest solution is to automatically promote to a node failure rather than to try using complex logic within your customized scripts to determine whether or not this is necessary.

For those of you unfamiliar with HACMP, customized scripts can be created to perform whatever actions you desire when entries are placed in the system error log, and SSA loop breaks may produce a number of different entries, depending on where the break occurred and which failure caused the break. It is these scripts that contain the logic to define the actions to be taken.

QUORUM AND MIRRORING

When a volume group is varied on and **quorum** is turned on, for the volume group to remain accessible, AIX requires more than 50% of the Volume Group Descriptor Areas (VGDAs) contained on the disks to be available. The VGDAs contain volume group information, such as physical partition maps, which contain lists of partitions allocated to all logical volumes within the volume group, and also time stamps, which are compared between disks to identify the most up-to-date copy of the VGDA.

Allied to the VGDA is the Volume Group Status Area (VGSA), which contains additional information about the physical partitions, such as which partitions are stale, or which disks in the volume group are missing. Like the VGDA, a quorum is needed to ensure the data integrity of logical volumes with multiple copies, and the VGDA references below should be interpreted as meaning both VGDA and VGSA.

A single disk volume group contains two VGDAs on the disk. In a two-disk volume group, one disk contains two VGDAs and the second disk contains a single VGDA. Any volume group containing three or more disks has single VGDAs on each disk. If you add one or more disks to an existing two-disk volume group, the VGDAs are redistributed so that there is only one on each disk.

In a two-disk volume group, if you lose the disk containing two VGDAs and quorum is on, you will lose quorum, the volume group will vary off, and data will no longer be accessible, even though mirroring may be in place. In both three and four-disk volume groups, you can lose only a single disk for the volume group to remain active since you must at all times have 50+% of the VGDAs available.

It is possible to turn quorum off for a volume group, and this should be done only when mirroring is in place. This effectively means that you could lose all disks except one and the volume group would still remain active, although the data may not be usable, depending on the particular application. Under these circumstances there is also the possibility of data corruption.

In AIX, mirroring is at the logical volume level with either one or two mirrored copies possible. All copies should be located on separate disks, both from an availability and performance perspective, although this is not essential; it is possible to place all the copies on a single disk but this defeats the object of mirroring. Having three copies obviously provides greater protection, but offset against this is the increase in cost. Normally two copies are sufficient, but under certain circumstances it may be desirable to have three – for example, to take one copy temporarily off-line to perform back-ups and still leave two copies to give protection against data loss. After the back-up is performed the third copy is reintegrated and synchronized.

Ideally, in an HACMP environment mirroring should always be present and quorum should always be on in order to protect data integrity, but, as we shall see, this may not always be possible owing to costs constraints allied to the number of disks contained in the volume group. Unfortunately, in volume groups which contain just two disks (one disk mirrored), this could lead to a potential reduction in availability if quorum remained on and the disk containing the two VGDAs were lost.

Although the two-disk volume group is a special case, in all other cases the balance has to be drawn between the need for availability and the potential for data corruption should quorum be switched off. This is especially true when an even number of disks are spread evenly across only two disk subsystems, where the loss of one subsystem will automatically mean the loss of quorum.

If quorum is intended to be on, disks can be arranged across SSA subsystems in the following configurations to ensure that a volume group remains active in the event of a subsystem failure. It should be remembered that mirroring will be at the logical volume level and not

at disk level, although in the following examples it is assumed that all logical volumes on each disk are mirrored to the same or similar locations on the corresponding mirrored disks.

It should be noted that in the following examples the volume groups should not be considered in isolation. While it may be considered expensive, and indeed frivolous, to have subsystems containing just single or small numbers of disks, cluster nodes are quite often configured with multiple volume groups used by several applications, so that disk subsystems may be heavily populated. In these situations the additional costs of extra subsystems and quorum-buster type disks (see below) can often be justified since they represent a relatively small increase in cost compared with the whole.

TWO-DISK VOLUME GROUP

A two-disk volume group is shown in Figure 1.

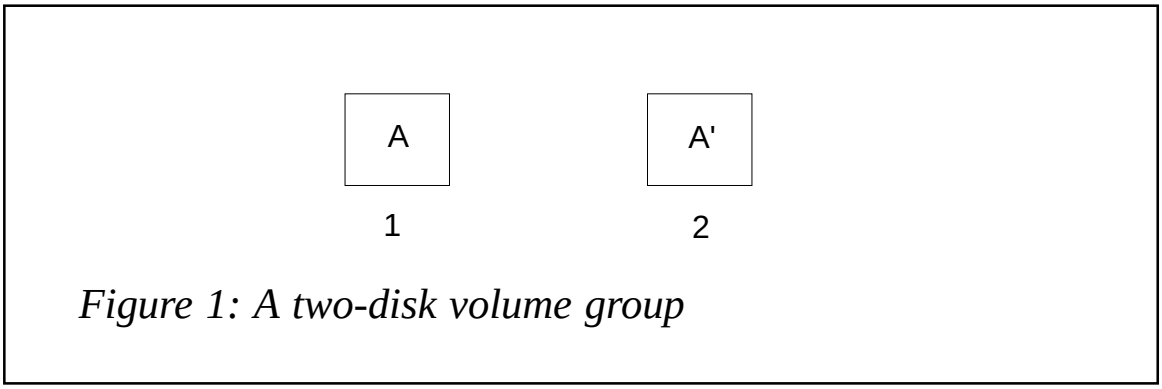


Figure 1: A two-disk volume group

For the reasons discussed above, it is not possible to arrange two disks across two subsystems to ensure that a volume group will always remain active if a subsystem fails while quorum is turned on.

It is recommended that quorum be turned off since this will allow the volume group to continue to be accessed no matter which disk fails. Data corruption is unlikely to occur since if a further disk is lost all disks will be missing and no data can be written or read.

In the above situation it is essential that the system clocks between all cluster nodes are accurate so that, should there have been an HACMP takeover prior to the subsystem rejoining, and the missing subsystem

again becomes available either during or after the takeover, re-synchronization of the logical volumes will be performed from the accurate up-to-date copy to the stale copy, and not the other way round.

The alternative to turning quorum off is to add a third quorum-buster disk to the volume group and locate it in a third disk subsystem. Although this will involve added expense, it would then be possible to utilize this extra disk by spreading the logical volumes across the three disks, as shown in the example below, using odd numbers of disks.

FOUR-DISK VOLUME GROUP

A four-disk volume group is illustrated in Figure 2.

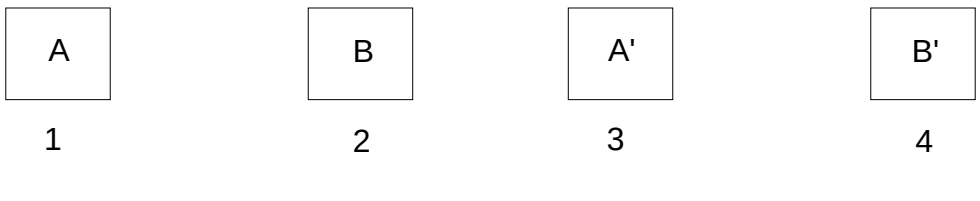


Figure 2: A four-disk volume group

A four-disk volume group must be spread across four subsystems to ensure that the volume group remains active no matter which subsystem fails. For a single volume group, this is a very expensive configuration; a much cheaper solution would be to use two subsystems and add a quorum-buster disk. The disks could then be spread across two subsystems in a 3-2 configuration.

The quorum-buster disk can be added to any of the following even-disk volume group configurations if the intention is to keep quorum on and reduce the number of subsystems to two. Having an odd number of volume group disks spread across an even number of subsystems means that you cannot have strict single-disk-to-single-disk mirroring of the logical volumes if you want to make full use of the quorum-buster.

SIX-DISK VOLUME GROUP

A six-disk volume group is illustrated in Figure 3.

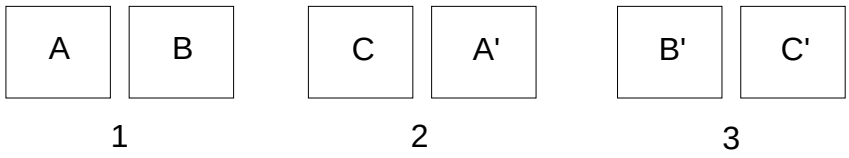


Figure 3: A six-disk volume group

A six-disk volume group must be spread across three subsystems to maintain quorum in the event of a subsystem failure.

EIGHT-DISK VOLUME GROUP

An eight-disk volume group is illustrated in Figure 4.

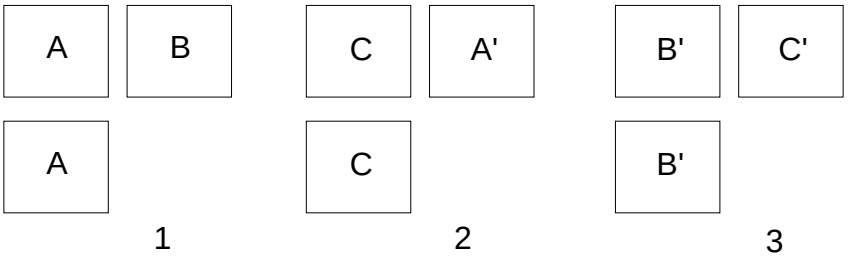


Figure 4: An eight-disk volume group

An eight-disk volume group must also be spread across three subsystems to maintain quorum in the event of a subsystem failure.

QUORUM AND MIRRORING DISK COMBINATIONS

Figure 5 summarizes the ideal locations of even numbers of volume group disks and their corresponding mirrors (assuming one-to-one mirroring at logical volume level), so that, apart from the two-disk situation, quorum can remain on.

Total disks	Disks in each subsystem				Quorum
	1	2	3	4	
2	A	A′			off
4	A	B	A′	B′	on
6	A, B	C, A′	B′, C′		on
8	A, B, C	D, A′, B′	C′, D′		on
10	A, B, C, D	E, A′, B′	C′, D′, E′		on
12	A, B, C, D	E, F, A′, B′	C′, D′, E′, F′		on
14	A, B, C, D, E	F, G, A′, B′, C′	D′, E′, F′, G′		on
16	A, B, C, D, E, F	G, H, A′, B′, C′	D′, E′, F′, G′, H′		on

Figure 5: Volume group disks and their mirrors

MIRRORING WITH ODD NUMBERS OF DISKS

The examples above have all assumed an even number of disks, where it is possible to mirror all the logical volumes on a disk to the exact same locations on the corresponding mirrored disk. It is also possible to provide mirroring, with quorum turned on, across an odd number of disks, but in such a situation careful thought must be given to the placement of the logical volumes on each disk. We cannot use strict one-to-one disk mirroring and Figure 6 illustrates this.

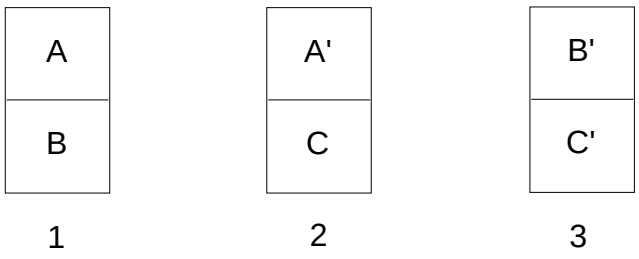


Figure 6: Mirroring with an odd number of disks

Figure 6 represents three logical volumes mirrored across disks in three separate subsystems in such a way as to maintain quorum in the event of a subsystem loss, and also to allow access to any of the logical volumes no matter which subsystem fails. This design principle can

be extended to a volume group containing any odd number of disks, although, depending on the size and number of logical volumes, a 2-2-1 or 3-2-2 combination may present greater challenges than a 3-3-3 or 5-5-5 configuration.

Having an odd number of disks (apart from 1) has the advantage that only three subsystems need ever be used (provided you don't require more than 48 disks), but the added disadvantage is that much greater thought, planning, and administrative overhead is required in the placement and maintenance of the logical volumes. Increasing the size of logical volumes 12 months down the line can be a nightmare!

LOCATION OF MIRRORED COPIES IN RACK-MOUNTED SUBSYSTEM

To ensure the highest possible availability of data, mirrored copies must be located in separate SSA subsystems. If rack-mounted subsystems are used then they must be located in different disk racks. Ideally, three racks should be used, each containing a subsystem, so that in the event of failure of the power supply to a disk rack then quorum will be maintained and data will still be accessible through the mirrored copies in the remaining racks. As the number of racks and subsystems increases, you have the same problem at rack level that you had at subsystem level, ie in which racks should I locate my subsystems to maintain the highest level of availability? This sort of complexity frequently occurs in SP systems containing multiple HACMP clusters requiring access to a number of racks shared by large numbers of nodes.

CONFIGURING LOGICAL VOLUMES ACROSS MULTIPLE DISKS

In a volume group containing either odd or even numbers of disks, logical volume copies can span multiple disks in a single subsystem, since, if a single disk containing part of the logical volume is lost, or a complete disk subsystem fails, a mirrored physical partition copy will always be accessible in another subsystem.

Configuring volume groups containing an uneven distribution of disks across subsystems can become much more complex, particularly

when you want to use more than half the disks in the volume group (perhaps you have space problems and this is your only, or temporary, solution) since the location of the partitions and their mirrored copies must now be considered at the physical partition level and the use of partition map files is usually required. My advice is to avoid this type of configuration like the plague!

When you want to increase the size of a logical volume which spans multiple disks in a high availability environment, in many cases you cannot allow AIX to extend it simply by adding a number of partitions, since the operating system, despite its best intentions, may locate the partitions on the wrong disks in the wrong subsystem, with the potential of reducing your data availability. In such cases a carefully planned distribution of partitions will be required to ensure that the logical volume is totally accessible if a disk or subsystem fails. If you are using three mirrored copies then the complexity increases even further.

This type of configuration should be installed only after careful planning since there could be future performance implications, in addition to the administrative and technical complexity of maintaining such a system. In any installation where your data is expected to grow substantially you should plan your logical volume locations and sizes so that you can cope with at least two years' data growth without the need to reorganize your volume groups across your disks – if only management would allow us to use such foresight, which unfortunately costs more than they are usually prepared to spend!

Tonto Kowalski
Guru (UAE)

Have you come across any undocumented features in AIX 5L? Why not share your discovery with others? Send your findings to us at any of the addresses shown on page 2.

An introduction to emacs

WHAT IS EMACS?

What does emacs stand for? To some it is ‘emacs makes a computer slow’, to others, ‘escape, meta, alt, control, shift’, which is silly because it leaves out ‘hyper’ and ‘super’, amongst others.

There is a long-standing holy war among AIX users (and other Unix users in general) as to whether they prefer vi or emacs. It is normal to claim that vi will be found everywhere while emacs will not. While this itself is only a partial truth, in the process of arguing for vi, however, people neglect to mention that emacs is much more than a text-processor. The difference starts from the ground up, that text editing is vi’s entire *raison d’etre*, while to emacs it is only a side effect. This means it is far more efficient to keep one emacs session open all the time, unlike the normal use of vi.

In this introduction for AIX users unfamiliar with it, we will examine emacs in more detail.

VERSIONS OF EMACS

There are two projects providing emacs – GNU emacs and Xemacs. In the wild, one encounters mostly emacs20 (the most common), emacs21 (the most recent, released only a month ago), and xemacs21.4. Xemacs forked away from emacs many years ago, and is a separate project. It differs in being the first to have a full GUI with a toolbar, an extended emacs LISP function-set as standard, and a complete ‘customize’ user-interface, whilst still maintaining (for the most part) compatibility with emacs. Note that the name Xemacs does not imply that it requires the X-window system in order to run, merely that there is a strong bias towards running in that environment (one has to disable options at build-time to get a terminal-only version).

The major changes in emacs21 over emacs20 have been the addition of a toolbar, a much more extensive customize menu system, and colour support in console mode.

First, read the splash-screen when Emacs starts. There are lots of other

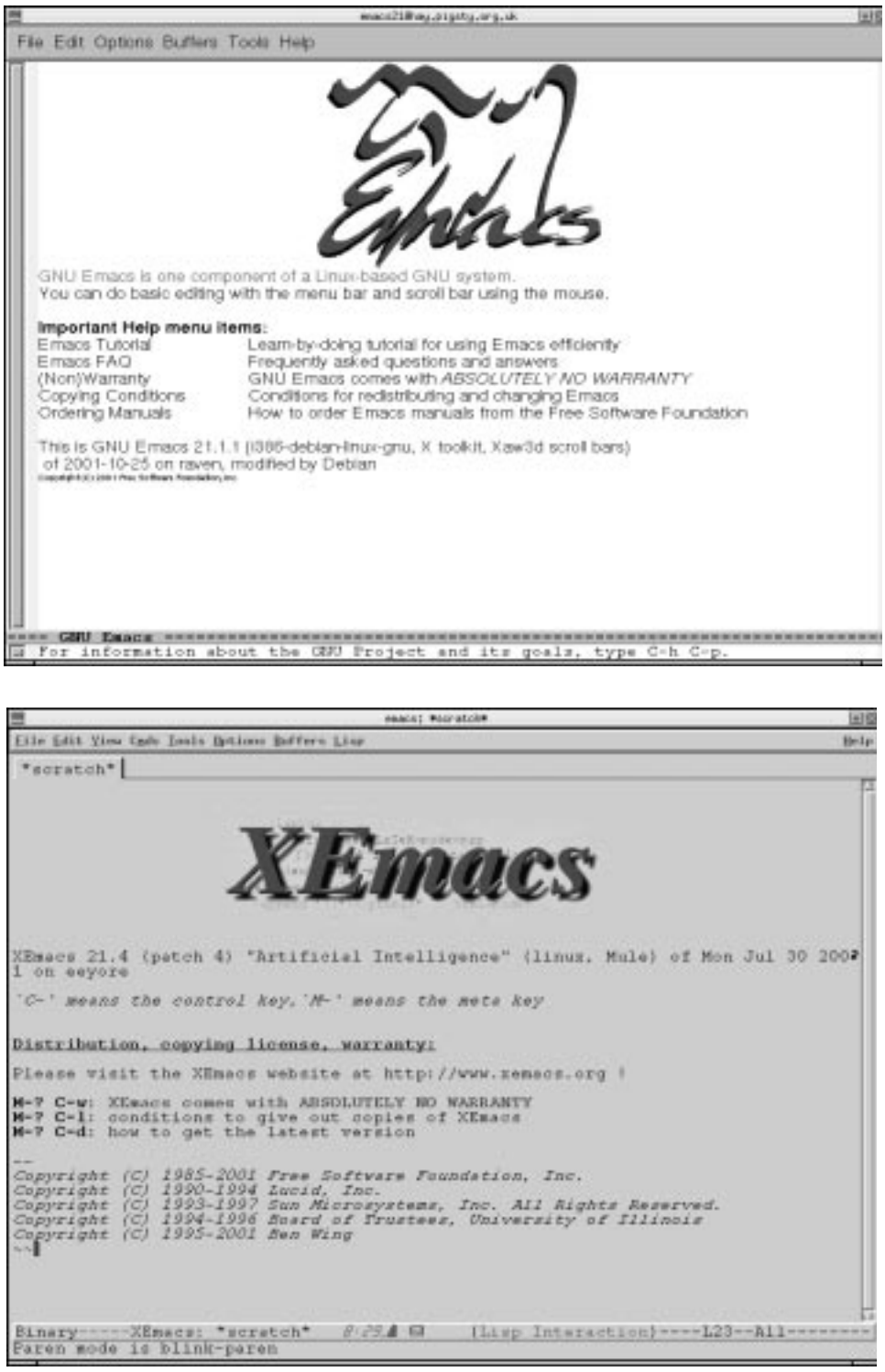


Figure 1: Splash-screens

reference works including an FAQ and a very detailed tutorial, both of which are alluded to in the splash-screen – see Figure 1.

A QUICK TOUR OF EMACS FOR AIX VI USERS

Modes

Emacs is a highly modal editor. If you are used to vi with its *insert*,

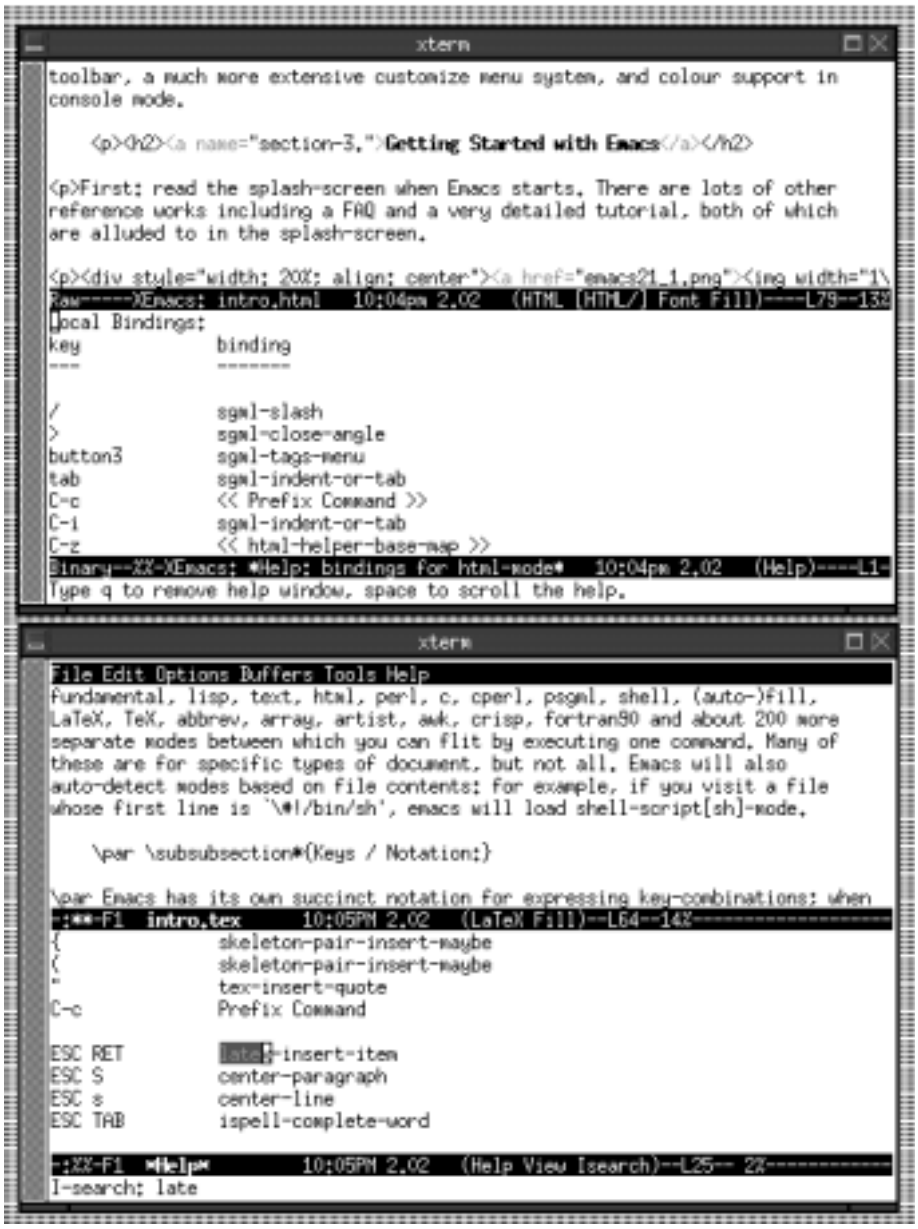


Figure 2: Xterms running Xemacs and emacs21

command, and one or two other modes, prepare to be blown away by fundamental, LISP, text, html, perl, c, cperl, psgml, shell, (auto-)fill, LaTeX, TeX, abbrev, array, artist, awk, crisp, fortran90, and about 200 additional separate modes between which you can flit by executing one command. Many of these are for specific types of document, but not all. Emacs will also auto-detect modes based on file contents: for example, if you visit a file whose first line is '#!/bin/sh', emacs will load shell-script[sh]-mode.

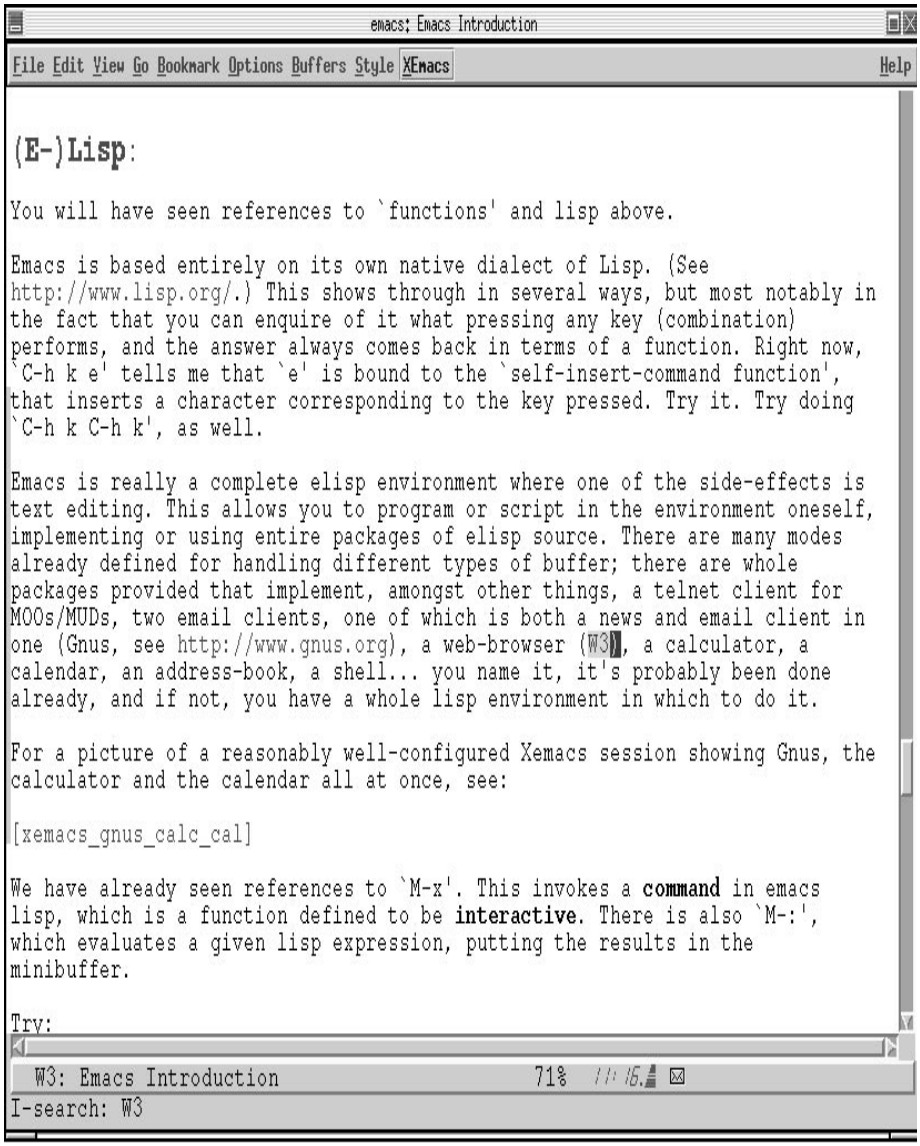


Figure 3: Xemacs in use as a Web browser

Figure 2 shows two xterms, one running Xemacs, the other emacs21, with Xemacs viewing this document in HTML format (hence, in HTML mode), and emacs21 viewing it in LaTeX format, hence in LaTeX mode.

Figure 3 shows Xemacs in use as a Web browser, searching through the text of the current document.

One thing to note with the programming and SGML modes is that the TAB key is frequently used to re-indent the current line to the correct location, rather than merely inserting a fixed number of spaces every time. Thus, something like:

```
<html>
<head>
</head>
</html>
```

can be converted into:

```
<html>
  <head>
  </head>
</html>
```

by pressing TAB anywhere on the middle two lines; once TAB is pressed, the line jumps to the correct indentation and further TABs will have no effect.

Keys/notation

Emacs has its own succinct notation for expressing key combinations – when you see *C-x C-f* it means you press control for both *x* and *f*. If you didn't, you wouldn't get the same consequences, as one finds a (new) file, while the other sets the width of a line for wrapping. Semi-confusingly, *M-* ('meta') is the escape key, but it might also be bound to Alt on your keyboard; you press escape before the modified key, but Alt with it. The notation is that *C-* means press control at the same time, *S-* for shift, *A-* for Alt, *H-* for hyper, *M-* for meta. (Several keys pressed together in this way are known as a chord.) These can be combined, for example, *C-x h M-C-* selects the whole buffer and indents it all according to the current mode (the second chord being achieved by pressing Escape first, then control+\).

There are several keys without which basic use is impossible:

- *C-x C-c* – quit.
- *C-g* – abort whatever you're trying to do; terminate current function, and remove current highlighted region.
- *C-x C-f* – open new file.
- *C-x C-s* – save current file.
- *M-<* and *M->* – move to start/end of buffer.
- *C-x C-w* – write current buffer to a new file.
- *C-a* – go to beginning of line.
- *C-e* – go to end of line.
- *C-S-_* – undo.
- *C-k* – cut to end of line.
- *C-y* – yank from cut-buffer (equivalent to paste).
- *C-w* – kill the current region into cut-buffer.
- *C-u* – numerical prefix argument, repeater.
- *C-x (* and *C-x)* – keyboard macro start, end .
- *C-h -* – entrance to the help system.
- *C-l* – re-centre the display around the current line.
- *C-q* – quote the next character, eg to insert a control character.

A word of explanation about the *C-u* key, above: if you press it immediately before another key that inserts a character, you will get that key pressed four times, eg:

```
C-u *
```

comes out as:

****

If you insert a number between the *C-u* and a character, then that many of the character will be printed – so:

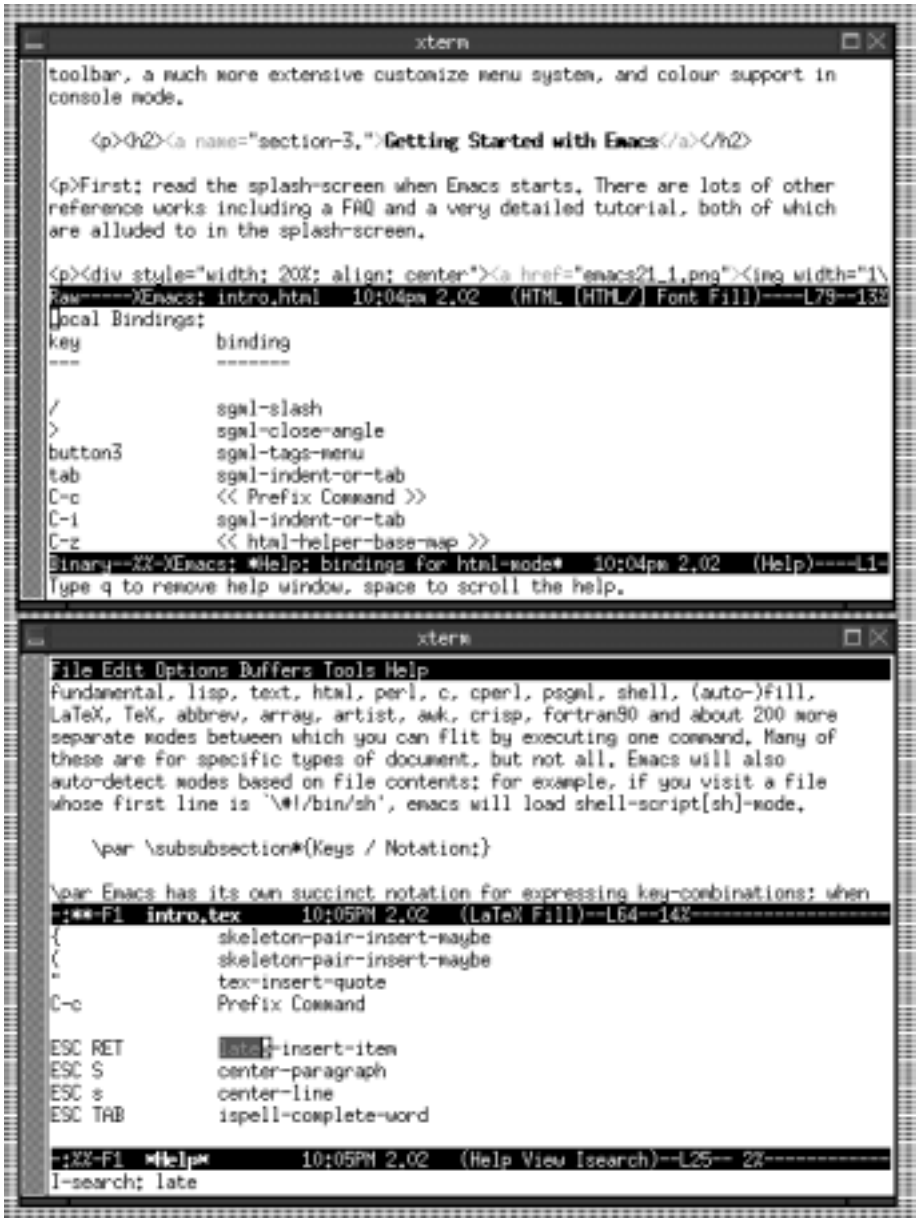
```
C-u 5Ø #
```

comes out as:

```
############################################################
```

Buffers and frames

A buffer is a view onto a file. A frame is analogous to a ‘window’, in



S

Figure 4: Xterms running Xemacs and emacs21

the Netscape 'Open new window' sense. Buffers are portable between frames in the same emacs session.

You can clone a frame by pressing *C-x 5 2* – this will duplicate the current frame showing the same buffer that you were visiting (X interface only – not applicable in a terminal or on console). A common action is to spawn a second frame and immediately open a file in it; for this purpose you can press *C-x 5 f* instead.

Figure 4 shows two xterms, the upper one running Xemacs, the lower with emacs21.

Both are viewing a document: in the Xemacs frame, the HTML source is displayed, while in emacs21 it's in LaTeX format, the point being that the status bar shows different modes for the different file formats. Both frames have been split into two buffers, the lower of which shows the keys available in the current mode. (For example, pressing TAB in Xemacs would indent the current line to the correct level corresponding to depth of tag.)

Regions, rectangles, and narrowing

To begin marking out a region to be operated on later, press *C-space*. The mini-buffer will say 'Mark set'; now, as you move around, that mark is still anchoring one end of the region, while your current location is the other end.

You can press *C-w* to kill the region; this will make it available for later use pasted in elsewhere with *C-y*.

Rectangles are similar to regions, except that, instead of always running to the end of a line before moving to the next, actions on them apply only to the rectangle marked out by the two corners. For example:

Source: Results of pressing *C-x r c* (clear-rectangle):

```
AAAAAAAAAAAAAAAA    AAAAAAAAAAAAAAAA
BBBBBBBBBBBBBBBB    BB            BBB
CCCCCCCCCCCCCCCC    CC            CCC
DDDDDDDDDDDDDDDD    DD            DDD
EEEEEEEEEEEEEEEE    EE            EEE
FFFFFFFFFFFFFFFF    FFFFFFFFFFFFFFFF
```

Obviously, calling *C-x r k* (kill-rectangle) would have removed the contents of the rectangle instead.

Here we come across one of the differences between emacs and Xemacs; if using a default emacs installation, you'll have had to take it on trust that it really was marking out a region or rectangle (after saying 'Mark set' in the mini-buffer), whereas in Xemacs it highlights the region as it goes.

Narrowing is the process of restricting your view of a buffer to just the current region, and obviously widening means restoring the view to the whole buffer. Open a file using *C-x C-f*, use *C-space* to start marking a region, move to the other end of a paragraph using *C-up*

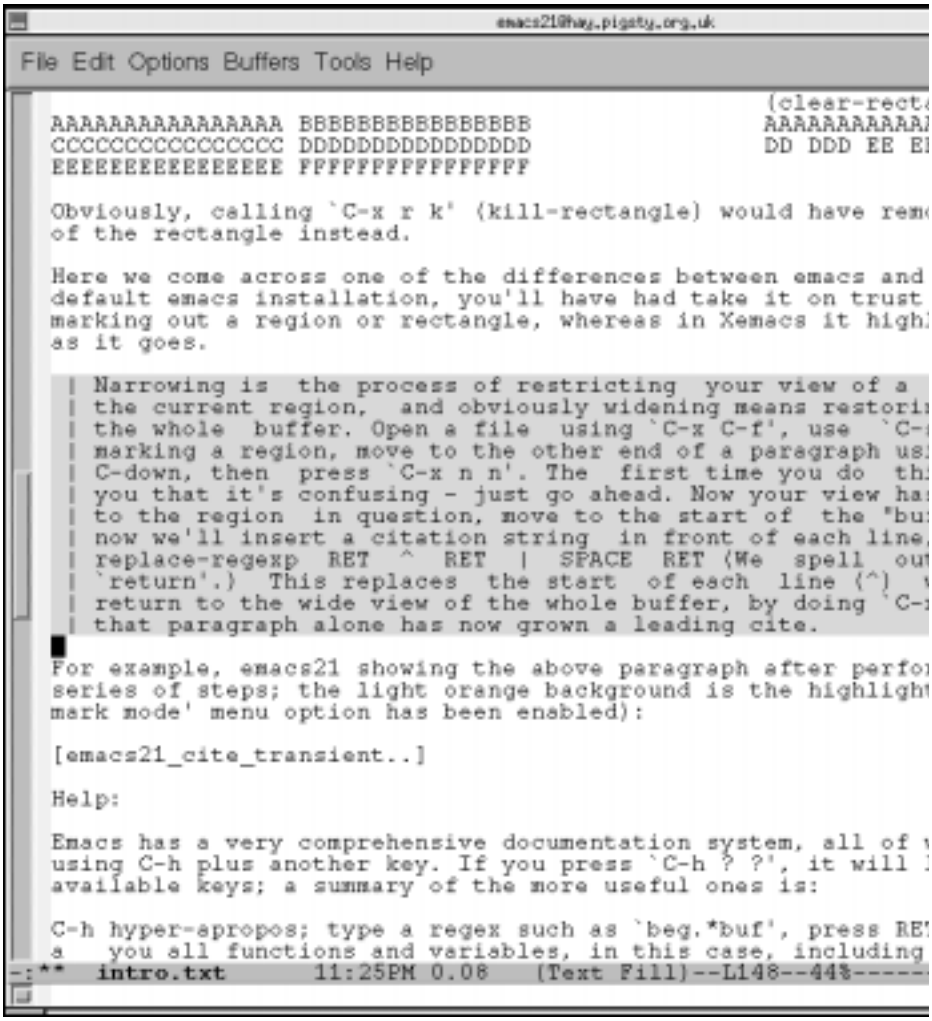


Figure 5: Emacs21 results

and/or *C-down*, then press *C-x n n*. The first time you do this, emacs warns you that it's confusing – just go ahead.

Now your view has been narrowed to the region in question, move to the start of the 'buffer' with *M->*; now we'll insert a citation string in front of each line, by typing:

```
M-x
replace-regexp RET ^ RET | SPACE RET
```

(We spell out 'space' and 'return'.) This replaces the start of each line (^) with '| '. Now return to the wide view of the whole buffer, by doing *C-x n w*. That paragraph alone has now grown what's know as a leading 'cite', such as one would use when quoting in an e-mail.

For example in Figure 5, emacs21 is showing the above paragraph after performing the same series of steps; the light background is the highlight (the 'transient mark mode' menu option has been enabled).

Help

Emacs has a very comprehensive documentation system, all of which is accessed using *C-h* plus another key. If you press *C-h ? ?*, it will list all the available keys; a summary of the more useful ones is shown below:

- *C-h a* – hyper-apropos; type a regex such as 'beg.*buf', press RET, and it will show you all functions and variables, in this case, including 'beginning-of-buffer'.
- *C-h i* – enter Info mode; this is the exact same mode that the 'info' command would give, when executed from the shell.
- *C-h f* – presents documentation on the given function.
- *C-h b* – lists key combinations and the functions they perform in the current mode/environment.
- *C-h k* – displays the function that pressing a given key will evaluate.

Fill

Here we introduce a small stream of useful key presses and functions.

Take the text document you used in the *Narrowing* section above. Type *C-u 70 C-x f*. This invokes the function set-fill-column with an argument of 70, meaning that text will now be wrapped at that column. From somewhere within the paragraph you edited, press *M-q*. This will reformat the paragraph to be at most 70 columns wide. If you have emacs20 or Xemacs21, this will probably have messed up the paragraph so that the citation pipe characters are all over the place. Undo the damage by pressing *C-_*. Move to the start of one of the lines (*C-a*), then move right 3 characters so that the cursor is underneath the first letter of the first word, then press *C-x .*. This sets the fill-prefix to '|', so you can repeat the *M-q* and it will no longer mess up the whole paragraph. Any pattern of text can be used as the prefix; if the line already begins with the prefix, all to the good, otherwise it will be inserted. You can reset the fill-prefix by moving to the start of the line with *C-a* and pressing *C-x .* again.

As an aside, you can right-justify the text by using *C-u M-q* instead of just *M-q*.

Editor's note: this introduction will be concluded next month.

Tim Haynes
Open Source and Free Software Consultant (UK)

AIX Update on the Web

Code from individual articles of *AIX Update* and complete issues in PDF format can be accessed on our Web site at:

http://www.xephon.com/aix

You will be asked to enter a word from the printed issue.

The p690, eLiza, and the future of AIX

INTRODUCTION

On 5 October 2002, IBM unveiled the pSeries 690, code-named Regatta. In this article we will review the features and strategic directions of this latest high-end pSeries, and we will look at the implications that the new box will have for AIX.

AIX 5L Version 5.1 is unusual for software in that it provided support and functionality for the new pSeries six months in advance of the release of the new boxes. It is much more usual for software to lag behind hardware functionality.

The first striking point about the p690 is the wealth of functionality and characteristics that the box has inherited from the mainframe world. The most noticeable is the highly granular dynamic logical partitioning. This sharing of mainframe technologies is going to be seen across the IBM eServer range, and the pSeries and the xSeries (Summit Architecture) are the first of the non-mainframe eServer range to deploy this functionality. And it should be noted that these links are not just skin-deep: a large number of IBM mainframe staff have been 'borrowed' for the development and maintenance of the pSeries. In addition to mainframe technology, IBM has also included some technology borrowed from its Project eLiza, its new strategy for self-managing systems, which anticipates potential failures and takes automatic by-pass actions. It is eLiza which is going to dictate the directions of AIX.

IBM is also talking about the intention to deliver a mainframe-like Capacity Upgrade on Demand (CuoD) capability, which will be known as the 'Capacity Advantage' and supports on-the-fly activation of previously installed inactive processors and memory to active partitions, without requiring the box to go off-line.

CPU

The p690 is powered by up to 32 POWER4 microprocessors with two

CPUs with associated Level 2 cache memory on each chip. The POWER4 is IBM's latest generation of chipset using copper chip technology, running on silicon-on-insulator.

The POWER4 processors come in Multchip Modules (MCMs) similar to those used in S/390 G5/G6 and z900 mainframes. A p690 MCM contains either four or eight POWER4 processors, packaged on four chips, as well as 5.6MB of L2 cache shared between the processors. The L2 cache is distributed across the four chips. In addition, each chip has 128MB of L3 cache. The MCMs go inside a Central Electronic Complex (CEC). A CEC consists of one to four MCMs with associated cache memory, 8GB to 256GB of system memory, and a service processor.

The four-processor MCM is the High Performance Computing (HPC) option and is targeted at customers who have floating-point intensive applications. The eight-processor MCMs are optimized to handle both commercial and technical workloads.

PARTITIONING

Raw horsepower in the pSeries servers competes well against competitive products from Compaq, Hewlett-Packard, and Sun. However, in the past, the lack of multiple domain support has been a weak spot for AIX servers. Unix can experience problems with mixed workloads, where software or process errors can cause downtime, but AIX 5L Version 5.1 in conjunction with the new pSeries solves this by providing the mainframe-based logical partitioning, which exploits IBM's 25 year experience in software and hardware-based mainframe partitioning.

With logical partitioning, the processors, memory, and storage capabilities of a computer are grouped into multiple sets of resources that can be operated independently, each running their own operating system and applications.

The p690 supports 16 partitions, but there is no minimum number of processors per partition so it is possible to have partitions running on a single processor. This makes the partitioning highly flexible and

granular. The p690 partitioning scheme called 'Virtual Servers' is reminiscent of the original mainframe 'cloning' approach seen in VM (Virtual Machine) in the 1970s, and is similarly implemented using software. But it should be noted that since the early 1990s IBM has adopted a hardware-focused approach for mainframe partitioning using PR/SM (Processor Resource/System Manager). It is therefore likely that in the future the hardware partitioning scheme for the pSeries is also going to follow this hardware-centric route. Currently, the partitions can run either AIX 5L (Version 5.1 with the 5100-1 maintenance package) or Linux. It is expected that 64-bit Linux for the p690 partitions will be available from Linux vendors by the end of 2001 or early 2002.

CLUSTERING

IBM has been noticeably reluctant to exceed 32 processors with the p690. However, it is clear that the company did explore the possibility of developing a 128-processor machine, but abandoned this configuration because of performance degradation issues. The reason for this is the phenomenon of 'non-linearity' found in Symmetric Multi-Processor (SMP) designs. There comes a point when adding extra processors to increase system horsepower actually leads to diminishing returns. This is why IBM uses Parallel Sysplex on mainframes and the multi-node architecture in Massively Parallel Processing (MPP) systems such as the RS/6000 SP2. Certainly, large-scale multiprocessor systems are an area in which IBM has considerable experience.

Parallel Sysplex-like clustering is IBM's mid- to long-term strategic direction for increasing the scalability of the p690 systems. This is the reason for the AIX-based Parallel System Support Program (PSSP), which enables up to 16 p690 servers to be clustered together and managed as a single entity from a central point of control. In the first half of 2002 IBM will extend this clustering scheme further to permit clustering between p690s and existing RS/6000s, including the SP2 systems. It is expected that this functionality will find its way to the smaller boxes in the pSeries range in the short- to mid-term future.

The most important element of the new pSeries design is the incorporation of eLiza hardware and software functionality. The future of AIX is now intimately bound with that of the eLiza Project. eLiza is actually short for ‘electronic Lizard’, and is based on the autonomic nervous system and brain functions of an adult lizard.

The eLiza initiative was announced in March 2001, with the stated aim of making complex heterogeneous server environments easier to manage. To do this eLiza focuses on four system attributes: self configuration, self-optimization, self-protection, and self-healing. The goal is to produce future servers that require little or no operator intervention. However, at first the self-managing self-healing technologies will be deployed on the hardware, operating systems, and microcode, and in the future will begin to spill over into IBM’s storage products and middleware applications such as MQseries, CICS, and the Tivoli product set.

The project is certainly one of the most extensive of its type, with 25-50% of the Enterprise Server Group’s two billion dollar research and design budget being spent on eLiza. It will be the largest single focus of investment for IBM in the short- to mid-term future, dwarfing the one billion dollars that IBM has devoted to Linux. It also encompasses technologies ranging from chips to middleware to operating systems. Because of its wide-ranging nature the initiative involves thousands of personnel from IBM’s Research Division, working in conjunction with the pSeries Group and the other Enterprise Server Groups. Although the server division will drive the initiative, considerable support will have to come from the software division. This mammoth effort is coordinated by a new eLiza group.

Examples of eLiza technologies and functions include ‘system-wide error survivability’, which enables the system to keep running even if components fail. To achieve this, failed components are taken off-line automatically. Another eLiza technology is a diagnostic capability called ‘error safeguard’, which is designed to detect problems early in their life cycle, and therefore warn operators before they can escalate. There is also functionality designed to determine the cause of system failures before they can bring about more errors and malfunctions.

AIX 5L Version 5.1 also incorporates a range of self-healing technologies, including over 5,000 sensors to monitor performance and system health. These help to contain failures so that the system can be kept running even if some malfunctions occur. A memory-scrubbing function checks for memory errors and makes a note of bad data, to prevent such data from being used or reused.

IBM has provided a roadmap of eLiza functionality. Functions such as authentication and management of distributed applications performance and end-to-end automation will not be deployed on all eServer platforms until 2003. In addition to self-healing, eLiza encompasses dynamic workload management across heterogeneous systems, super-scalable clusters (with a potential for thousands of servers), and distributed server management for hundreds of servers. In the short- to mid-term we can expect IBM pServers to:

- Activate built-in redundant components when failures occur.
- Automatically balance bandwidth or application capacity when necessary.
- Monitor for intrusions.
- Cluster with other servers on-the-fly to balance workload and for failover, redundancy, and increased availability.
- Automatically configure and install operating systems, applications, and data.

As can be seen above, eLiza is not just about hardware; it will require software functionality such as enhanced workload management, clustering, and security to be built into the operating systems such as AIX.

The current mainframe environment encompasses many of the desirable attributes of eLiza, but in an essentially homogenous environment. eLiza will encompass the heterogeneous IT environment. We will see the incorporation of hardware such as the Intelligent Resource Director, which allocates resources to jobs according to demand. The fact that much of the functionality is already used on the mainframe platform means that the time-to-market for the eLiza functionality could be reduced.

AIX 5L VERSION 5.2

AIX is once again in a period of transition. By providing support for Intel's new 64-bit IA-64 architecture, beginning with the Itanium processor, AIX 5L Version 5.1 was the first version of AIX to support platforms other than IBM's Power-based hardware. This was a result of Project Monterey, the alliance with Santa Cruz Operation (SCO) to develop AIX 5L. Project Monterey involved porting IBM's new 64-bit AIX kernel to IA-64, endowing it with a number of user-level features from SCOs UnixWare operating system, and then porting the result back to the Power architecture. Much of the 'open' functionality of AIX 5L was the result of Monterey. We provided a review of AIX 5L in *AIX Update* Issue 65, in March 2001, pages 7-11.

Now the deployment of eLiza functionality will once again result in more changes to AIX. In the short-term future (mid-2002), AIX 5L Version 5.2 will acquire the following functionality:

- Advanced system scalability.
- NUMA/SMP performance tuning.
- McKinley enabling for Itanium-based platforms.
- RAS enhancements.
- Dynamic partitioning support.
- System management enhancements.

Some of this new functionality such as the systems management enhancements will be a direct result of the incorporation of eLiza functionality.

LINUX

IBM has always considered AIX to be its strategic enterprise-class Unix operating system, while it has recognized Linux as being its strategic, open, volume operating system. As such, both operating systems will address distinct but complementary goals and requirements for the indefinite future. However, Linux is clearly establishing the *de facto* standard for Unix APIs. By supporting Linux

APIs, and the necessary tools for developing Linux applications and managing Linux systems, IBM will potentially boost the AIX application base significantly, because many Unix developers will increasingly target the Linux platform first.

Despite the surge of Linux in the Unix-on-Intel space, a clear need remains for high-end enterprise-class Unix functions which can run on a variety of industry-standard hardware. In the meantime, AIX 5L Version 5.1 takes maximum advantage of IBM's existing Power-based systems.

FUTURE IMPLICATIONS

AIX 5.1 has already made substantial improvements to its system management capabilities through enhancements related to event management, storage management, and remote administration. With the future deployment of more self-managing technologies, this will reduce the need for systems administrators to purchase, install, and configure (expensive) enterprise-management frameworks such as CA Unicenter, HP OpenView, or Tivoli TME10. These frameworks still have a role in managing large networks of hundreds of systems or of heterogeneous systems. Integrating this capability into the base operating system allows a greater range of system-specific information to be gathered and tracked, since the native mechanism can track parameters at far greater levels of detail.

Some may consider self-managing systems to be a threat to AIX systems programmers. But it is possible to see some of this functionality (for example IRD) at work in current z/OS mainframe systems, and here the self-managing capabilities simplify system configuration definition, reducing the skills required to manage z/OS. This means that operators can actually spend more of their time running the business rather than running the servers. Self-management does not usually result in fewer programmers.

CONCLUSIONS

The self-managing capabilities that will become a part of AIX and other eServer Operating Systems will fulfil an essential need in the

enterprise. The eLiza-enhanced AIX and pSeries will provide valuable differentiation in the highly competitive server market where hardware cost will become much less of a meaningful selling point.

In the coming years we will see an increasingly rapid pace of technological deployment, in conjunction with massive increases in data volumes. This needs to be viewed with the increasing cost of skills, and the increasing liabilities associated with unplanned system downtime. eLiza-enabled AIX will allow users to manage IT environments that are hundreds of times more complex than those of today.

It is probable that the initial deployments of the new pSeries systems running AIX version 5.1 will be used primarily for server consolidation, but, in the future, we may see applications as diverse as multimedia, e-commerce, and data warehousing, which require huge amounts of processing power, being deployed on the platform.

Systems Programmer (UK)

Mirrored-disk recovery

This article offers some advice on how to deal with a failed disk in a mirrored volume group. This particular case involves a mirrored rootvg volume group.

The steps that were originally taken to mirror rootvg (member disks hdisk0 and hdisk1) were the following:

```
[root@my-system]/> mirrorvg rootvg            (mirror all LVs in the VG)
[root@my-system]/> bosboot -ad /dev/hdisk1
                                        (make bootblock on mirror disk)
[root@my-system]/> bootlist -m normal hdisk0 hdisk1
                                          (add hdisk1 to bootlist)
```

First we will display the volume group as it appeared before the disk failed.

```
[root@my-system]/> lspv (list the physical volumes on the system)
hdisk0          000b5a2d9dc260b7    rootvg
hdisk1          000b5a2dcae1a63f    rootvg
hdisk2          000b5a2dcae04016    datavg
hdisk3          000b5a2dcae0b2f1    datavg

[root@my-system]/> lsvg rootvg
VOLUME GROUP:   rootvg          VG IDENTIFIER:  000b5a2d129473e6
VG STATE:       active          PP SIZE:        16 megabyte(s)
VG PERMISSION:  read/write      TOTAL PPs:      1084 (17344 megabytes)
MAX LVs:        256             FREE PPs:       808 (12928 megabytes)
LVs:            12              USED PPs:       276 (4416 megabytes)
OPEN LVs:       10              QUORUM:         1
TOTAL PVs:      2               VG DESCRIPTORS: 3
STALE PVs:      0               STALE PPs:      0
ACTIVE PVs:     2               AUTO ON:        yes
MAX PPs per PV: 1016            MAX PVs:        32

[root@my-system]/> lsvg rootvg -prootvg:P
V_NAME      PV STATE    TOTAL PPs   FREE PPs    FREE DISTRIBUTION
hdisk0      active      542         402         108..68..09..108..109
hdisk1      active      542         406         108..72..09..108..109
```

Here is a listing of the logical volumes in the volume group:

```
[root@my-system]/> lsvg rootvg -l
rootvg:
LV NAME            TYPE       LPs   PPs   PVs  LV STATE      MOUNT POINT
hd5                boot       1     2     2    closed/syncd  N/A
hd8                jfslog     1     2     2    open/syncd    N/A
hd6                paging     32    64    2    open/syncd    N/A
paging00           paging     32    64    2    open/syncd    N/A
hd4                jfs        4     8     2    open/syncd    /
backupdirlv        jfs        1     2     2    open/syncd    /backupdir
hd1                jfs        1     2     2    open/syncd    /home
hd3                jfs        2     4     2    open/syncd    /tmp
hd2                jfs        57    114   2    open/syncd    /usr
hd9var             jfs        2     4     2    open/syncd    /var
hd7                sysdump    4     4     1    open/syncd    N/A
```

Note that, since the VG is mirrored, for each LP (logical partition) there are two PPs (physical partitions). The exception is the hd7 system dump device, by design.

The LV state is normally ‘syncd’, which indicates that the mirrored partitions are currently fully synchronized with each other.

Now the disk suffers a failure and things begin to change for the worse:

```
[root@my-system]/> lsvg rootvg -p
rootvg:
PV_NAME        PV STATE    TOTAL PPs   FREE PPs    FREE DISTRIBUTION
hdisk0         active      542         405         108..71..09..108..109
hdisk1         missing     542         409         108..75..09..108..109
```

The PV state becomes ‘missing’ instead of active:

```
[root@my-system]/> lsvg rootvg -l
rootvg:
LV NAME             TYPE       LPs   PPs   PVs  LV STATE      MOUNT
POINT
hd5                 boot       1     2     2    closed/stale  N/A
hd8                 jfslog     1     2     2    open/stale    N/A
hd6                 paging     32    64    2    open/stale    N/A
paging00            paging     32    64    2    open/stale    N/A
hd4                 jfs        4     8     2    open/stale    /
backupdirlv         jfs        1     2     2    open/syncd    /backupdir
hd1                 jfs        1     2     2    open/stale    /home
hd3                 jfs        2     4     2    open/stale    /tmp
hd2                 jfs        57    114   2    open/syncd    /usr
hd9var              jfs        2     4     2    open/stale    /var
hd7                 sysdump    4     4     1    open/syncd    N/A
```

One by one the mirrored LVs start to become stale:

```
[root@my-system]/> lsvg rootvg
VOLUME GROUP:   rootvg            VG IDENTIFIER:  000b5a2d129473e6
VG STATE:       active            PP SIZE:        16 megabyte(s)
VG PERMISSION:  read/write        TOTAL PPs:      1084 (17344 megabytes)
MAX LVs:        256               FREE PPs:       814 (13024 megabytes)
LVs:            11                USED PPs:       270 (4320 megabytes)
OPEN LVs:       10                QUORUM:         1
TOTAL PVs:      2                 VG DESCRIPTORS: 3
STALE PVs:      1                 STALE PPs:      22
ACTIVE PVs:     1                 AUTO ON:        yes
MAX PPs per PV: 1016              MAX PVs:        32
```

The VG now shows that one of the two physical volumes is ‘stale’, and 22 physical partitions as well.The error log fills up with errors related to the problem:

```
[root@my-system]/> errpt
IDENTIFIER TIMESTAMP  T C RESOURCE_NAME  DESCRIPTION
EAA3D429   1116151201 U S LVDD           PHYSICAL PARTITION MARKED STALE
EAA3D429   1116151201 U S LVDD           PHYSICAL PARTITION MARKED STALE
EAA3D429   1116151201 U S LVDD           PHYSICAL PARTITION MARKED STALE
EAA3D429   1116151201 U S LVDD           PHYSICAL PARTITION MARKED STALE
(much repetition suppressed)
```

```
EAA3D429   1116151201 U S LVDD           PHYSICAL PARTITION MARKED STALE
35BFC499   1116151201 P H hdisk1         DISK OPERATION ERROR
35BFC499   1116151201 P H hdisk1         DISK OPERATION ERROR
F7DDA124   1116151201 U H LVDD          PHYSICAL VOLUME DECLARED MISSING
52715FA5   1116151201 U H LVDD  FAILED TO WRITE VOLUME GROUP STATUS AREA
613E5F38   1116151201 P H LVDD           I/O ERROR DETECTED BY LVM
35BFC499   1116151201 P H hdisk1         DISK OPERATION ERROR
613E5F38   1116151201 P H LVDD           I/O ERROR DETECTED BY LVM
35BFC499   1116151201 P H hdisk1         DISK OPERATION ERROR
EAA3D429   1116151201 U S LVDD           PHYSICAL PARTITION MARKED STALE
613E5F38   1116151201 P H LVDD           I/O ERROR DETECTED BY LVM
35BFC499   1116151201 P H hdisk1         DISK OPERATION ERROR
```

Trying to logically remove the bad hdisk will not work since there are still references to it:

```
[root@my-system]/> rmdev -dl hdisk1
Method error (/etc/methods/ucfgdevice):
        0514-062 Cannot perform the requested function because the
                 specified device is busy.
```

We must first attempt to break the mirrors using the unmirrorvg command:

```
[root@my-system]/> unmirrorvg rootvg
0516-1246 rmlvcopy: If hd5 is the boot logical volume, please run 'chpv
-c <diskname>'
        as root user to clear the boot record and avoid a potential boot
        off an old boot image that may reside on the disk from which
        this logical volume is moved/removed.
0301-108 mkboot: Unable to read file blocks. Return code: -1
0516-1155 lreducelv: Last good copy of a partition cannot reside on a
missing disk.
```

Try again after reactivating the disk using chpv and varyonvg:

```
0516-922 rmlvcopy: Unable to remove logical partition copies from
logical volume hd2.
0516-1135 unmirrorvg: The unmirror of the volume group failed.
        The volume group is still partially or fully mirrored.

[root@mbusa-ho-db1]/> lsvg rootvg -l
rootvg:
LV NAME            TYPE       LPs   PPs   PVs  LV STATE      MOUNT POINT
hd5                boot       1     1     1    closed/syncd  N/A
hd8                jfslog     1     1     1    open/syncd    N/A
hd6                paging     32    32    1    open/syncd    N/A
paging00           paging     32    32    1    open/syncd    N/A
hd4                jfs        4     4     1    open/syncd    /
```

```
backupdirlv         jfs         1      1      1     open/syncd     /backupdir
hd1                 jfs         1      1      1     open/syncd     /home
hd3                 jfs         2      2      1     open/syncd     /tmp
hd2                 jfs         57     114    2     open/stale     /usr
hd9var              jfs         2      4      2     open/stale     /var
hd7                 sysdump     4      4      1     open/syncd     N/A
```

The unmirrorvg command was reasonably successful. It managed to unmirror most of the LVs in the volume group. Many times, it will not be so easy. In this case, the hd2 and hd9var LVs are still in an unstable state. First, we will try the chpv command as recommended by the unmirrorvg/rmlvcopy output above, in order to remove the boot image from hdisk1:

```
[root@my-system]/> chpv -c hdisk1
Ø3Ø1-1Ø8 mkboot: Unable to read file blocks. Return code: -1
Ø516-1248 chpv: mkboot failure
```

The LVs that are still stale must be unmirrored individually now:

```
[root@my-system]/> rmlvcopy hd2 1 hdisk1
[root@my-system]/> rmlvcopy hd9var 1 hdisk1
```

(Note that, depending on the circumstances and nature of the failure, the PVID of the disk may have to be used instead of the hdisk name when using rmlvcopy. In this case, the PVID of the failed disk is 000b5a2dcae1a63f, as shown in the **lspv** command earlier. The command would be rmlvcopy hd2 1 000b5a2dcae1a63f, for example.)

```
[root@my-system]/> lsvg rootvg -l
rootvg:
LV NAME             TYPE       LPs   PPs   PVs  LV STATE      MOUNT POINT
hd5                 boot       1     1     1    closed/syncd  N/A
hd8                 jfslog     1     1     1    open/syncd    N/A
hd6                 paging     32    32    1    open/syncd    NA
paging00            paging     32    32    1    open/syncd    N/A
hd4                 jfs        4     4     1    open/syncd    /
backupdirlv         jfs        1     1     1    open/syncd    /backupdir
hd1                 jfs        1     1     1    open/syncd    /home
hd3                 jfs        2     2     1    open/syncd    /tmp
hd2                 jfs        57    57    1    open/syncd    /usr
hd9var              jfs        2     2     1    open/syncd    /var
hd7                 sysdump    4     4     1    open/syncd    N/A
```

Now the rootvg volume group is back to one PP for one LP, and each LV is in the ‘syncd’ state again. However, the hdisk1 drive still is missing from rootvg:

```
[root@my-system]/> lsvg rootvg -p
rootvg:
PV_NAME      PV STATE    TOTAL PPs   FREE PPs    FREE DISTRIBUTION
hdisk0       active      542         405         108..71..09..108..109
hdisk1       missing     542         542         109..108..108..108..109
```

Now we can remove hdisk1 from the rootvg volume group, leaving only hdisk0:

```
[root@my-system]/> reducevg -d rootvg hdisk1
[root@my-system]/> lsvg rootvg -l
rootvg:
PV_NAME        PV STATE    TOTAL PPs   FREE PPs    FREE DISTRIBUTION
hdisk0         active      542         405         108..71..09..108..109
```

Then the hdisk1 disk can logically be removed from the system:

```
[root@my-system]/> rmdev -dl hdisk1
hdisk1  deleted
```

At this point a replacement disk can be installed into the system. In order to recognize the new disk, we will run **cfgmgr** to walk the bus and discover it:

```
[root@my-system]/> cfgmgr
[root@my-system]/> lspv
hdisk0         000b5a2d9dc260b7    rootvg
hdisk1         000b5a2dcae1a54e    None    (new disk)
hdisk2         000b5a2dcae04016    datavg
hdisk3         000b5a2dcae0b2f1    datavg
```

Now the new disk can be added into rootvg:

```
[root@my-system]/> extendvg -f rootvg hdisk1
[root@my-system]/> lspv
hdisk0         000b5a2d9dc260b7    rootvg
hdisk1         000b5a2dcae1a54e    rootvg
hdisk2         000b5a2dcae04016    datavg
hdisk3         000b5a2dcae0b2f1    datavg

[root@my-system]/> lsvg rootvg -p
rootvg:
PV_NAME        PV STATE    TOTAL PPs   FREE PPs  FREE DISTRIBUTION
hdisk0         active      542         405       108..71..09..108..109
hdisk1         active      542         542       109..108..108..108..109
```

Now the rootvg volume group can be mirrored again, using the new disk:

```
[root@my-system]/> mirrorvg rootvg
Ø516-1126 mirrorvg: rootvg successfully mirrored, user should perform
        bosboot of system to initialize boot records.  Then, user must
        modify bootlist to include:  hdiskØ hdisk1.

[root@my-system]/> bosboot -ad /dev/hdisk1
bosboot: Boot image is 82Ø5 512 byte blocks.

[root@my-system]/> bootlist -m normal hdiskØ hdisk1
[root@my-system]/> bootlist -m normal -o (display the bootlist)
hdiskØ
hdisk1

[root@my-system]/> lsvg rootvg -p
rootvg:
PV_NAME           PV STATE    TOTAL PPs   FREE PPs FREE DISTRIBUTION
hdiskØ            active      542         4Ø5      1Ø8..71..Ø9..1Ø8..1Ø9
hdisk1            active      542         4Ø9      1Ø8..75..Ø9..1Ø8..1Ø9
```

There are once again two PPs for each LP, as shown below:

```
[root@my-system]/> lsvg rootvg -l
rootvg:
LV NAME            TYPE       LPs   PPs   PVs  LV STATE      MOUNT POINT
hd5                boot       1     2     2    closed/syncd  N/A
hd8                jfslog     1     2     2    open/syncd    N/A
hd6                paging     32    64    2    open/syncd    N/A
paging00           paging     32    64    2    open/syncd    N/A
hd4                jfs        4     8     2    open/syncd    /
backupdirlv        jfs        1     2     2    open/syncd    /backupdir
hd1                jfs        1     2     2    open/syncd    /home
hd3                jfs        2     4     2    open/syncd    /tmp
hd2                jfs        57    114   2    open/syncd    /usr
hd9var             jfs        2     4     2    open/syncd    /var
hd7                sysdump    4     4     1    open/syncd    N/A
```

The sooner problems like this, infrequent as they are, are detected, the better. In a future article, we will cover modifications to the ODM that will alert the System Admin via e-mail about serious failures such as the loss of a disk, so that quick action can be taken to restore system integrity.

Michael G Stanton
Supervisor Midrange Systems
Mercedes-Benz (USA)

AIX news

BMC has announced PATROL for Unix, which ensures performance and availability for a wide variety of Unix platforms, including AIX.

For further information contact:
BMC Software, 2101 City West Blvd, Houston, TX 77042-2827, USA.
Tel: (512) 343 1961.
URL: www.bmc.com/patrol/.

* * *

IBM has announced WebSphere Voice Server for AIX Version 2.0. New in this version are a more natural-sounding synthesized speech with a new text-to-speech engine, support for Intel Dialogic telephony platform, and enhanced language support.

There's also consolidated packaging, which includes support for most telephony platforms (Intel Dialogic, Cisco, and WebSphere Voice Response for AIX, with DirectTalk Technology) in a single product.

To assist in application development, the new version comes with WebSphere Voice Toolkit, WebSphere Voice Server SDK, WebSphere Application Server, Advanced Developer Edition V4.0, and WebSphere Studio 4.0 Entry Edition.

Separately, the company has announced WebSphere Voice Response for AIX, Version 2 Release 3. WebSphere Voice Response was previously known as DirectTalk.

For further information contact your local IBM representative.
URL: www-4.ibm.com/software/speech/enterprise/ep_11.html.

* * *

IBM has announced DCE V3.2 for AIX, which offers integrated security services for the client/server environment.

New features include the ability to migrate security information stored in the DCE Security Registry to an LDAP directory, support of DCE GSSAPI delegation with the Kerberos V5 mechanism by implementing support of a DCE initiator to a Kerberos acceptor with delegation, and the use of Entrust V5 with DCE certificate-based authentication.

The new release supports the use of the AIX VisualAge C++ V5.0 compiler with DCE applications.

For further information contact your local IBM representative.
Web address: http://www.ibm.com/software/network/dce.

* * *

IBM has announced availability of the latest version of the Globus Toolkit for IBM eServer systems running AIX. The software enables application developers to create computer grids and grid-based applications.

For further information contact your local IBM representative.
URL: http://www.ibm.com/servers/aix.