27

AIX

January 1998

update

In this issue

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 550955
From USA: 01144 1635 33823
E-mail: HarryLewis@compuserve.com

North American office

Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75067
USA
Telephone: 940 455 7050

Australian office

Xephon/RSM
GPO Box 6258
Halifax Street
Adelaide, SA 5000
Australia
Telephone: 08 223 1391

Contributions

If you have anything original to say about AIX, or any interesting experience to recount, why not spend an hour or two putting it on paper? The article need not be very long – two or three paragraphs could be sufficient. Not only will you actively be helping the free exchange of information, which benefits all AIX users, but you will also gain professional recognition for your expertise and that of your colleagues, as well as being paid a publication fee – Xephon pays at the rate of £170 ($250) per 1000 words for original material published in AIX Update. To find out more about contributing an article, without any obligation, please write to any of the addresses above.

Editor

Harold Lewis

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.

Subscriptions and back-issues

A year's subscription to *AIX Update*, comprising twelve monthly issues, costs £170.00 in the UK; $255.00 in the USA and Canada; £176.00 in Europe; £182.00 in Australasia and Japan; and £180.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the November 1995 issue, are available separately to subscribers for £15.00 ($22.50) each including postage.

AIX Update on-line

Code from *AIX Update* is available from Xephon's Web page at www.xephon.com (you'll need the user-id shown on your address label to access it).

Printed in England.

Disk mirroring, step-by-step

INTRODUCTION

By supporting various schemes for backing up data and the use of RAID storage, AIX provides you with several ways of securing data and ensuring its easy recoverability. AIX's facilities for data security go beyond those offered by other versions of Unix, mostly as a result of AIX's Logical Volume Manager (LVM), which is a powerful tool for controlling data storage at the physical level.

This article describes the process of implementing mirroring using the LVM. However, I'll start with a brief look at the LVM, starting with a short description of terms and acronyms that relate to it.

- *VG*
  A Volume Group is a collection of one or more physical disks, which are aggregated to form 'a storage bundle'.
- *PV*
  A Physical Volume is a disk. Therefore, one or more PVs form a VG. A maximum of 32 PVs is allowed per VG.
- *LV*
  A Logical Volume is the unit of storage that contains filesystems. It is also possible to define 'raw LVs', if your application (for instance, a database) supports it. LVs consist of Logical Partitions, and they can be extended while the system is on-line by adding Physical Partitions to them. At the moment there is no way of reducing the size of an LV (you can 'work around' this problem, but there's still no way to reduce the size of an LV that's anywhere near as easy as it is to extend it). Therefore, it's good practice to define LVs to be as lean as possible, given the constraints of performance, and to increase their size only when performance begins to suffer. A maximum of 256 LVs is allowed per VG.
- *LP*
  A Logical Partition is the unit used to build LVs. At least one LP is needed per Logical Volume. The size of an LP is determined by that of a PP.

- *PP*
  A Physical Partition is the actual physical unit of storage that a Logical Partition maps to. While every PP in an LV maps to one LP, an LP can map to up to three PPs, depending on the mirroring set-up. The size of a PP has to be defined when the VG is created and cannot be changed later. The standard size of a PP is 4 MB for disks larger than 300 MB. The range of sizes available starts at 1 MB, doubling with each step (2, 4, 8, ...) up to 256 MB. As there is a limit to the number of PPs allowed per Physical Volume (the maximum is 1016 per disk), care should be taken when deciding the size of PPs when the VG is set up. A PP size of 4 MB, for example, limits the usable size of disks to 1016 x 4 MB (about 3.97 GB), which is smaller than the average 4.5 GB disk. To use all the storage available on a disk, it is necessary to define the size of PPs for the Volume Group to be at least 8 MB (if you intend to use disks larger than 1016 x 8 MB (about 7.94 GB), such as 9 GB drives, the PP size should be set to 16 MB).

With these definitions in mind we can describe the concepts of mirroring using the LVM.

AIX LVM CAPABILITIES

Mirroring

Mirroring with the LVM is based on Logical Volumes. Every LP can be mapped to up to three PPs; data stored in the LP is written to the PP(s) onto which it is mapped. The filesystem doesn't know about mirroring – it sees only one set of data in the LV. Applications working with 'raw LVs' are also unaware of what the LVM is doing with the data.

Striping

While mirroring is used to store multiple copies of data in case of disk failure, striping is used to speed up data transfer. Striping works by writing data to a number of volumes to which there are separate physical data paths. This allows data to be written to and read from the volumes concurrently, thereby increasing speed of transfer. The LVM is used to implement this feature using striped LVs.

Mirroring

Essentially the mirroring done by the LVM is the same as that defined by the RAID level 1 specification, with the exception that RAID levels are defined on the basis of how they manage data on physical disks, while the LVM works with LVs. Obviously, by writing data to both the primary disk and the mirror volume, twice as much space is consumed as with no mirroring (assuming only one mirror copy is kept – if two copies are kept, three times as much space is required). However, as the cost of disk megabytes continues to fall, this is not such an onerous burden, and is well worth thinking about because of the benefits it offers in terms of fast data recovery when a disk crashes.

Striping

Striping is equivalent to RAID level 0. You implement striping by defining *striped sets*. An identical numbers of LPs have to be defined on each disk that comprises the striped set. This means, for instance, that, if you've got 20 spare LPs on one disk, you can't define them as part of a striped set if you haven't got at least 20 spare LPs on each disk that you wish to include in the striped set. Writing to a striped LV results in concurrent writes of pre-defined chunks of data. For example, if a stripe size of 16 KB is defined, then data written to the LV is transferred in 16 KB chunks. This way all disks in the set write synchronously to their stripes, speeding up disk writes. The same applies to reading stripes. Striping is often used in environments where streamed audio and video needs a defined data transfer rate to work seamlessly.

There is no added data security in striping – if a disk in a striped set fails, data on the striped set can be lost.

STEPS FOR MIRRORING

The following steps were tested on a uniprocessor PowerPC-based RS/6000 system running AIX 4.1.5 with all necessary fixes applied. If you use other versions of AIX, then some of the procedures may differ slightly at some points.

To demonstrate mirroring, an additional disk is connected to the system. As it makes no sense to mirror an LV on the same disk, you'll need to do this if there isn't already another disk available.

There are two ways to connect disk drives to an RS/6000. The first one (also the first one available) is via the widely-used and well-known SCSI interface. SCSI interfaces are very sensitive, so you should not connect disks while either the computer or the disk drive is powered on – while it's possible to change SCSI disks while the system is up and running, this is *not* recommended. SCSI uses a terminated bus, and this means that reflected electrical signals can damage attached devices, especially controllers.

The other, more recent way to connect disks to an RS/6000 is using SSA (Serial Storage Architecture – see *AIX Update* Issue 11 page 35 and Issue 13 page 27). This uses a serial connection to all attached devices. You can connect devices using a duplex loop, which results in a degree of fault tolerance as a fault in one part of the loop results in service being 'failed over' to the other part. Additionally SSA devices can be added or removed while the system is up and running without the risks associated with SCSI.

If SCSI is used, disks are 'daisy chained' to the SCSI port on the chassis. Check that all cables are correctly connected and that a terminator has been connected to the SCSI outlet on the last device in the chain. You also need to ensure that the SCSI ID used by each device is unique and is different from the one assigned to the controller. Now boot the system. After logging in as root, issue the following command:

```
lsdev -Cc disk
```

This should produce a response similar to the one below:

```
root.system@beryllium
/home/root # lsdev -Cc disk

hdisk0 Available 04-B0-00-0,0 1.0 GB SCSI Disk Drive
hdisk1 Available 04-B0-00-1,0 Other SCSI Disk Drive
```

This verifies that the disk has been correctly identified by the OS. However, using the command **lspv** (see code below) shows that the disk is not ready for use yet.

```
root.system@beryllium
/home/root # lspv

hdisk0          005d88157c96b883    rootvg
hdisk1          005d881577712281    None
```

The output above shows that the newly-added disk *hdisk1* doesn't belong to a volume group.

The next step is to assign *hdisk1* to a volume group – in this instance *rootvg*. This is done using the command **extendvg**:

```
root.system@beryllium
/home/root # extendvg rootvg hdisk1
```

Using the command **lspv** again now shows that *hdisk1* belongs to *rootvg* volume group:

```
root.system@beryllium
/home/root # lspv

hdisk0          005d80d48bc3524d    rootvg
hdisk1          005d80d4e9c4b34f    rootvg
```

Example 1: Mirroring an LV

Now the system is ready to mirror an LV from *hdisk0* to *hdisk1*. Mirroring is set up using the **mklvcopy** command:

```
root.system@beryllium
/home/root # mklvcopy hd2 2 hdisk1
```

The command above tells **mklvcopy** to add a copy of LV *hd2* (the */usr* part of the filesystem) and place it on *hdisk1*. The output of **lslv** now shows that LV *hd2* has two copies.

```
root.system@beryllium
/home/root # lslv hd2

LOGICAL VOLUME: hd2                 VOLUME GROUP: rootvg
LV IDENTIFIER:  005d80d4e97c66a1.5 PERMISSION:   read/write
VG STATE:       active/complete     LV STATE:     opened/stale
TYPE:           jfs                 WRITE VERIFY: off
MAX LPs:        512                 PP SIZE:      4 megabyte(s)
COPIES:         2                   SCHED POLICY: parallel
LPs:            135                 PPs:          270
STALE PPs:      135                 BB POLICY:    relocatable
INTER-POLICY:   minimum             RELOCATABLE:  yes
INTRA-POLICY:   center              UPPER BOUND:  32
```

```
MOUNT POINT:    /usr                LABEL:        /usr
MIRROR WRITE CONSISTENCY: on
EACH LP COPY ON A SEPARATE PV ?: yes
```

The output also shows that there are 135 ‘stale PPs’, and that the ‘LV state’ is ‘opened/stale’, where the stale PPs belong to the newly created copy. This is because synchronization of the mirror copies doesn’t occur until one of the following takes place:

- The VG is varied on (for example, during a system boot)
- The **syncvg** command is issued
- The command **mklvcopy -k** is used as in the example below.

```
root.system@beryllium
/home/root # mklvcopy -k hd2 2 hdisk1
```

The command **lslv -l** lists the allocation of PPs on used disks:

```
root.system@beryllium
/home/root # lslv -l hd2

hd2:/usr
PV          COPIES        IN BAND    DISTRIBUTION
hdisk0      135:000:000   32%        000:000:044:051:040
hdisk1      135:000:000   76%        000:000:103:032:000
```

There is now a mirror copy of the LV, and data is written to both *hdisk0* and *hdisk1* simultaneously.

Example 2: Mirroring a data VG

Essentially the steps used to mirror a VG are the same as those used to mirror a single LV. As is the case with mirroring an LV, it’s best to place mirror sets on a different disk. This holds true even if there is more than one disk in the VG. As there is a limit to the number of disks you can include in a VG (currently you can have maximum of 32) bear this in mind when you plan a mirror VG.

To mirror a VG, first mirror all LVs to be included in the VG:

```
root.system@beryllium
/home/root # mklvcopy -k LV_name 2 PV_name
```

Here *LV_name* is the name of the LV to mirror and *PV_name* is the name of the disk on which it is to be mirrored (eg *hdisk2*).

If there are JFSs in the LVs to be mirrored, then the Log-LV file is mirrored also. Log-LVs are created automatically when the first JFS is placed on an LV in the VG, and they have names like ‘loglv00’ in data VGs and ‘hd8’ in *rootvg*.

After completing these procedures, information about the VG and its associated LVs should be up-to-date in the ODM. However, make sure this is the case using the command:

```
root.system@beryllium
/home/root # syncvg -v VG_name
```

This should show that all information about the mirrored LVs is synchronized with that in the ODM.

Example 3: Mirroring rootvg

Mirroring *rootvg* is somewhat different from mirroring other VGs. This is because there are some special LVs in *rootvg*, such as *boot-LV hd5*, *sysdump*, and *log-LV hd8*, and so special care has to be taken when mirroring the whole VG to ensure continuous operation.

There are a few prerequisites to mirroring *rootvg* successfully:

- The disk has to be ‘bootable’. This can be checked by using the command:

  ```
  root.system@beryllium
  /home/root # bootinfo -B hdisk1

  1
  ```

  The value ‘1’ indicates that the disk is bootable. Non-IBM disks are not guaranteed to be bootable, so bear this in mind when considering mirroring *rootvg*.

- AIX’s standard LVs (*hd1*, *hd2*, *hd3*, *hd4*, *hd5*, *hd6*, *hd8*, and *hd9var*) must be on one disk, and their mirror must be on one disk also.

- *hd6* must exist and has to be an active paging device.

- A ‘parallel mirroring policy’ and ‘mirror strictness’ must be used on all LVs in *rootvg* (both are used by default).

- You've got to install PTF IX58121 before you can mirror *rootvg*. Check that this fix is installed using the command:

```
root.system@beryllium
/home/root # instfix -i -k IX58121

All filesets for IX58121 were found.
```

Another important task is to change the QUORUM for *rootvg*, as a VG cannot be varied on if the QUORUM is not fulfilled (in other words, if fewer than 51% of the PVs in that VG are available). This is essential if there are only two disks in the *rootvg*. To disable QUORUM for *rootvg*, use the following command:

```
root.system@beryllium
/home/root # chvg -Qn rootvg
```

Mirroring LVs in *rootvg* is handled in the same way as in Example 1, using the command **mklvcopy**. Once this is done for all LVs, including 'special' LVs (other than *sysdump* – see later), the other LVs can then be handled. The steps to handle special LVs are as follows:

First, information about the newly-mirrored LVs in *rootvg* has to be synchronized with the ODM using the following command:

```
root.system@beryllium
/home/root # syncvg -v rootvg
```

1 Process *hd5*, the boot device, as follows:

If *hd5* consists of more than one LP (as would be the case if the 'PP size' for *rootvg* is less than 4 MB), then check that the mirrored PPs for *hd5* are contiguous using the command **lslv**:

```
root.system@beryllium
/home/root # lslv -m hd5

LP    PP1  PV1               PP2  PV2               PP3  PV3
0001  0001 hdisk0            0006 hdisk1
```

If the PPs are contiguous, then no additional processing is required. If they aren't, then delete the copy of *hd5* using the **rmlvcopy** command and make a fresh copy using the command **mklvcopy -m** (the '-m' flag ensures that the PPs are contiguous). Next use the **syncvg** command once more to synchronize the mirror.

If proper configuration of the mirror is shown, the next step is to initialize both copies of *hd5* as boot devices:

```
root.system@beryllium
/home/root # bosboot -a

bosboot: Boot image is 5554 512 byte blocks.
```

The boot image is now stored on both copies of *hd5*. To activate both copies for booting, use the following command:

```
root.system@beryllium
/home/root # bootlist -m normal hdisk0 hdisk1
```

This ensures that, in case of a disk failure, the mirror copy of *hd5* can be used as the boot device. (In a few rare instances this might fail.)

2 Process *hd6*, the paging device, as follows:

Paging space can be mirrored in the same way as other LVs. However, if there are other paging devices defined, then they have to be mirrored as well. Note that this applies to paging spaces in both *rootvg* and other VGs.

3 Process *hd7*, the *sysdump* device, as follows:

The discussion that follows doesn't just apply to installations where system dumps are collected – it applies to all installations where *rootvg* is mirrored, as the paging space, *hd6*, is defined in AIX 4 as the primary *sysdump* device. If paging space is mirrored, then *sysdump* will fail as it's not capable of handling mirrored devices.

For these reasons a *sysdump* device should always be specified, even if you have no intention of doing anything with dumps. For historical reasons (dating back to AIX 3) the *sysdump* device is called *hd7*.

The *sysdump* device is used to store the 'famous last words' of the operating system. In AIX 4, the *sysdump* device is (by default) *hd6*, the paging space. If a *sysdump* occurs, then it is placed on *hd6*. When booting after a crash, the system tries to copy the dump to */var/adm/ras* as *vmcore0*. Depending on the size the

applications that were running when the *sysdump* occurred, the dump may be too big for that location. In such instances the administrator has to copy the dump to an external device, such as tape. This can be time-consuming and is not feasible at sites where the system is configured to reboot automatically after a crash.

Use the steps below to create a *sysdump* device with the correct settings.

First, it makes sense to estimate the size of a *sysdump* before defining the device on which to locate it, as you'll cause additional problems by defining a *sysdump* device that's too small to hold the dump. The following command should be issued with maximum load on the machine in question:

```
root.system@beryllium
/home/root # sysdumpdev -e

0453-041 Estimated dump size in bytes: 25323520
```

The above data is taken from a system running CDE along with all standard daemons, and supporting one active user. As always, err on the side of caution.

First, create an LV of the appropriate size on the original disk:

```
root.system@beryllium
/home/root # mklv -y hd7 rootvg 7 hdisk0

hd7
```

Once this is done, the LV has to be made the new permanent primary *sysdump* device:

```
root.system@beryllium
/home/root # sysdumpdev -P -p /dev/hd7

primary              /dev/hd7
secondary            /dev/sysdumpnull
copy directory       /var/adm/ras
forced copy flag     TRUE
always allow dump    FALSE
```

Next the settings have to be changed so that a system dump may always be produced, and this change has to be verified:

```
root.system@beryllium
/home/root # sysdumpdev -p /dev/hd7 -K

root.system@beryllium
/home/root # sysdumpdev -l
primary              /dev/hd7
secondary            /dev/sysdumpnull
copy directory       /var/adm/ras
forced copy flag     TRUE
always allow dump    TRUE
```

Once these procedures are correctly carried out, a *sysdump* device is defined.

Now, *sysdump* is a special device as it cannot be mirrored. If you consult the manual pages, they'll explain that dumps are not displayed if the dump device is mirrored, and that all subsequent dumps to that device will fail. This requires a different approach to mirroring to ensure *sysdump*'s correct operation, involving the definition of a secondary dump device. A new LV, with a new name but the same size as the primary dump device, has to be created on the disk where the mirrored LVs are located:

```
root.system@beryllium
/home/root # mklv -y hd7a rootvg 7 hdisk1

hd7a
```

Next this LV is configured as the secondary dump device:

```
root.system@beryllium
/home/root # sysdumpdev -P -s /dev/hd7a

primary              /dev/hd7
secondary            /dev/hd7a
copy directory       /var/adm/ras
forced copy flag     TRUE
always allow dump    TRUE
```

The secondary dump device will now operate, ensuring that a dump is available should one of the two disks fails.

WHAT TO DO IF A DISK CRASHES

It is a fact of life that disks fail. After all, they are the most commonly used computer devices that have intricate moving parts, and so their

lifetime is limited despite the latest improvements in disk technology. The procedures to recover crashed mirrored disks are discussed below.

Case 1 – recovering a mirrored LV

If only one mirrored LV is resident on the failed disk, then the procedure is relatively simple. Just reduce the number of copies for that LV, as in the following example:

```
root.system@beryllium
/home/root # rmlvcopy hd3 1 hdisk1
```

The output of **lslv** now shows that there is only one copy left:

```
root.system@beryllium
/home/root # lslv hd3

LOGICAL VOLUME: hd3                VOLUME GROUP: rootvg
LV IDENTIFIER:  005d80d4e97c66a1.7 PERMISSION:   read/write
VG STATE:       active/complete    LV STATE:     opened/syncd
TYPE:           jfs                WRITE VERIFY: off
MAX LPs:        128                PP SIZE:      4 megabyte(s)
COPIES:         1                  SCHED POLICY: parallel
LPs:            3                  PPs:          3
STALE PPs:      0                  BB POLICY:    relocatable
INTER-POLICY:   minimum            RELOCATABLE:  yes
INTRA-POLICY:   center             UPPER BOUND:  32
MOUNT POINT:    /tmp               LABEL:        /tmp
MIRROR WRITE CONSISTENCY: on
EACH LP COPY ON A SEPARATE PV ?: yes
```

Next log off, shut down, power off the system, replace the defective disk, and re-implement mirroring.

Case 2 – recovering a mirrored VG

The procedure to recover a mirrored VG is basically the same as that outlined in Case 1: reduce the number of mirrored copies for every LV resident on the failed disk to the appropriate number. In addition, the defective disk has to be removed from the VG and information about it has to be removed from the ODM. The steps below describe this procedure in detail.

First, the number of copies of each LV has to be reset to the number of copies that are actually available:

```
root.system@beryllium
/home/root # rmlvcopy LV_name 1
```

The following commands remove all information about the defective disk from the ODM and VGDA. Next, the disk has to be removed from the VG:

```
root.system@beryllium
/home/root # reducevg datavg hdisk2
```

In case either the *sysdump* device or its mirror is located on the failed disk, *sysdump* has to be set temporarily to */dev/sysdumpnull*. It can be set back after the recovery is complete.

Next, the OS's definition of the disk is deleted:

```
root.system@beryllium
/home/root # rmdev -l hdisk2 -d
```

The next step is to shut down the system, power it off, and physically replace the disk. After rebooting the system, the new disk has to be integrated in the VG and the number of copies for each LV has to be set to the previous value.

If a paging device has been set up in this VG, then some additional steps are required. First, the paging space has to be de-activated:

```
root.system@beryllium
/home/root # chps -an LV_name
```

LV_name is the name of the paging space. Next the machine is rebooted. Only then can the paging space be removed:

```
root.system@beryllium
/home/root # rmps LV_name
```

Now the failed disk can be replaced, and the paging space in the VG can be mirrored and re-activated. To activate the mirrors of the LVs, perform the procedures described in *Mirroring a data VG*.

Case 3 – recovering a mirrored rootvg

As was the case when mirroring *rootvg*, recovering *rootvg* from a failed disk requires special attention. This is especially true of the 'special' LVs *hd5* and *hd7*. The following steps have to be performed before the defective disk is changed:

As before, the number of copies of mirrored LVs has to be reduced using the command **rmlvcopy**:

```
root.system@beryllium
/home/root # rmlvcopy LV_name 1
```

Next the *sysdump* device is temporarily moved (it will be set back to its original value when the new disk is integrated).

```
root.system@beryllium
/home/root # sysdumpdev -P -p /dev/sysdumpnull

primary              /dev/sysdumpnull
secondary            /dev/hd6
copy directory       /var/adm/ras
forced copy flag     TRUE
always allow dump    TRUE
```

The *bootlist* for normal mode has to be modified to reflect the new situation:

```
root.system@beryllium
/home/root # bootlist -m normal hdisk0
```

Next the failed disk is removed from the ODM and VGDA:

```
root.system@beryllium
/home/root # reducevg rootvg hdisk1

root.system@beryllium
/home/root # rmdev -l hdisk1 -d
```

It is now safe to shut down the system and replace the failed disk. After rebooting with the replacement disk, the steps in *Mirroring rootvg* should be performed.

DOCUMENTATION

General information on disk mirroring can be found in AIX's InfoExplorer, while the manual pages contain more up-to-date information regarding the necessary commands. Always read the manual pages carefully, as they often contain important information that is easily overlooked.

Two books to take a look at are: *AIX Version 4 System Management Guide: Configuring a Storage System for Maximum Availability* and

AIX Version 4 Problem Solving Guide and Reference Appendix C (*Recovering Volume Groups*).

There are also some good on-line sources of information. *www.rs6000.ibm.com* is a good starting point, while *www.rs6000.ibm.com/resource/aix_resource/Pubs/redbooks* gives more detailed information. 'Redbooks' can be browsed on-line and provide tested procedures for many different tasks, including disk mirroring and volume group design. Another source of information is *www.rs6000.ibm.com/resource/links.html*. This contains links to the newsgroup *comp.unix.aix* as well as to some archives that can be searched by topic. There is also a link to Team RS6000, a user group with a close relationship with IBM's AIX labs.

Peter Wuestefeld
res nova Unternehmensberatung (Germany)

Understanding performance inhibitors

As a systems administrator, you may occasionally be approached by users concerned that their AIX systems have suddenly started to run slow. They'll complain that jobs are taking much longer than usual to complete.

Another common complaint is that users have processes that never seems to run to completion, or that there are windows on their workstations that refuse to close.

If it's your job to assist users with such problems, then you may find yourself faced with the difficulty of knowing where to start looking for the cause of the slowdown or failure.

The aim of this article is to illustrate some good approaches to troubleshooting system performance problems. In particular, this article addresses three common types of performance problem:

1 Processes that are consuming excessive resources.

2 Filesystems that are running out of space.

3 Remotely-mounted filesystems that have terminated.

PROCESSES THAT CONSUME EXCESSIVE RESOURCES

The first type of problem occurs when a process is consuming a large enough share of the system's resources to cause performance degradation for other processes.

This category could include:

- Programs that are in an endless loop.
- Programs that are processing large amounts of data.
- Programs that cause excessive disk activity when processing large volumes of input/output.

Two typical examples of the last category above are compilers working on large source files and programs that perform 'extracts' or check-in/checkout operations on a large number of files.

Another problem that can result in similar symptoms are programs that have finished processing, and have left 'remnants' of themselves. These remnants can be leftover processes or windows that won't close.

ARE PROCESSES AFFECTING PERFORMANCE?

A very useful command that can be used to investigate processes is the **ps** command. This command shows the current status of processes. A useful variant of this command is **ps -ef**. The **-ef** flags tell the command to generate a full listing of all processes except kernel processes (see *Managing processes under AIX*, *AIX Update* Issue 10, page 36). Figure 1 opposite shows what the various columns in the output of the command **ps -ef** mean.

If you want to filter the list of processes so as to include only those that you suspect are causing the performance problem, use the command:

```
ps -ef | grep xxxxx
```

where *xxxxx* is the name of the process, user, or program that you suspect is failing.

Below is a shell script (**/usr/bin/p**) that you can use to allow you and your users quickly to determine which processes are running. I suggest that you locate the tool in */usr/bin* and change the execute status using the command **chmod 755 /usr/bin/p**, so that any user who has */usr/bin* in their PATH can run the tool.

/USR/BIN/P

```
if test $# -ne 0
   then ps -ef | grep $1
   else ps -ef | grep 'whoami'
fi
```

Whenever you want information on a specific process, user, or program, simply enter **p** followed by the specific designation. If **p** is entered without a parameter, then all processes belonging to user that issued the command are displayed.

The first line of the script checks to see if a parameter is supplied on input. If so, the second line issues **ps -ef** and then 'greps' on this parameter. If no parameter is specified, the third line issues the

UID	The user ID of the process's owner
PID	The process ID
PPID	The parent process ID
C	The CPU utilization of process
STIME	The process's start time
TTY	The process's controlling workstation
TIME	The total execution time of the process
CMD	The process's full command name and parameters

*Figure 1: The information provided by the command **ps -ef***

command **ps -ef** and 'greps' on the user's login ID (which is obtained by the command **whoami**).

The following examples show how to identify processes that may be causing a system resource problem. The section below entitled *How do I stop a process?* tells you how to terminate a process that you suspect is failing.

Example 1

Suppose you know that a program called *myprog1* is looping and utilizing system resources. If you enter **p myprog1**, you might get output similar to that below:

```
jjones 24580 34040  2  Jun 10 pts/8  0:00 myprog1
root   32350 23470  0  Jun 10 pts/0  0:00 grep myprog1
```

In this example, the process ID is *24580*, so this is the process that needs to be terminated. Note that the tool also shows the process number of the **grep** operation.

Example 2

Now that you've used the **p** tool to locate information about a specific program, here's how you can use it to collect information about the processes belonging to a specific user. In this example, a user (*jjones*) reports that his sessions seem to be running slowly, but he doesn't know which process is causing the problem. From your system console, you could enter **p jjones** to find out which processes are owned by *jjones*. The **p** tool effectively issues the command **ps -ef | grep jjones**, and this may typically result in the following output:

```
jjones 36440 34060  2  Jun 10 pts/8  0:00 myprog2
jjones 34060 32240  0  Jun 10 pts/8  0:00 -ksh
root   32350 23470  0  Jun 10 pts/0  0:00 grep jjones
```

In this example, if you have reason to suspect that program **myprog2** was causing the slowdown, then process *36440* would need to be terminated.

Example 3

Suppose you want to investigate the parent process of the *-ksh* shell

in Example 2 above. As the parent process ID (PPID) of *-ksh* is *32240*, the command to use is **p 32240**, which produces the output:

```
jjones 34060 32240  0  Jun 10 pts/8  0:00 -ksh
root   32240  5440  0  Jun 10 pts/0  0:00 telnetd
root   42910  6878  0  Jun 10 pts/0  0:00 grep 32240
```

If you suspect that it's the *-ksh* shell that's responsible for the performance problem, then this shows that this process's parent is a telnet session. In this instance it may be appropriate to terminate the session in order to free resources.

Example 4

Suppose you experience a sudden and unexpected slow-down in your system's performance. Typically you don't know which user is running the process that's affecting performance. The first thing to do is issue the **who** command to find out which users are currently logged on to the AIX system.

So you issue the **who** command, and get the following output:

```
root    pts/0   Jun 10 15:20
rsmith  pts/2   Jun 10 08:35   (mach1.xyz.com)
jjones  pts/8   Jun 12 12:20
```

The output of the **who** command shows the user's login name, the line name, the date and time when the user logged on, and the host from which the user logged on.

You could then use the **p** tool to search for processes owned only by each of the users that are currently logged on to the system. If you know, for instance, that user *rsmith* often runs large compiles, then you could check whether *rsmith* is logged on using **who**, and then use the command **p rsmith** to see if *rsmith* is running a compile. You wouldn't want to terminate *rsmith*'s process in the middle of his compile, but at least you know what caused the problem and can then determine an appropriate solution.

HOW DO I STOP A PROCESS?

The command used to terminate an AIX process is **kill**. Once you've determined the PID of the process that's affecting system performance,

use the **kill** command to remove it from the system. This requires you be either a root user or the initiator of the process.

For example, to kill process number *24040*, enter:

```
kill 24040
```

If your diagnostics were successful, killing the specified process should free the system resources and/or close the windows that failed to close.

It's important to note that randomly killing processes is not always an ideal way of dealing with system resource problems. In particular, some processes spawn other processes that themselves consume system resources, making them a better target for a kill (you also have to consider what processes are doing – you don't want to kill off vital system functions!). With experience, you'll learn which types of process you can terminate to free system resources.

FILESYSTEMS THAT REACH MAXIMUM SIZE

The next type of resource problem occurs when certain filesystems reach their size limit. For instance, it's common for the */home* filesystem periodically to reach its maximum size as this is the filesystem in which your users store most of their data. Another filesystem that's prone to growing relentlessly is */usr*. If you tend to load many programs into */usr*, then you'll find it running out of space quickly. This problem usually surfaces when someone complains that attempts to load new code failed.

It's also not unusual for the root (*/*) filesystem to become overloaded. Some administrators create directories directly off the root filesystem to run temporary programs and store data. However, they may not be aware that the root filesystem is often set to be very small. If the root filesystem becomes full, certain root and system processes may fail, and – what's more important – the cause of the failure may not be readily apparent.

ARE FILESYSTEMS AFFECTING PERFORMANCE?

There are several AIX commands that can be used to determine

whether filesystems are running out of free space. Some are used to determine whether there are filesystem in your storage that are full and causing performance problems. Once you've identified that such problems exist, other commands are used to determine exactly where the problems are.

Diagnosing full filesystems

The first command you should use to determine whether you have one or more full filesystems that are causing performance problems is **df**. This command displays space information for all currently mounted filesystems. Figure 2 below shows a sample output of the **df** command.

In this example, the */home* filesystem is set to contain up to 204,800 512-byte blocks (approximately 100 megabytes). Of those, about 13K blocks are free, which represents about 94% utilization. The */usr* filesystem is about 95% utilized, and the root filesystem is about 82% full. If these, or any other mounted filesystem, were to reach or exceed 99% utilization, you could experience complete filesystem failure.

Once you've found that you have one or more filesystems that are full, you have two courses of action open to you. If the problem is a result of a process or user generating too much output, then you should try to identify the user or process that's caused the problem. Alternatively, if the problem has arisen from a steady growth in the number of files, and is not the result of one user or process, then you'll have to find files to delete or move so as to free up some space.

```
Filesystem  512-blocks    Free  %Used  Iused %Iused  Mounted on
/dev/hd4         57344   10872    82%   1391     9%  /
/dev/hd2       1204224   61104    95%  20470    14%  /usr
/dev/hd9var      24576    3656    86%    311     8%  /var
/dev/hd3         57344   33840    41%     68     1%  /tmp
/dev/hd1        204800   13016    94%   5761    22%  /home
```

*Figure 2: Sample output of the command **df***

The examples below illustrate what happens when you run out of free space in filesystems. Typically such problems arise in the */home* filesystem, though they can arise in any other filesystem. The purpose of the examples is to show which commands can be used to locate and analyse filesystems that show symptoms of being full, and when each is appropriate. The section titled *How do I manage a filesystem's size?* shows you how to handle such problems.

Example 1

A system administrator calls to say that a root process has failed. After using the command **df**, you find that the root directory is 100% full. To find out which files or directories were written most recently, you could issue the command **ls -lrt** from the root directory. This shows a long list, sorted by write time, with the most recent entries appearing last. (You could enter the command without the **r** flag to show the most recent files first, but data will scroll off the screen if there are too many files.)

The result shows the most recent activity last, which may help you identify which file(s) caused the root filesystem to fill to its limit. The most recently active entries may include files or directories.

Suppose that the most recent entry is a directory called *admin*. Entering **ls -lrt /admin** shows that the most recent file in that directory is *nohup.out*. This file is created as a result of a **nohup** command – a command that allows work to be performed even if the user logs off the system. The **nohup** command appends output to the file *nohup.out*. What the administrator that issued the command failed to take into account is that *nohup.out* can continue to grow as long as the **nohup** command is still running. In this case, the file caused the root filesystem to run out of free space.

If you delete a *nohup.out* file, don't forget to stop the **nohup** command as well, or it may continue running (and consuming resources).

Example 2

Many users have probably had programs fail, resulting in core dumps. What they may not know is that the core dumps (which are often very large) remain in their directories until they are removed.

You can use **find** to locate all the core dumps in the */home* filesystem, and then ask your users to delete them if they are no longer needed. Do so with the following command:

```
find /home -name core -print
```

The output of this command is a list of directories (with paths) that contain core dumps.

Example 3

Many programs are stored as archive files so that they are easier to transfer and consume less space than they otherwise would. Such files typically have names that end with either *.tar*, if they're uncompressed **tar** files, or *.tar.z*, if they're compressed **tar** files. After you've loaded compressed **tar** files on to your system, you uncompress them and extract the accompanying directories and files into their appropriate filesystems. Some users, however, may forget to remove the archive file itself after the extract. This leaves excess redundant data on the system, which has little use if the files have already been extracted. The **find** command may be useful in locating such files, and helping to identify potential space to free.

Use the following command to find **tar** files in the */home* filesystem,:

```
find /home -name '*.tar*' -print
```

This command displays the paths to directories that contain either uncompressed *.tar* files or compressed *.tar.Z* files. It's up to you and your users to determine which of these files can be safely removed. You could also use this command on the */usr* directory to find **tar** files used to install programs in that filesystem.

Example 4

Suppose you're trying to find ways of freeing up space in the */home* filesystem. A good tip is to remember that developers often keep endless copies of every version of all their source files as they develop code. Typically, they'll create several subdirectories, each containing a complete version of their product, which might consist of hundreds or even thousands of files.

A useful command to help find instances of this practice is **du**. This

```
du -sk /home/*


5194    /home/jjones
43093   /home/rsmith
382     /home/bbaker
19092   /home/guest
```

*Figure 3: Using **du** on the /home filesystem*

command summarizes disk usage. The variant that you may find most useful is **du -sk**. This displays the total number of blocks used in a directory tree in 1 KB increments.

Figure 3 above shows an example of the command and its output used to find which user is utilizing the most disk space. You can see from the illustration that the *rsmith* subdirectory contains the most data in the */home* filesystem.

Figure 4 opposite shows the **du** command being used to discover which subdirectory of */home/rsmith* is using the most space. Note that the command **du -sk** shows disk space for both files and directories. If there are many individual files in the specified directory, you may want to redirect the output of the command to a file, and then edit that file to see the results.

The illustration reveals that *dev1* is the largest directory in */home/rsmith.* Further investigation may reveal that *rsmith* has four or five subdirectories under *dev1*, each of which contains copies of several hundred development files. The **du** command is usually the best one to use to track down this kind of potential saving.

Example 5

A user comes to you claiming to be unable to free space in his home directory. According to him, he's removed all files in his development directories, which he confirms using the command **ls** (the command

```
du -sk /home/rsmith/*


5       /home/rsmith/test.file
127     /home/rsmith/data
42638   /home/rsmith/dev1
29      /home/rsmith/note
```

*Figure 4: Using **du** on files belonging to 'rsmith'*

shows that there are no files there). However, when he enters **du -sk**, specifying the directory, the resulting report shows several kilobytes of data are present. To understand what may be happening, it is important to know how AIX handles certain file names.

Some programs have a safeguard to stop them overwriting existing files. One way they do this is by renaming the existing file using the same name, but with a character added or changed. For example, if the program writes a file called *abc22000.txt*, and a file of this name already exists in the target directory (probably generated by the same program in a previous run), the program may preface the old file with a 'dot' or 'period' (.), thus renaming the old file to *.abc22000.txt* before writing the new one.

One reason for prefacing the file with a period is that AIX treats the period as a 'special character', which causes the file to be ignored by some commands. This includes the **ls** command, which displays a list of files in a directory, and ignores files with names that begin with a period. Use the **ls -a** command to display files that begin with a period (as well as other files normally displayed by **ls** – see *AIX Update* Issue 13, page 32, for more information on the **ls** command).

HOW DO I MANAGE A FILESYSTEM'S SIZE?

Once you've found that a filesystem has reached its limit, there are several things that can be done.

- Every computer user should know that the first and most obvious step is to erase files that are no longer needed.
- If you've got large files or subdirectory structures that contain numerous files that are rarely used, then you can use **tar**'s 'create' method, followed by the **compress** command, to reduce the total space needed to store the files. This procedure copies all the specified files and directories into a single compressed file. Don't forget to delete the original data once the archive file is created! Not all files are candidates for this form of storage, as you need to use the **uncompress** command along with **tar**'s 'extract' method to access the data again.
- You can move the data to a Distributed Filesystem (DFS), if you have one available. This method off-loads your data to a secure filesystem that can be located on a number of servers across the network.
- You can increase the size of the filesystem using **smit** so that it's large enough to contain your current data. Figure 5 below shows how to navigate **smit** to the screen that allows you to change the size of a Journaled Filesystem such as root, */usr*, and */home*. **smit** uses the **chfs** ('change filesystem') command to increase the size of a filesystem. Beware – you can only increase the size of a filesystem, you can't decrease it, so you can't 'temporarily' give a filesystem more room.

1 System Storage Management (Physical and Logical Storage)
2 File Systems
3 Add|Change|Show|Delete File Systems
4 Journaled File Systems
5 Change|Show Characteristics of a Journaled File System
6 Select from File System Name list

*Figure 5: Navigating **smit** to re-size a filesystem*

REMOTELY-MOUNTED FILESYSTEMS THAT HAVE TERMINATED

The third type of problem occurs when remotely-mounted filesystems terminate, either normally or abnormally. It's not unusual to have key programs and utilities that are located on remote servers mounted on local AIX systems. This enables users to use these functions without the necessity of having the code installed locally.

One drawback to this set-up is that, if the server containing the programs terminates (for instance, as a result of the remote server being rebooted or a network fault) this will affect local users, some of whom may be unaware that they're using services on the remote server.

Are remotely-mounted filesystems affecting performance?

Two commands that can help troubleshoot remotely-mounted filesystems are **df** and **mount**.

As stated previously, the **df** command displays information about mounted filesystems. This applies to both local filesystems, such as */home* and */usr*, and remote filesystems, such as those that might contain your site's commonly-used tools and utilities.

Figure 6 below shows an example of information about remotely-mounted filesystems displayed using **df**. This example shows that the filesystems */public* and */tools* mounted on the local system physically reside on a server called *prodserv*.

```
                    512K                                    Mounted
Filesystem         blocks    Free  %Used  Iused  %Iused      on
prodserv:/public   614400  381936   38%     -       -      /public
prodserv:/tools    819200  497792   40%     -       -      /tools
```

*Figure 6. Listing remote file systems using **df***

The next command to consider is **mount**. When you use this command with no parameters, the following information is displayed about all remotely-mounted filesystems:

```
                    mounted
node     mounted over    vfs date         options

prodserv /public /public nfs Jun 05 11:57 rw,hard,bg,intr,noacl

prodserv /tools  /tools  nfs Jun 05 11:57 rw,hard,bg,intr,noacl
```

Figure 7: Listing remote file systems using ***mount***

- Node
- Object mounted
- Mount point
- Virtual filesystem type
- Time mounted
- Mount options (if any).

Figure 7 above shows an example of information about remotely-mounted filesystems displayed using this command. The example shows the same filesystems as the **df** command, also showing the mount options. This can be useful in troubleshooting performance problems.

The examples below illustrate ways of detecting problems with remotely-mounted filesystems that are affecting performance.

Example 1

Shortly after you notice a severe slowdown in your system's performance, you get calls from users complaining of the same problem. You use **df** to check your local filesystems, and find that one or more remotely-mounted filesystems fail to respond.

At this point two strong possibilities exist:

1 If directories from all remotely-mounted servers display except those from one particular server, then that server is probably down.

2 If none of your remotely-mounted filesystems display, then you've probably got a network or other communications failure.

Example 2

You are aware that a particular server is down. Most of your users are running without complaints of system slowdown, but one user notices that every command seems to take a long time to complete.

From the user's workstation you issue the command **echo $PATH**, which reveals that a directory on the failing server is one of the first entries on his path. A brief examination shows that most of the other users on your system have that directory towards the end of their paths.

When a user executes a command, the command processor searches the user's path for the directory that contains the desired command, ending the search once the command is located. If the command processor attempts to search a directory on a server that is unavailable, this can result in a delay. This example shows how the position of directories in your user's PATH statement can affect performance. So it's a good idea to place remote directories at the end of the path, where appropriate.

Managing remote filesystems better

One way to prevent remotely-mounted servers from affecting local performance is to ensure that the mounts are specified as 'soft' rather than 'hard'. When an attempt is made to mount a filesystem, a soft mount returns an error if the server does not respond, whereas a hard mount results in the system continually re-sending the request until the server responds. Soft mounts can prevent remote outages from causing performance problems in future, as well as improving performance when the volume is mounted. You can use the **mount** command shown in Figure 7 to establish whether remote volumes use hard or soft mounts. Figure 8 overleaf shows you how to navigate **smit** to the screen that allows you to change the type of mount from hard to soft. **smit** uses the **chnfsmnt** command ('change NFS mount') to alter the options of a remotely-mounted filesystem.

If all your remotely-mounted filesystems are hard mounted, then

1 System Storage Management (Physical and Logical Storage)
2 File Systems
3 Add|Change|Show|Delete File Systems
4 Network File System (NFS)
5 Network File System (NFS)
6 Change|Show Attributes of an NFS File System
7 Enter the name of the file system

*Figure 8: Navigating **smit** to change the mount type*

there's not much you can do if remote servers fail. Otherwise, you can use the **umount** command (along with a list of directories mounted from that server) to unlock you from a server that's not responding. If a network outage is preventing remote server access, then use **umount all** to try to unmount all currently mounted filesystems. It can take several minutes for the **umount** command to complete but, when it does, you should find your resources freed. With no remotely mounted filesystems, you can only perform local processing until the network becomes available again. At that time, entering **mount all** causes the system to attempt to mount all filesystems in */etc/filesystems* that have the *mount=true* attribute set.

Also consider investigating your users' PATHs to establish whether their sequence can be changed to improve the resilience of their systems to remote servers that fail to respond. If possible, locate critical directories at the front and those of remote filesystems at the rear. It is not always practical nor possible to have certain directories precede others, so you may have to experiment with directory placement in the PATH statement.

SUMMARY TABLE

The following table summarizes the commands and steps presented in this article.

RUNNING PROCESSES THAT UTILIZE EXCESSIVE SYSTEM RESOURCES	
Determine if running processes are affecting performance.	ps ef grep xxxxx (where xxxxx is the process, user or program you suspect is failing) who
Manage a running process.	kill nnnnn (where nnnnn is the process number to terminate)
FILESYSTEMS THAT HAVE REACHED THEIR MAXIMUM SIZE	
Determine if you have a full filesystem problem.	df
Locate the source of a full filesystem problem.	ls lrt find /home name core print find /home name '*.tar*' print du sk ls al
Manage the size of a filesystem.	Erase files that are no longer needed. Use tar and compress for infrequently used directories and files. Move the data to DFS. Increase the size of the filesystem using SMIT.

REMOTELY MOUNTED FILESYSTEMS THAT HAVE TERMINATED	
Determine if remotely mounted filesystems are affecting performance.	df mount echo $PATH
Manage remotely mounted filesystems.	Use "soft" mounts rather than "hard" mounts. mount umount DIRECTORY umount all mount all Check PATH statements.

David Chakmakian (USA)

User administration

Like many of the tasks that system administrators have to carry out, such as managing priority, availability, and reliability, the amount of work that has to be done to manage user accounts always seems to grow. Tasks such as the ones below seem to take up an increasing amount of administrators' time.

- Checking user accounts, looking for redundant ones. Typically such accounts have not been used for some time and are often associated with projects that have moved to other systems or ones belonging to users that have left the company or changed to another department.

- Looking for user accounts that are locked as a result of too many attempts to login with incorrect passwords. These can be a result of users forgetting their passwords or users trying to gain access to an account to which they have no right (of course, the users concerned may feel – possibly quite rightly – that they do have a right to access the account concerned).

The problem that this article addresses is that of reducing the time it takes for administrators to carry out user management and increasing system security by (among other means) using AIX's in-built system utilities.

THE SOLUTION

There are three common tasks associated with the administration of user accounts that can be substantially reduced using a combination of good practice and automation – they are adding user accounts, removing user accounts, and reviewing user accounts (finding redundant accounts, merging accounts, moving accounts, etc).

CREATING USER ACCOUNTS

It's good practice to use settings that relate to all accounts – these are set in directories that contain system-wide login profiles for all users, such as */etc/profile*. For example, the variable TMOUT, which determines the idle time at the **ksh** prompt before a user is logged off the system, and TIMEOUT, which determines the same parameter for **bsh**, can be set centrally to affect all users that use shells. It's often worth foregoing settings that are individually tailored to groups of users to simplify user administration.

For other applications, use centrally-defined environment scripts. Include them in your default profile, */etc/security/.profile*, which is the prototype that is assigned to a new user account when it's created.

Customize the 'make user' script, */etc/security/mkuser.sys*, to include modifications specific to your site, such as creating user-specific temporary directories, or restricting FTP access to the system using an entry in the file:

```
/etc/ftpusers
```

It's a good idea to reduce the maximum size of files that the user can create to an appropriate limit, such as 10 megabytes, by changing the corresponding statement in */etc/security/limits*.

Next, set the necessary security parameters in */etc/security/user*:

- *pwdwarntime*
  Sets the number of days' warning that AIX gives users before passwords expire.
- *loginretries*
  Sets the number of invalid login attempts before the user account is locked.
- *minlen*
  Defines the minimum length of passwords.
- *maxage*
  Defines the maximum permitted age (in weeks) of passwords.
- *minage*
  Defines the minimum age (in weeks) of passwords.
- *minalpha*
  Defines the minimum number of alphabetic character that a password must contain.
- *minother*
  Defines the minimum number of non-alphabetic characters that a password must contain.
- *mindiff*
  Defines the minimum number of different characters in a new password.
- *maxrepeats*
  Defines the maximum number of times a character can be repeated in a password.

REMOVING USER ACCOUNTS

Below is a simple procedure for removing unwanted user accounts – substitute *username* with the name of the account you want to remove.

The first thing you need to do is create a list of user accounts that are no longer needed and should be removed. This may require you to go through a list of accounts by hand, identifying ones that can be dispensed with. Use the command below to create a list of user accounts and discuss it with the relevant managers. (Under *Reviewing user accounts* you'll also find scripts to identify old and unused accounts automatically.)

```
find / -user username -print
```

To ensure recoverability, make a tape back-up of all important files belonging to each user whose account is to be terminated:

```
lsuser -f username > /home/username/user.info
chown username /home/username/user.info
find / -user username -print | backup -vf/dev/rmt0

find / -user username type -f -exec rm {} \;
find / -user username type -d -exec rmdir {} \;
```

Repeat this procedure for all entries you can find that belong to *username*. Depending on how methodically the user has maintained their directories (you may find they contain files that are owned by other users) this may take some time.

When you have deleted all files and directories belonging to the user you can delete their authentication information using the command:

```
rmuser -p username
```

REVIEWING USER ACCOUNTS

Run the scripts below periodically (either weekly or monthly should suffice). Their output gives you an overview of old, expired, and locked user accounts. This enables you to carry out user management pro-actively.

Setting up the system

To install the two scripts below you need to make the following changes. Change any paths in the scripts (**_check_user.ksh** and **_check_user.awk)** to suit your directory structure and set the variable *username*. Set the permissions. Run the script. Select your default settings for **week_count** and **ship_to**.

_CHECK_USER.KSH

```
#!/usr/bin/ksh
# scriptname : _check_user.ksh
#
# Userreport - all old,expired,locked and su acounts
# paramter 1 : week counter
# paramter 2 : e_mail address for report receipt
#
tempfile=/tmp/system/user.tmp
parmfile=/tmp/system/_check_user.tmp
report_file=/tmp/system/check_user_report.tmp
check_file=/tmp/system/usrck.tmp
exclude_file1=/tmp/system/usrck_exclude_expired.tmp
exclude_file2=/tmp/system/usrck_exclude_locked.tmp
exclude_file3=/tmp/system/usrck_exclude_intruder.tmp
exclude_file4=/tmp/system/usrck_exclude_rlogin.tmp
exclude_file_all=/tmp/system/usrck_exclude_all.tmp
scriptfile=/scripts/_check_user.awk
#
if test $# -lt 2
then
  week_count=8
  ship_to=no
else
  week_count=$1
  ship_to=lastname@x123.perth
fi
#
cat /etc/security/lastlog  \
  | egrep -v '(^^\*)' | egrep '(^^.*:$|time_last_login)' > $tempfile
#
echo ${week_count} > ${parmfile}
#
# To get an actual last-login date from today -
# Please substitute the USERNAME with an acutal userid - who logs in
# daily
#
ixxx=`lsuser -a time_last_login USERNAME`
echo ${ixxx#USERNAME time_last_login=} >>  ${parmfile}
#
# Generate a list of all expired,locked,intrudered
# or only su user
#
usrck -n ALL  > $check_file 2>&1
cat $check_file | egrep '(3001-664)' \
      | awk '{print "^^"$6}' | sort > $exclude_file1
cat $check_file | egrep '(3001-662)' \
      | awk '{print "^^"$3}' | sort > $exclude_file2
cat $check_file | egrep '(3001-661)' \
```

```
      | awk '{print "^^"substr($12,1,length($12)-1)}' \
      | sort  > $exclude_file3
#
lsuser -a rlogin ALL | grep =false  \
     | awk  '{print "^^"$1 }' | sort > $exclude_file4
lsuser -a login ALL | grep =false  \
     | awk  '{print "^^"$1 }' | sort >> $exclude_file4
#
# put alle the exclude ids together
#
cat $exclude_file1 > $exclude_file_all
cat $exclude_file2 >> $exclude_file_all
cat $exclude_file3 >> $exclude_file_all
cat $exclude_file4 >> $exclude_file_all
#
# Generate a single outputfile
#
{
echo "User accounts with " $anz_wo  weeks older last-logins at
`hostname`
echo "Userreport generated : " `date`
echo
awk -f  $scriptfile $tempfile | egrep -v -f $exclude_file_all | sort
echo
echo All expired user accounts  ---------------
echo
cat /etc/passwd | egrep -f $exclude_file1
echo
echo All locked  user accounts  ---------------
echo
cat /etc/passwd | egrep -f $exclude_file2
echo
echo All intruder user accounts ---------------
echo
cat /etc/passwd | egrep -f $exclude_file3
echo
echo All only-su  user accounts ---------------
echo
cat /etc/passwd | egrep -f $exclude_file4
} > $report_file
#
#
if test ${ship_to} = "no"
then
   cat $report_file
else
   cat $report_file | mail -s "User-Check at `hostname` " ${ship_to}
fi
#
# cleaning up the workfiles
```

```
#
rm ${parmfile} ${tempfile}
rm ${report_file} ${check_file}
rm ${exclude_file1} ${exclude_file2} ${exclude_file3} ${exclude_file4}
rm ${exclude_file_all}
#
# eof _user_check.ksh
```

_CHECK_USER.AWK

```
**************************************************************************
#  _user_check.awk check user lastlogins
#
BEGIN { zz = " "
#                    seconds  minutes   hours    days
        one_week  =   60   *   60   *   24   *   7
#
        getline week_count < "/tmp/system/DADC_check_user.tmp"
        getline actual_login < "/tmp/system/DADC_check_user.tmp"
        past_login = actual_login - ( one_week * week_count )
}
#
#
{
#
if ( zz != " " )
  {
#
# situation  jkjljj:
#            jksdfjl:
   if ( (match($0,":") > 0 ) && (match(zz,":") > 0 )   )
      {
# next line : too on the end - then
# user never made a login to the machine
#
        x=system("egrep '^^" zz "' /etc/passwd" )
      }
   else
# Situation  jklsdlfj:
#                  last login
# This user has a last_login line
#
        if ( (match(zz,":") > 0 ) && ( match($0,":") == 0 ) )
      {
      if ( $3   <  past_login )
        {
#
```

```
# This is an old account
            user=substr(zz,1,length(zz)-1)
            x=system("egrep '^^" zz "' /etc/passwd" )
                 }
       else
# Thats okay - do nothing
        innnn= "  "
       }
   }
zz = $0
}
#        eof _user_check.awk
```

SAMPLE OUTPUT

```
User accounts with  8 weeks older last-logins at  x123.perth
Userreport generated :  Wed Jul 2 07:54:26 1997

adsmop:!:315:201:Adsmoperator :/home/adsmop:/usr/bin/ksh
bspe01:!:261:201:Bob Spencer :/home/bspe01:/usr/bin/ksh
orastart:!:310:201:User for orasrv restart:/home/orastart:/usr/bin/ksh
All expired user accounts ---------------

daemon:!:1:1:AIX System_User:/etc:
bin:!:2:2:AIX System_user:/bin:
sys:!:3:3:AIX System_User:/usr/sys:
lpd:!:104:9:AIX System_Line_printer:/:

All locked user accounts ---------------

guest:!:100:100:Guest_user nologin:/home/guest:
All intruder user accounts ---------------

imhotep:!:434:201:Michael Imhotep:/u/imhotep:/bin/ksh
All only-su user accounts ---------------

uucp:!:5:5:AIX System_User:/usr/spool/uucppublic:/usr/lib/uucp/uucico
guest:!:100:100:Guest_user nologin:/home/guest:
nobody:!:4294967294:4294967294:AIX System_User:/:
oracle7:!:352:205:Pgm owner :/oracle7:/bin/ksh
servdir:*:229:1::/home/servdir:/usr/bin/ksh
```

Michael Imhotep (Australia) © Xephon 1998

Converting AIX1 to Excel 97

Last year we published a number of utilities in *AIX Update* for monitoring AIX with PCs (see *Monitoring AIX with PCs*, Issue 23 pages 3–32 and Issue 24 pages 12–38). The PC tools themselves were written using the version of Visual Basic for Applications (VBA) that comes with both Excel 5 and Excel from Office 95 (the two are essentially the same product). At the end of the series we gave some brief pointers on how to adapt the tools for use with the latest version of Excel, which comes with Office 97. This has a version of VBA that's now recognizable as the same product as Visual Basic 5.

Given the increasingly widespread use of Office 97, we now offer more detailed instructions on how to carry out the conversion.

In the Excel workbook for AIX1:

1 Enter *Tools* menu

2 Select *Macro* option

3 Select *Macros*

4 Select *Var* macro (it should be the last one in the list)

5 Click *Edit.*

This takes you to Microsoft's Visual Basic Editor. From the *Edit* menu select *Replace*. Perform the three replace operations that are shown in Figure 1 opposite (note that, in both of the first two operations, there is a space between the two characters being replaced, and also between the two characters being inserted). In each case, click *Replace All*. This should result in 22, 15, and two replacements respectively.

You need to make three other changes to code. The *runsetup* macro needs to insert a new sheet, as VBA no longer stores macros as module sheets. If there is already a sheet called *Tempry* in your workbook before you run this macro, the macro fails. Replace *runsetup* from AIX1 with the version below.

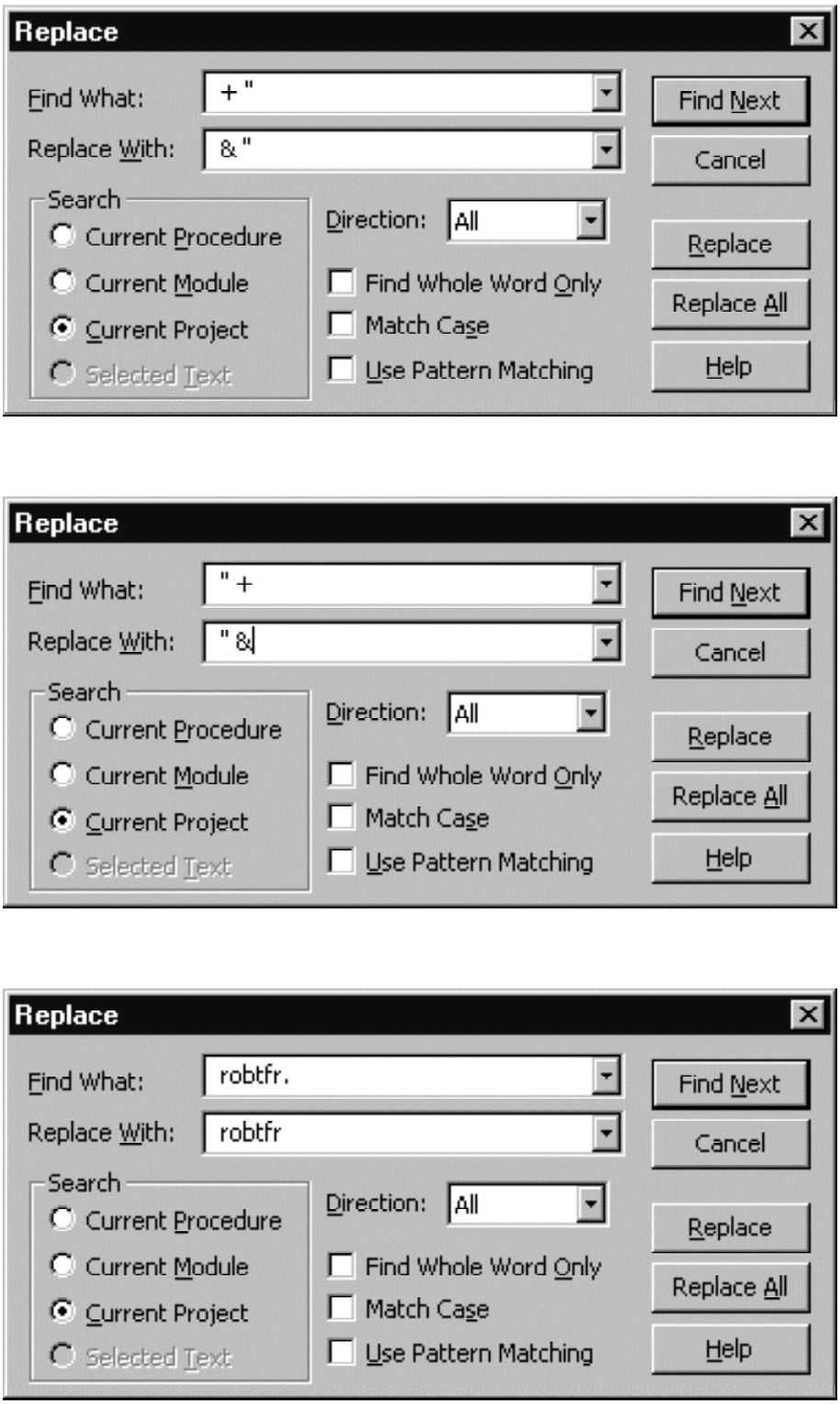

Figure 1: Global changes to be made to AIX1

RUNSETUP

```
Sub runsetup()
auto = 1
Sheets.Add
ActiveSheet.Name = "Tempry"
For Each sh In ActiveWorkbook.Worksheets
                    Application.DisplayAlerts = False
                    If sh.Name <> "Tempry" Then
                    sh.Delete
                    End If
                    Application.DisplayAlerts = True
                Next sh
                For Each sh In ActiveWorkbook.DialogSheets
                    Application.DisplayAlerts = False
                    sh.Delete
                    Application.DisplayAlerts = True
                Next sh
                Sheets.Add
                ActiveSheet.Name = "sheet9"
                Sheets.Add
                ActiveSheet.Name = "sheet8"
                Sheets.Add
                ActiveSheet.Name = "sheet7"
                Sheets.Add
                ActiveSheet.Name = "sheet6"
                Sheets.Add
                ActiveSheet.Name = "sheet5"
                Sheets.Add
                ActiveSheet.Name = "sheet4"
                Sheets.Add
                ActiveSheet.Name = "sheet3"
                Sheets.Add
                ActiveSheet.Name = "sheet2"
                Sheets.Add
                ActiveSheet.Name = "sheet1"
                Sheets.Add
                ActiveSheet.Name = "robtfr"
                dial
setup
Application.DisplayAlerts = False
Worksheets("Tempry").Delete
End Sub
```

The *drawchrt* macro also needs to be changed, as the font property ‘autoscale’ may be selected on your system. In *drawchrt*’s code (the subroutine is a bit over half way down the code listing) locate the following group of lines:

```
    Sheets("Sheet8").Select
```

```
ActiveSheet.ChartObjects.Add(1.5, 10.25, 650.25, 250.5).Select
If filet <> "CPU Stats" Then
ActiveChart.Type = xlLine
chtitle = chtitle & " Filesystem : " & filet & " :"
```

Now insert the statement setting the AutoScaleFont property of the ChartArea object to 'False', as shown below.

```
Sheets("Sheet8").Select
ActiveSheet.ChartObjects.Add(1.5, 10.25, 650.25, 250.5).Select
ActiveChart.ChartArea.AutoScaleFont = False
If filet <> "CPU Stats" Then
ActiveChart.Type = xlLine
chtitle = chtitle & " Filesystem : " & filet & " :"
```

Similarly, the autoscale property also affects the *statuschrt* macro, so it too needs to be changed. Locate the code for *statuschrt* (it's the last subroutine in the listing), and find the following lines in the code:

```
ActiveSheet.ChartObjects.Add(2, 11, 550, 310).Select
ActiveChart.Type = xlColumn
For x = 1 To d
```

Now insert the statement to set the AutoScaleFont property of the ChartArea object to 'False', as before.

```
ActiveSheet.ChartObjects.Add(2, 11, 550, 310).Select
ActiveChart.ChartArea.AutoScaleFont = False
ActiveChart.Type = xlColumn
For x = 1 To d
```

Depending on your exact set-up, the program may need a bit of minor tweaking, but should nonetheless work.

Robert Russell (UK)

We're going to publish more PC utilities for monitoring and analysing AIX system performance written by Robert Russell in the coming months. If you've written any utilities for managing AIX performance yourself, whether they run on AIX or PCs, we'd like to hear about them. Articles and listing can be sent to the editor at harrylewis@compuserve.com. Don't forget we pay for published material at $250 (£170) per 1000 words and $140 (£90) per 100 lines of code.

Timestamp to date

Many Unix files and commands use a date format called the 'epoch', which specifies the date in terms of the number of seconds since the 'epoch' – 00:00:00 GMT, January 1, 1970 (see *Y2K – the 'Year 2000 problem'*, AIX Update Issue 24, page 45). Two examples are */etc/security/passwd* and */etc/security/lastlog*. If you need to process files such as these, you'll find this utility useful. Note that this utility is a pre-requisite of **login_check**, a utility published in last month's issue (see *Checking login attributes*, *AIX Update* Issue 26, page 49), and should, therefore, have been included in last month's issue – our apologies to readers for this oversight.

The C utility listed below, **epoch2date**, takes the epoch timestamp as an argument and returns the date in human-readable format.

Example:

```
./epoch2date 844408405

Fri Oct  4 07:53:25 1996
```

EPOCH2DATE.C

```
/*
 * FUNCTION:
 * Convert epoch timestamp to human-readable date
 *
 * SYNTAX:
 * epoch2date timestamp
 *
 * ARGUMENTS:
 * epoch time (the number of seconds since 1970)
 *
 * RETURNS:
 * The date in the format: www mmm dd hh:mm:ss yyyy
 *     www  weekday
 *     mmm  month
 *     dd   day
 *     hh   hour
 *     mm   minute
 *     ss   second
 *     yyyy year
 *
```

```
 * DESCRIPTION:
 * This program converts an epoch timestamp (the number of seconds
 * since the start of 1970) and return the date in a human-readable
 * format.
 *
 * Compile with:
 * cc -o epoch2date epoch2date.c
 */

#include <time.h>

#include        <stdio.h>
#include        <stdlib.h>
#include        <sys/time.h>
#include        <sys/types.h>

main(
int argc,
char *argv[]
)
{

time_t  epoch;

                  if(argc==2)
                  {
                         epoch=atol(argv[1]);
                          printf(asctime(localtime(&epoch)));
                  } else {
                          printf("Usage: %s epochdigits\n",argv[0]);
                  }
}
```

System Programmer (Switzerland)

The RS/6000 model S70 server

The 64-bit RS/6000 model S70 is the newest and most powerful member of IBM's RS/6000 PCI server family. It is the first RS/6000 server to be made in IBM's AS/400 plant in Rochester, NY, and it represents a departure from the 'traditional' RS/6000, as reflected in

its packaging – the cabinet is tall and black, rather than the usual RS/6000 cream colour, and bears a striking resemblance to an AS/400 model 650. It provides approximately twice the OLTP performance of an RS/6000 model R50. Put another way, it has 60 times the OLTP performance of a model 250, the base RS/6000 against which the performance of other RS/6000s is measured. (IBM prefers to state the performance of RS/6000s relative to that of the base model, in the same way as the company states the performance of mainframes.)

The S70 also departs from the traditional 'all-in-one' RS/6000 as it has a tower for the Central Electronic Complex (CEC – IBM's term for the CPU and memory) and one to four I/O racks for drawers containing PCI expansion slots, disks, tape drives, and other media. These racks (model 7014-S00) are to be standard on other rack-mounted RS/6000s in future.

The system runs AIX 4.3, the new 64-bit implementation of AIX. As with all new RS/6000s, a free copy of AIX, including a two-user licence, is bundled with the system. (For an overview of AIX 4.3, see *AIX version 4.3 previewed*, *AIX Update* Issue 22, page 8.) The system also includes a Bonus Pack, comprising Lotus Domino (Notes Server), Netscape Fast Track Server, Netscape Navigator, Ultimedia Services, Adobe Acrobat, DCE Client Station, and Network Station Manager.

CEC TOWER

The CEC can contain between four and twelve SMP processors. The base machine uses four PowerPC RS64 processors running at 125 MHz. These processors were developed by the AS/400 group in Rochester. The system has 64 KB instruction and data caches. Each processor also has 4 MB of Level 2 cache.

The base machine comes with 512 MB of RAM, and the system can accommodate a maximum of 16 GB memory in 20 slots. A multi-path switch handles communication between the CPUs and memory (it provides a total bandwidth of 5,300 MB/s).

I/O TOWER

The I/O tower holds 19-inch rack-mounted components. Each tower

accommodates a maximum of four IBM rack drawers. The top drawer of the first rack contains PCI expansion slots. There are four 64-bit slots and seven 32-bit slots available to the user in each I/O drawer. Using the maximum of four I/O drawers, this increases to 56 slots. The serial ports, keyboard, mouse, and service processor connectors are in this drawer also. The disk drives are hot-pluggable. Each drawer can hold up to 12 hot-pluggable disks, yielding a raw capacity of 54 GB per drawer. (The 9.1 GB drives each takes up two bays.) The system comes with a 20 x CD-ROM drive, service processor (see below), and a diskette drive. It can support a maximum of 14 terabytes of external SSA storage.

RESILIENCY

The S70 is marketed as an 'enterprise server', with much attention paid to 'RAS' (Reliability, Availability, and Serviceability). The general engineering and build quality also seems much higher than some previous PCI offerings.

For example, the processors and memory are packaged in smart aluminium 'books'. Each has holes for guidance pins, making it easy to slot a card in without damaging contact pins. The aluminium casing also protects the card from electrostatic and physical damage.

IBM has made numerous other enhancements to the design of this server to improve its availability – as well as the features mentioned above, the S70 has multiple cooling fans, which can be replaced in-flight. The power subsystem provides resilience against failures in either the bulk or regulated power subsystems by having redundant units, which are also designed so that a repair or replacement can be carried out while the machine is powered up.

The new PCI SSA card, which is also used in other servers, provides support for HACMP. Other components include ones for 'predictive failure analysis' (early warning of component failure), and improved memory error correction.

Real-time machine surveillance is provided by the *service processor* and the *service director*. The service processor continually monitors AIX for problems and takes appropriate action – for example, by

restarting the system (usually a last resort) or reporting problems to administrators by 'phone. The service director allows IBM and its business partners to check warranty details, analyse and log problems, and notify customers of problems.

The console can be mirrored, and the NVRAM (where the firmware, etc is stored) can be updated remotely.

UPGRADING FROM OTHER MODELS

IBM provides an upgrade path for users of some previous systems, such as some J-series processors (J30, J40, J50) and R-series processors (R30, R40, R50). As these systems bear little resemblance to the S70, there are very few bits that remain in the target system after the upgrade. The DIMM memory from these machines can be used, but there are some restrictions that mean this is probably not worth the effort (the performance of the S70 can be reduced, and there are limitations as to the placement of the boards). If future memory upgrades are contemplated, then the transferred memory must be replaced anyway.

THE FUTURE

IBM's predictions are usually on the conservative side. The company has indicated that it intends to use the Northstar chip in RS/6000s in 1998, which would improve performance to an estimated 36,000 tpm. There are several other plans for improving RS/6000 performance over the next three years. These include supporting up to 64 GB of RAM and using the 630++ 332 MHz processor. (The machine is said to be 'Year 2000 compliant' – this comes as quite a relief at this late stage in the century!)

CONCLUSION

Although this level of performance does not come cheap, the S70 makes a good option for consolidating multiple, smaller RS/6000s onto one unit. It has enough expansion capacity in terms of CPU, memory, and disk to satisfy most users, and its build quality and RAS enhancements give it credibility for use on mission critical applications.

The reviews of the machine have generally been very positive and it does appear that IBM has got it right – at last – with the S70.

Contributing to *AIX Update*

AIX Update is primarily written by practising AIX specialists in user organizations – not journalists, or consultants, or marketing people. In our view, the information and advice provided by such people – people like you and your colleagues – are far more valuable to their fellow professionals than the alternatives available from other sources.

We don't expect you to write an original article from scratch – just send us listings or specifications of any relevant programs, utilities, scripts, user modifications, or other code that might be of use to other installations, with a short explanation of why it was developed and what it does. And most IS departments produce a great many internal technical reports and other documents, many of which can easily be adapted for publication. Xephon's editorial staff will transform even the most informal listing or document into a polished article fit for publication. So you don't have to spend any time on reformatting or rewriting your contribution – just send it as it is and we'll do the rest.

Contributors aren't just helping their fellow professionals – they also receive a significant material reward themselves. We pay good rates for the articles we publish: $250 (£170) per 1000 words if we get copyright, and $140 (£90) per 100 lines of code.

For a copy of *Notes for contributors*, please contact the editor, Harold Lewis, at any of the addresses shown on page 2, or e-mail him at *HarryLewis@compuserve.com*.

AIX news

IBM has announced MQSeries version 5.0. Improvements over the previous version include database-message syncpoint coordination, support for 100MB messages and files, DCE connection security, and Novell SPX support. It runs on AIX, HP-UX, Sun Solaris, OS/2 Warp, and Windows NT, and is out now priced at US$3,100 per licence.

IBM has also announced ADSM version 3.0. It includes better back-up and restore performance, and various fault-tolerant features. There will also be a new server-to-server feature to share data on multiple ADSM servers, and a new Web-based administrative interface for operating the software from an intranet. IBM says it will integrate it into its Network Storage Manager. ADSM v3 is available on AIX, MVS, and NT.

For further details contact your local IBM agent.

* * *

Platinum has announced its ProVision suite of integrated system management tools. The suite has a modular design, allowing tools to be deployed when needed. Basically it includes new versions of existing Platinum products, including AutoSys for job management; WireTap, ServerVision, and DBVision for performance management; Apriori for Problem Resolution; etc.

It runs on AIX (and some other versions of Unix) and NT. No details on prices.

For further information contact:
Platinum Technology, 1815 S Meyers Road, Oakbrook Terrace, IL 60181, USA
Tel: +1 630 620 5000
Fax: +1 630 691 0718
Web: www.platinum.com

Platinum Technology, Platinum House, North Second Street, Central Milton Keynes MK9 1BZ, UK
Tel: +44 1908 248400
Fax: +44 1908 248444

* * *

BGS Systems has launched BEST/1 for Distributed Systems, its performance management product for Unix and NT systems. The software collects, analyses, models, and presents performance data, looking at both applications and system resources. Among its features are facilities for capacity planning for new applications and for managing service levels. There's a data collection agent that picks up data for tracking the performance of resources, applications, and users. It's shipping now (contact BGS for prices).

For further information contact:
BGS Systems, One First Avenue, Waltham, MA 02254, USA
Tel: +1 617 891 0000
Fax: +1 617 890 0000
Web: www.bgs.com

BGS Systems, Bridge Gate, 55-57 High Street, Redhill, Surrey RH1 1RX, UK
Tel: +44 1737 778400
Fax: +44 1737 779060