104

AIX update

June 2004

In this issue

AIX Update

Published by
Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor
Trevor Eddolls
E-mail: trevore@xephon.com

Publisher
Nicole Thomas
E-mail: nicole@xephon.com

Subscriptions and back-issues

A year's subscription to *AIX Update*, comprising twelve monthly issues, costs $275.00 in the USA and Canada; £180.00 in the UK; £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 2000 issue, are available separately to subscribers for $24.00 (£16.00) 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.

Disclaimer

Contributions

When Xephon is given copyright, articles published in *AIX Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. 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.

Renaming PTF files

Fixes to IBM software can be downloaded from one of many IBM Internet sites, although IBM has tried to centralize them under a 'fix central' site. The fixes themselves can be retrieved in a number of different formats. Single fixes tend to be in the usual fileset.name.version.revision.modification format. This is quite handy when talking to IBM about problems and matching up with the output from the **lslpp** command. However, if you download a maintenance package, IBM bundles all the patches nicely under their PTF name, eg Unnnnnn.bff.

The name itself does not bother AIX as the .toc file in the fix file directory maps the filename to the fileset name. But if you download a maintenance package and then try to work out which filesets need extra fixes downloaded, that can take quite some time.

Below is a small Perl script that reads the contents of the .toc file from a directory containing fixes in the Unnnnnn.bff format and changes the names of the fixes to the fileset.name.version format.

```
#!/usr/bin/perl -w
#
# Perl script to alter PTF file names from Uxxxxxx.bff format
# to fileset.v.r.m.x format
#
$|=1;
use strict;
use File::Copy;
my $file;
my $new_name;
my $line;
my @split_line;
my $nextline;
my $dirname="/AIX/ptf/directory";
system("/usr/sbin/inutoc \"$dirname\"");
open(DT,"$dirname/.toc")|| die "Cannot open dot toc\n";
while ($line=<DT>) {
  next if ( $line !~ /^U\d\d\d\d\d\d\.bff/ );
  $file=(split(/ /,$line))[0];
  $nextline=<DT>;
```

```
    @split_line=split(/ /,$nextline);
    $split_line[1] =~ s/\b0+//g;
    $split_line[1] =~ s/\.\.\./.0.0./g;
    $split_line[1] =~ s/\.\./.0./g;
    $split_line[1] .= "0" if ( $split_line[1] =~ /\.$/ );
    $new_name=join(".",$split_line[0],$split_line[1],"bff");
    print "File $dirname/$file -> $dirname/$new_name\n";
    move("$dirname/$file","$dirname/$new_name");
}
close DT;
system("/usr/sbin/inutoc \"$dirname\"");
chmod(0644,"$dirname/.toc");
```

Phil Pollard
Unix and Tivoli Administrator (UK)

AIX error logging in your mailbox on your PC

Editor's note: this article follows on from the article AIX mail in the mailbox on your PC*, published in* AIX Update*, issue 103, May 2004.*

INTRODUCTION

Once you have configured sendmail you can post mail from AIX to your PC. If there is mail generated it will be forwarded to you – but that is all!

If entries are added to the errorlog, you will miss them – no mail is generated. You can choose to log in to every AIX server once a day (or week) and watch the records in the errorlog by using the command **errpt**. With one or two servers this is a viable option, but if you have lots of servers it will just generate extra work, and, if there have been important errors, you will be too late (as usual!).

To make your work more interesting (not just looking at errors that are 'garbage') and to be better informed, you can implement mail_errorlog. It will check every quarter of an hour whether there are any important errors and mail the serious errors to you.

This script does not clean up the errorlog, you have to do it by using the command **errclear**.

You have to do something to get rid of the errors, but what? It depends. You could:

- Call service to resolve disk errors.
- Expand filesystems or clean up filesystems if they are 100% full.
- Check whether the RSF call was made and clean the callout log.
- Etc.

But usually, if a problem occurs, you can do something about it in time. And when you get a message from end users saying, “It doesn’t work any more”, you can say, “I know, I’m busy fixing it right now”.

STEPS TO IMPLEMENT THE SCRIPT

Step one: download the script from Xephon’s Web site (www.xephon.com/aix), edit the script, and copy it to your AIX machine.

There are three variables you have to set:

- date_file – file name and location

  This variable is used to check whether all errors have to be sent or just the new ones.

  Example: */home/error_report.date.*

- save_file – file name and location

  This variable is used to save all the errors that have been sent each day.

Example: */home/error_report.mail.*

- mail_address – mail address of a user or group of users who have to be informed of the errors.

  Example: AIX_administrators@yourcompany.com.

You can change the errors, delete them if you do not use the entries, or add other entries if there are more errors that you would like to have mailed to you.

You can choose to get the full format of the message by uncommenting the lines between the lines:

```
# Uncomment below this line to get full messages
```

and:

```
# Uncomment above this line to get full messages
```

This will change the output from:

```
IDENTIFIER TIMESTAMP  T C RESOURCE_NAME  DESCRIPTION
613E5F38   0212131104 P H LVDD           I/O ERROR DETECTED BY LVM
```

to:

```
------------------------------------------------------------------------
LABEL:          LVM_IO_FAIL
IDENTIFIER:     613E5F38

Date/Time:       Thu Feb 12 13:11:30
Sequence Number: 477824
Machine Id:      00509B5A4C00
Node Id:         afts5
Class:           H
Type:            PERM
Resource Name:   LVDD

Resource Class:  NONE
Resource Type:   NONE
Location:        NONE

Description
I/O ERROR DETECTED BY LVM

Probable Causes
POWER, DRIVE, ADAPTER, OR CABLE FAILURE

        Recommended Actions
```

```
          RUN DIAGNOSTICS AGAINST THE FAILING DEVICE

Detail Data
PHYSICAL VOLUME DEVICE MAJOR/MINOR
ØØ16 ØØ25
ERROR CODE AS DEFINED IN sys/errno.h
           5
BLOCK NUMBER
    39356664
LOGICAL VOLUME DEVICE MAJOR/MINOR
ØØ2F ØØØ5
PHYSICAL BUFFER TRANSACTION TIME
           Ø
SENSE DATA
ØØØ2 5888 ØØ5Ø 9B5A Ø4E3 1F1F ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ Ø843 1341 F271
ØØØØ ØØØØ
ØØØØ ØØØØ
----------------------------------------------------------------------
```

I prefer to show the errors on one line. Some errors will generate lots of lines so you cannot see what is happening. If you want to see the full error message, you can login on that specific machine to look at it.

Step two: make an entry in the crontab to execute the script on a regular basis.

I decided to execute the script every 15 minutes. This time can be configured as you like.

Example:

```
Ø,15,3Ø,45 * * * * /usr/local/bin/mail_errorlog
```

MAIL_ERRORLOG SCRIPT

```
# Name              : mail_errorlog
# Last modification : 19-Ø2-Ø4 teun post - creation script
# Description       : list serious errors and mail them
#---------------------------------------------------------------------

# Site dependent variables
date_file=/home/error_report.date
save_file=/home/error_report.mail
mail_address=AIX_adminstrators@yourcompany.com

# Initialize script variables
DATE='date +%d%m'
hostname='hostname'
```

```
# Check whether $save_file exists
if [ ! -f $save_file ]
then # First time this script is executed: create file where mailed
     # messages are saved
     >$save_file
     fi

DATE_SEND='fgrep "$DATE" $date_file 2>/dev/null'
if [ ! "$DATE_SEND" ]
then # every day a full error list
     >$date_file
     fi

# Report the serious errors:
# Sysplanar things: eg ENVIRONMENTAL PROBLEM
errpt -N sysplan*     >/tmp/err_mail.tmp

# Harddisk errors: eg UNDETERMINED ERROR or DISK OPERATION ERROR
errpt -N hdisk*      >>/tmp/err_mail.tmp

# Memory errors
errpt -N mem*        >>/tmp/err_mail.tmp

# SCSI errors:
errpt -N scsi*       >>/tmp/err_mail.tmp

# File system messages eg UNABLE TO ALLOCATE SPACE IN FILE SYSTEM
errpt -N SYSPFS      >>/tmp/err_mail.tmp

# Errors on the Ethernet cards
errpt -N ent*        >>/tmp/err_mail.tmp

# Navisphere errors
errpt -N Navisphere  >>/tmp/err_mail.tmp

# Input Output (I/O) messages
errpt -N SYSIOS  >>/tmp/err_mail.tmp

# Only atf messages when failures or changes occurred
errpt -N atf*  | grep -E "FAILURE|REGISTERED CHANGE STATE" >>/tmp/
err_mail.tmp

# RSF messages, like RSF DIAGNOSTIC MESSAGE:
#                                         make sure a callout was made
errpt -N RSF         >>/tmp/err_mail.tmp

# Logical Volume Manager errors such like I/O ERROR DETECTED BY LVM
errpt -N LVDD        >>/tmp/err_mail.tmp

# Create the report
```

```
if [ -s /tmp/err_mail.tmp ]
then # There is something to report: format the report
     # Set flag : next run not the whole stuff,
     #                                       just the additional lines
     echo $DATE >$date_file

     # Sort it and get rid of duplicate headings
     cat /tmp/err_mail.tmp | sort -u -o /tmp/err_mail.tmp2

     # Get the heading of the report
     HEAD='fgrep IDENTIFIER /tmp/err_mail.tmp2'

     #  Remove the heading: it is not in proper sequence
     cat /tmp/err_mail.tmp2 | grep -v IDENTIFIER >/tmp/err_mail.tmp

     if [ "$DATE_SEND" ]
     then # Already data send: send only the new lines added to
          # the errorlog
          diff /tmp/err_mail.tmp $save_file | grep "^<" | cut -c3- >/
tmp/err_mail.tmp2
     else # send the whole file
          mv /tmp/err_mail.tmp /tmp/err_mail.tmp2
          fi

     # If there are any new error lines: mail it.
     if [ -s /tmp/err_mail.tmp2 ]
     then # rewrite the history,
          #                     add the additional lines we are sending
          cat /tmp/err_mail.tmp2 $save_file | sort -u >/tmp/
err_mail.tmp3
          mv /tmp/err_mail.tmp3 $save_file

# Uncomment below this line to get full messages
#          >/tmp/err_mail.tmp4
#          cat /tmp/err_mail.tmp2 | while read identifier rest
#               do
#               errpt -a -j $identifier  >>/tmp/err_mail.tmp4
#               done
#          mv /tmp/err_mail.tmp4 /tmp/err_mail.tmp2
#               done
#          mv /tmp/err_mail.tmp4 /tmp/err_mail.tmp2
# Uncomment above this line to get full messages

          # A header in a file gives some explanation
          echo "$HEAD" >/tmp/err_mail.tmp
          cat /tmp/err_mail.tmp2 >>/tmp/err_mail.tmp

          # mail the report
          mail -s "Error report from site $hostname" ${mail_address} </
tmp/err_mail.tmp
```

```
          fi
     fi

# Remove the work files
rm /tmp/err_mail.tmp* 2>/dev/null

# Exit script
```

EXAMPLE SENDMAIL OUTPUT

Heading in Outlook:

```
   Mail_user@unixa.domain_name  Error report  from site unixa      fri
5-3-04 11:03
```

Mail:

```
IDENTIFIER TIMESTAMP  T C RESOURCE_NAME  DESCRIPTION
0EC00096   0212131004 P U SYSPFS         STORAGE SUBSYSTEM FAILURE
1ED0A744   0212131004 P U SYSPFS         FILE SYSTEM LOGGING SUSPENDED
5DFED6F1   0211040004 I O SYSPFS         UNABLE TO ALLOCATE SPACE IN
FILE SYSTEM
5DFED6F1   0227110904 I O SYSPFS         UNABLE TO ALLOCATE SPACE IN
FILE SYSTEM
5F88611E   0212131004 P H atf1-hdisk15   ATF FAILOVER FAILURE
613E5F38   0212131104 P H LVDD           I/O ERROR DETECTED BY LVM
BFE4C025   0212133404 P H sysplanar0     UNDETERMINED ERROR
CD0FF9B1   0208090804 P H Navisphere     STORAGE SUBSYSTEM FAILURE
CD546B25   0212131104 I O SYSPFS         FILE SYSTEM RECOVERY REQUIRED
D2A1B43E   0212131104 P U SYSPFS         FILE SYSTEM CORRUPTION
```

Teun Post
AIX Administrator/Oracle DBA
Schuitema NV (The Netherlands)

Checking an AIX FAStT configuration

When configuring FAStT LUNs attached to machines running AIX, do not assume that just because a machine recognizes the external disks resident on the FAStT as a number of hdisks your configuration is a workable solution. It may be far from it, and elementary checks should be made to see that your configuration is indeed correct.

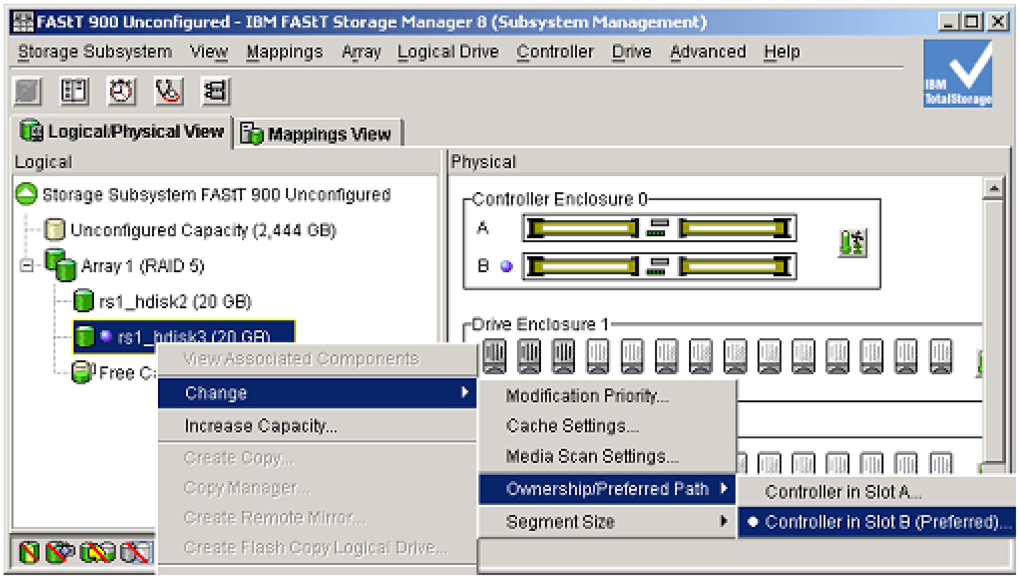

Figure 1: LogicalPhysical view

Disks on the FAStT are configured to a preferred controller, through which all I/O is normally sent (without load balancing), but a faulty configuration may not allow a failover to the redundant controller in the event of the failure of the preferred controller or the SAN switch to which it is attached. In either of these situations you will lose all access to your filesystems and logical volumes.

You may also find that volume groups that have been configured on a host which is incorrectly configured for its FAStT resources will not be automatically varied on after a reboot, depending on which controller is active and accessible and which disks are associated with a controller.

The preferred controller can be shown from the Subsystem Management Window after starting the Storage Manager software, by right-clicking on the array disk in the **Logical/ Physical View** – see Figure 1.

If, for example, hdisk2 has been configured with preferred controller A, then it will be associated with the AIX controller device dac0. Similarly, any disk with preferred controller B will be associated with the device dac1.

Both controllers are normally associated with a single router device, dar0, which provides a route through to each of the controllers. Should one controller fail, then dar0 must be capable of re-routing I/O requests through to the second controller.

CORRECT CONFIGURATION

To test whether your configuration is likely to be viable, run the **fget_config –A** command, which is supplied with the *devices.fcp.disk.array.rte* fileset. If the **–A** option is not supported for the RDAC fileset currently installed, then use **fget_config –l dar0**. If the output looks something like:

```
----darØ----
dacØ ACTIVE dac1 ACTIVE
hdisk2          dacØ
hdisk3          dac1
```

and if there is just a single FAStT attached and no other routers (dar*x*s) are displayed in the output, then the configuration is most likely correct. If you had to use **fget_config –l**, then repeat the command with **dar1** to convince yourself that a **dar1** has not been configured. Older versions of the RDAC fileset will produce slightly different output such as:

```
----darØ----
dacØ ACTIVE dac1 ACTIVE
dacØ-hdisk2
dac1-hdisk3
```

If you run the **lsattr –El dar0** command, you should also get output similar to:

```
act_controller    dacØ,dac1    Active Controllers                  False
all_controller    dacØ,dac1    Available Controllers               False
.
.
cache_size        1Ø24         Cache size for both controllers     False
.
```

This shows that a single disk array router, dar0, can see and use both controllers, dac0 and dac1. The output will also display the amount of cache available. If dac0 (controller A in the above example) is unavailable, then hdisk2 will become available through controller B (dac1) and **fget_config** will now show:

```
hdisk2          dac1
hdisk3          dac1
```

INCORRECT CONFIGURATION

If, however, the output from **fget_config -A** (or similar commands using the **–l** option) acquires some mysterious **dacNONE**s, and looks something like:

```
----darØ----
dacØ ACTIVE dacNONE ACTIVE
hdisk2          dacNONE

----dar1----
dac1 ACTIVE dacNONE ACTIVE
hdisk3          dac1
```

then you have problems.

When this output is displayed, **cfgmgr** has created the second router and then associated just one of the controllers with each router; usually dac0 with dar0, and dac1 with dar1. Any disks that have already been created on the FAStT and mapped to the host are then configured in AIX and associated with their preferred controller. Note that when you have a faulty configuration, it may take several executions of **cfgmgr** to configure all the disks that have been mapped to the host. The presence of the second router along with the *dacNONE*s means that you will never get a failover to a standby controller since the disks now have a one-to-one relationship with a controller and router.

If you also run the **lsattr –El dar0** command, you will now get output similar to:

```
act_controller     dacØ        Active Controllers                  False
```

```
all_controller     dacØ        Available Controllers            False
.
.
cache_size         Ø           Cache size for both controllers    False
.
```

Similarly, the **lsattr –El dar1** command will display dac1 for both the *Active* and *Available Controllers*. The output from both of these commands will also show the cache size as 0.

HOW, WHY, AND WHEN?

Perhaps the simple answer to this, apart from a few educated guesses, is – I haven't a clue!

The only certainty is that, if the problem is going to appear, it appears only when **cfgmgr** is run while both controllers are active/online. Although the intended design was undoubtedly to run **cfgmgr** with both controllers active, for reasons as yet unknown (to me), this sometimes creates the above features for which a satisfactory cause has not yet been found. Or if it has, then IBM is keeping it a close secret – since it is not mentioned in the latest *Storage Manager 8.4 Installation and Support Guide for AIX, HP-UX and Solaris*! Note that the *dacNONE* and multiple dar features have been around at least since Storage Manager 7.*x*.

Experience has shown that the problem appears not to be directly connected with zoning (although IBM AIX support may imply that it is – after extensive testing, the same problems have been discovered both with and without zoning on the SAN switches), nor is it likely to be in any way connected with the fibre cabling (unless it is truly in a mess), and it does not seem to be dependent on the type of Fibre Channel adapter installed, nor the model of FAStT, nor the various levels of RDAC software.

Even more frustratingly, this annoying feature does not always raise its ugly head and is believed to be in some way connected with the order in which the software and upgrades for the RDAC and Fibre Channel devices are installed, and at what stage **cfgmr** is run – although the exact sequence is not at all clear. Some installations have worked properly from the word go, while others using the same FAStT, and the same versions of AIX and RDAC software (although on different types of machine), have produced problems.

Similar problems have been experienced with AIX 4.3, 5.1, and 5.2 at various maintenance levels, and it probably affects all current levels of AIX.

SOLUTION

Equally frustrating is that sometimes the faulty configuration is easy to resolve and at other times it requires a different approach. The IBM mega-databases are full of 'solutions' that seem to work at some sites but not at others.

The following is a 'solution'that seems to work every time. A word of warning, though: it involves taking controllers offline, and, if you are configuring a host which is being attached to a FAStT that is currently in a production environment, you will probably have to plan downtime to stop all I/O to the FAStT

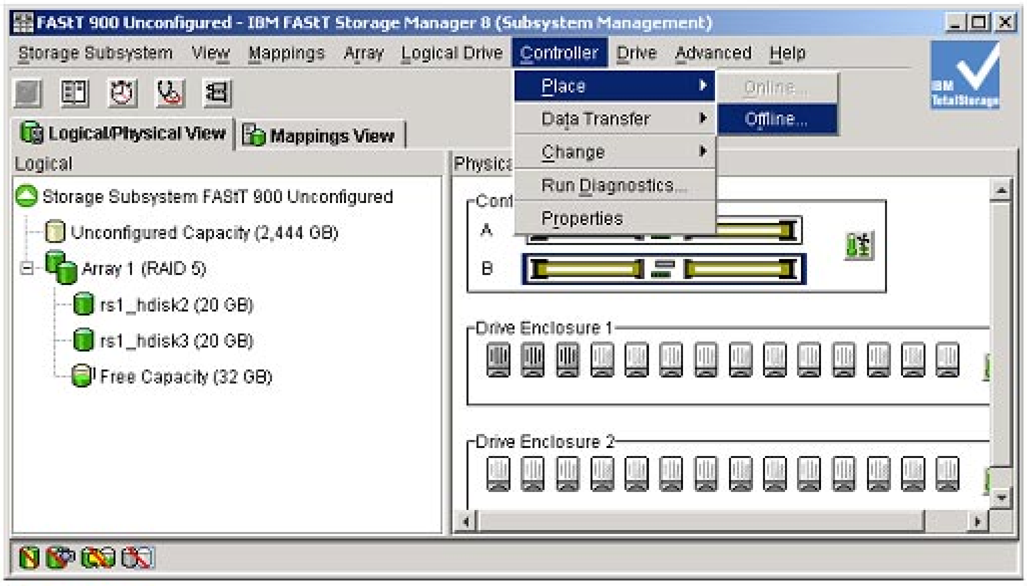

Figure 2: Taking controller B offline

disks. For completely new installations, however, this should not be a problem.

If you have *dacNONE*s and multiple unexpected dar*x*s, first delete all currently configured disks, routers, and controllers:

```
rmdev -Rdl darØ
rmdev -Rdl dar1
rmdev -dl dacØ
rmdev -dl dac1
```

You can try running **cfgmgr –v** again, followed by **fget_config –A**, to see whether the problem has resolved itself. This occasionally works. If it does not, delete the disks, routers, and controllers again.

In the Subsystem Management Window, now select the **Logical/Physical View** and disable Controller B by taking it Offline. To do this, select the controller by clicking on it in the Controller Enclosure box in the Physical view, and then **Controller\Place\Offline** – see Figure 2.

When you try to take a controller offline, you will get a warning message stating that if you make the change while an application is using any associated logical drives I/O errors will be caused unless there is a multi-path driver installed on the host. So beware.

Unfortunately, when you select *Yes* in the warning message, the FAStT will start honking at you, but ignore this for the time being – you may have to reassure operators that nothing serious is happening!

Now run **cfgmgr –v** to reconfigure the router, controllers, and disks. When you run **fget_config –A** the output will now be something like:

```
----darØ----
dacØ ACTIVE dacNONE RESET
hdisk2      dacØ
hdisk3      dacØ
```

or:

```
----darØ----
dacØ ACTIVE dacNONE RESET
dacØ-hdisk2
dacØ-hdisk3
```

depending on which version of *devices.fcp.disk.array.rte* is installed.

Still not right, but at least we do not have a dar1.

Now right-click on controller B in the controller enclosure, select **Place**, then **Online**, and bring it back online. When the honking subsides, run **fget_config –A** again and it should now show:

```
----darØ----
dacØ ACTIVE dac1 ACTIVE
hdisk2      dacØ
hdisk3      dac1
```

If it still shows *dacNONE*, such as:

```
dacØ ACTIVE dacNONE ACTIVE
```

and this is a distinct possibility, run **cfgmgr** again. This will get rid of the *dacNONE* and make dac1 ACTIVE. Check that everything is correct using **fget_config –A**.

Note that by taking one of the controllers offline, a number of LUNs (depending on how many have been configured) will no longer be on their preferred controllers. You will have to work your way through all the logical drives configured for each array in the **Logical/Physical View** and change the ownership back to the preferred controller. When all drives are back on their preferred controllers, the red error triangle against the Storage Subsystem name will change back to green.

Richard Handforth
AIX and HACMP Software Specialist (UK)

Graphing system performance based on SAR output

The SAR output command is one of the most effective and underrated tools for analysing system performance on Unix servers. SAR prepares reports using the system activity counters available in the Unix kernel and stores available data historically in log files. Like most tools for measuring performance, SAR provides a lot of data but little analysis – because it has only raw numbers. So it is up to the user to interpret the numbers and determine how a system is performing. That's why I have developed an automatic mechanism that works every month to collect the SAR output from all the AIX servers we have, graphing the system performance, and finally sending the graphics to my managers/ other people via Outlook mail. The old performance graphics for a server are stored in a directory so that past performance can be analysed when it is needed.

In our company we have groups of servers that are dedicated to different projects. So, I have designed my monthly performance report such that I have a file for each project consisting of three graphics (work hours/non-work hours/24 hours graphic) for each server, and three average graphics (work hours/non-work hours/24 hours graphic) for the whole project. To make it simpler, let's assume we have a ProjectA consisting of three servers named myserver1, myserver2, and myserver3.

The first step is to set the SAR files in each server (myserver1, myserver2, and myserver3) so they produce the same format of output.

Change *The System Activity Reports* entry in */var/spool/cron/ crontabs/adm* so it looks like this:

```
#=====================================================
#       SYSTEM ACTIVITY REPORTS
#  8am-5pm activity reports every 20 mins during weekdays.
#  activity reports every an hour on Saturday and Sunday.
#  6pm-7am activity reports every an hour during weekdays.
#  Daily summary prepared at 18:05.
#=======================================================
#0 8-17 * * 1-5 /usr/lib/sa/sa1 1200 3 &
#0 * * * 0,6 /usr/lib/sa/sa1 &
#0 18-7 * * 1-5 /usr/lib/sa/sa1 &
#5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -i 3600 -ubcwyaqvm &
30 * * * * /usr/lib/sa/sa1 600 6 &
#=======================================================
```

An example of SAR output would look like this:

```
--------------------------------------------
AIX myserver1 3 4 00409AFA4C00    03/04/04

00:30:00    %usr    %sys    %wio   %idle
00:40:00       5      15      14      66
00:50:00       9      25      14      52
01:00:00       9      23      21      47
01:10:00       8      20      22      50
01:20:00      10      24      22      45
01:30:00      10      17      39      34
01:40:00      11      19      47      23
01:50:00      10      17      48      25
…………………………………………………………..
23:00:00       1       3       1      96
23:10:00       2       4       5      89
23:20:00       0       2       1      97
23:30:00       0       2       1      97
23:40:00       0       2       1      97
23:50:00       1       2       5      92
00:00:00       3       6      31      60
00:10:00      11      26      26      38
00:20:01       9      23      19      49

Average        8       9      42      41
--------------------------------------------
```

You can check it by:

```
# sar –f /var/adm/sa/sa03
```

(for the date 03/03/2004).

We selected one of our machines as the performance server (myserver1), and the others are clients (myserver2 and myserver3), which will send their output to the performance server.

I recommend that you create a separate user (perf) on each server (myserver1, myserver2, and myserver3) for the file transfers (FTP) between servers.

SAR.SH

The first script, SAR.SH, works on every server in ProjectA (myserver1, myserver2, and myserver3) at the end of the month to collect and calculate SAR output for that month for the desired time ranges.

SAR.SH creates three files for each server. For example, the files created for myserver2 are:

- *sar_myserver2_24.out* – the 24-hour average SAR output.
- *sar_myserver2_8.out* – the work hours average, 09:00 – 18:00 SAR output.
- *sar_myserver2_rest.out* – the non-work hours average: 18:00 – 09:00 SAR output.

The versions of these files created on myserver2 and myserver3 will be sent to myserver1 via FTP. The files created on myserver1 will be copied to the directory */home/perf…*.

```
#!/bin/ksh

###################################################
#
# Adnan Akbas , Turkcell , 25.04.2002
#
# This script works from crontab at the end of the
# month to collect and calculate SAR output for
# myserver1, myserver2, and myserver3 for the
# month with three different time ranges:
#
# * work hours     (09:00 - 18:00)
# * non-work hours (18:00 - 09:00)
# * 24 hours       (00:00 - 24:00)
#
# If the script works on a client, these 3
# files will be sent to the perf server (myserver1)
# via FTP! If the script works on the perf server
# (myserver1) these 3 files will be saved to the
# perf home directory (/home/perf) …
```

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

# This function is for adding entries

function add {

# initializing variables

let i=0
let tot_usr=0
let tot_sys=0
let tot_wio=0

while read usr sys wio ; do

if [[ $usr = +([0-9]) ]] || [[ $sys = +([0-9]) ]] || [[ $wio = +([0-9])
]] ; then

let tot_usr=${tot_usr}+$usr
let tot_sys=${tot_sys}+$sys
let tot_wio=${tot_wio}+$wio
let i=$i+1

fi
done

}

logfile=/tmp/ftp.out
let today='date +%d'

# Defining the time ranges

sar_8="09 10 11 12 13 14 15 16 17"
sar_rest="18 19 20 21 22 23 00 01 02 03 04 05 06 07 08"

# Collecting  24 hours average SAR outputs (00:00 - 24:00)

echo "$(hostname) 24" > /home/perf/sar_`hostname`_24.out
echo "date %usr %sys %wio %idle" >> /home/perf/sar_`hostname`_24.out

ls -l /var/adm/sa | awk '{print $9}'| grep . |
while read data ; do
  da='echo $data | cut -c3-4'
  average='sar -f /var/adm/sa/${data} | grep Average | awk '{print $2,
$3, $4, $5}''
  echo "${da} $average" | grep -v NaNQ | read a b c d e junk

# getting rid of the "-" character that appears in SAR outputs and reset
```

```
  echo "$a $b $c $d $e" | grep "-" > /dev/null 2>&1
  if [[ $? = Ø ]] ; then
  b=Ø; c=Ø; d=Ø; e=1ØØ
  fi

  echo "${a}. $b $c $d $e" >> /home/perf/sar_`hostname`_24.out
done

# Calculating work hours average SAR outputs (Ø9:ØØ - 18:ØØ)

echo "$(hostname) 8" > /home/perf/sar_`hostname`_8.out
echo "date %usr %sys %wio %idle" >> /home/perf/sar_`hostname`_8.out
ls -l /var/adm/sa | awk '{print $9}'| grep . |
while read data ; do
  da='echo $data | cut -c3-4'
  rm /tmp/sar_8.out
  for ent in $sar_8
  do
  sar -f /var/adm/sa/${data} | grep "^${ent}" >> /tmp/sar_8.out
  done
  cat /tmp/sar_8.out | awk '{print $2, $3, $4}'| add

# Something wrong!, cannot devide with zero

  if [[ $i = Ø ]] ; then
     echo "error has occured"
     exit 1
  fi

  let tot_usr_avr=${tot_usr}/$i
  let tot_sys_avr=${tot_sys}/$i
  let tot_wio_avr=${tot_wio}/$i
  let tot_idle_avr=1ØØ-${tot_usr_avr}-${tot_sys_avr}-${tot_wio_avr}

# getting rid of the "-" character that appears in SAR outputs and reset

  echo "$tot_usr_avr $tot_sys_avr $tot_wio_avr $tot_idle_avr" | grep "-"
> /dev/null 2>&1
  if [[ $? = Ø ]] ; then
  tot_usr_avr=Ø; tot_sys_avr=Ø tot_wio_avr=Ø tot_idle_avr=1ØØ
  fi
  echo "${da}. $tot_usr_avr $tot_sys_avr $tot_wio_avr $tot_idle_avr" >>
/home/perf/sar_`hostname`_8.out
done

# Calculating non-work hours average SAR outputs (18:ØØ - Ø9:ØØ)

echo "$(hostname) rest" > /home/perf/sar_`hostname`_rest.out
echo "date %usr %sys %wio %idle" >> /home/perf/sar_`hostname`_rest.out
```

```
ls -l /var/adm/sa | awk '{print $9}'| grep . |
while read data ; do
  da='echo $data | cut -c3-4'
  rm /tmp/sar_rest.out
  for ent in $sar_rest
  do
  sar -f /var/adm/sa/${data} | grep "^${ent}" | grep -v "^00:30" >> /
tmp/sar_rest.out
  done
  cat /tmp/sar_rest.out | grep . | awk '{print $2, $3, $4}'| add
  if [[ $i = 0 ]] ; then
     echo „error has occured"
  fi
  let tot_usr_avr=${tot_usr}/$i
  let tot_sys_avr=${tot_sys}/$i
  let tot_wio_avr=${tot_wio}/$i
   let tot_idle_avr=100-${tot_usr_avr}-${tot_sys_avr}-${tot_wio_avr}

# getting rid of the "-" character that appears in SAR outputs and reset

  echo "$tot_usr_avr $tot_sys_avr $tot_wio_avr $tot_idle_avr" | grep "-"
> /dev/null 2>&1
  if [[ $? = 0 ]] ; then
  tot_usr_avr=0; tot_sys_avr=0 tot_wio_avr=0 tot_idle_avr=100
  fi

  echo "${da}. $tot_usr_avr $tot_sys_avr $tot_wio_avr $tot_idle_avr" >>
/home/perf/sar_`hostname`_rest.out
done

# changing the permissions of the files
chown perf:staff /home/perf/sar_'hostname'_*

perf_server=myserver1

# if it works on clients, sends the files (ftp) to performance server
(myserver1) ...

if [ 'hostname' != $perf_server ]
then

perf_user=perf
perf_password=projectA
target=myserver1

ftp -v -n ${target} << ! >> $logfile 2>&1
user $perf_user $perf_password
lcd /home/perf
cd /home/perf
mput sar_'hostname'_*
```

```
bye
!

fi
```

SAR_AVR.SH

The second script, SAR_AVR.SH, works on the performance server (myserver1) after all the SAR output has arrived on myserver1 (after sar.sh), and calculates the average performance for the whole ProjectA (myserver1+myserver2+myserver3).

It then saves the average output files in the directory */home/perf* where all the other output is collected, and uploads the overall file (sar_all.out) to a Windows PC.

```
#!/bin/ksh

####################################################
#
# Adnan Akbas , Turkcell , 28.04.2002
#
# This script works from crontab after sar_client.sh
# and sar_server.sh to calculate the average
# performance for the whole ProjectA. It uses all
# output that has arrived and calculates the averages for
# the month with three different time ranges:
#
# * work hours     (09:00 - 18:00)
# * non-work hours (18:00 - 09:00)
# * 24 hours       (00:00 - 24:00)
#
# Finally, these 3 average files will be saved to
# the perf home directory (/home/perf) and FTPs
# sar_all.out to a Windows PC.
#
####################################################

# variables ######################################

# list of servers in this project. The sequence is important

host_list="myserver1 myserver2 myserver3" # the sequence is important
let host_num=3
month_days="01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30 31"
```

```
time_interval="24 8 rest"
perf_dir=/home/perf

# check whether all perf files are there and are not null

for int in $time_interval
do
  for hostn in $host_list
  do
  if [ ! -s  ${perf_dir}/sar_${hostn}_${int}.out ] ; then
   exit 8
  fi
  done
done

# proccessing the files to calculate averages

for int in $time_interval
do
  echo "Average ${int}" > /home/perf/sar_average_${int}.out
  echo "date %usr %sys %wio %idle" >> /home/perf/sar_average_${int}.out

  for gun in $month_days
  do

    # initializing the counters

    let tot_usr=0
    let tot_sys=0
    let tot_wio=0

    for hostn in $host_list
    do

      let usr='grep "^${gun}" ${perf_dir}/sar_${hostn}_${int}.out | awk
'{print $2}''
      let tot_usr=${tot_usr}+$usr

      let sys='grep "^${gun}" ${perf_dir}/sar_${hostn}_${int}.out | awk
'{print $3}''
      let tot_sys=${tot_sys}+$sys

      let wio='grep "^${gun}" ${perf_dir}/sar_${hostn}_${int}.out | awk
'{print $4}''
      let tot_wio=${tot_wio}+$wio
    done
    # taking the averages
    let avr_usr=${tot_usr}/${host_num}
    let avr_sys=${tot_sys}/${host_num}
    let avr_wio=${tot_wio}/${host_num}
    let avr_idle=100-${avr_usr}-${avr_sys}-${avr_wio}
    echo "${gun}. $avr_usr $avr_sys $avr_wio $avr_idle" >> ${perf_dir}/
sar_average_${int}.out
  done
done
chown perf:staff ${perf_dir}/sar_average_*
# make all one file
rm ${perf_dir}/sar_all.out > /dev/null 2>&1
touch ${perf_dir}/sar_all.out
for int in $time_interval
do
  for hostn in $host_list
  do
    cat ${perf_dir}/sar_${hostn}_${int}.out >> ${perf_dir}/sar_all.out
  done
done
# Plus the average files:
cat ${perf_dir}/sar_average_24.out >> ${perf_dir}/sar_all.out
cat ${perf_dir}/sar_average_8.out >> ${perf_dir}/sar_all.out
cat ${perf_dir}/sar_average_rest.out >> ${perf_dir}/sar_all.out
# Uploading the file into windows enviroment
winuser=perf
winpassword=perf123
target_pc=10.44.205.63
local_dir=/home/perf
target_dir=e:/perf/projectA
cd $local_dir
ftp -v -n ${target_pc} << !
user $winuser $winpassword
prompt
ascii
cd ${target_dir}
put sar_all.out
bye
!
# after ftp, moving the files under bck directory in case later needed
mv ${perf_dir}/sar*.out ${perf_dir}/bck
```

An example sar_all.out file is shown below.

For every server three files are output each month:

- 24 – 24-hour average SAR output.
- 8 – the work hours average (09:00 – 18:00 SAR output).
- rest – the non-work hours average (18:00 – 09:00 SAR output).

```
myserver1 24
date %usr %sys %wio %idle
Ø1. 5 Ø 1 93
Ø2. 23 1 2 74
Ø3. 11 1 2 86
Ø4. 9 1 2 87
Ø5. 7 1 2 89
Ø6. 13 2 2 83
Ø7. 5 Ø 1 94
Ø8. 5 1 1 93
Ø9. 26 1 2 71
1Ø. 26 1 2 7Ø
11. 23 1 2 73
12. 16 1 2 81
13. 25 2 2 72
14. 3 1 1 95
15. 2 Ø 1 97
16. 8 1 2 89
17. 8 1 2 89
18. 11 2 2 85
19. 8 1 2 89
2Ø. 13 1 2 84
21. 16 1 1 82
22. 5 1 Ø 94
23. 26 1 2 71
24. 28 1 3 68
25. 29 1 2 68
26. 27 2 2 69
27. 14 2 2 82
28. 2 1 1 96
29. 1 Ø 1 97
3Ø. 13 1 2 84
31. 4 1 1 94
myserver2 24
.......................
myserver3 24
.......................
myserver1 8
.......................
.......................
```

There are crontab entries for the scripts so they are executed automatically.

For myserver1, the entries are:

```
Ø Ø 1 * * /usr/local/sbin/sar.sh > /dev/null 2>&1
Ø 1 1 * * /usr/local/sbin/sar_avr.sh > /dev/null 2>&1
```

For myserver2 and myserver3 the entry is:

```
Ø Ø 1 * * /usr/local/sbin/sar.sh > /dev/null 2>&1
```

(0 0 1 * * is the first day of the month at 00:00 to get the previous month's SAR output before the new month's output overwrites the older ones.)

PERF.VBP

The third script is PERF.VBP (Visual Basic Program). This program works on the Windows PC and will produce graphs of the system performance of the AIX servers in ProjectA (myserver1, myserver2, and myserver3) using the raw numbers in overall SAR output files (sar_all.out). The graphics will be copied every month to a shared Unix drive (*F:/perf/*) and sent to the managers via Outlook mail.

```
Private Sub Form_Load()
'Defining excel varables
Dim xlApp As Excel.Application
Dim xlBook As Excel.Workbook
Dim xlSheet As Excel.Worksheet
Dim dirname As String
Dim nowdate As Date
Dim prevdate As Date
Form1.Visible = False
dirname = "ProjectA"
'Adding excel object
Set xlApp = CreateObject("Excel.Application")
xlApp.Visible = False
Set xlBook = xlApp.Workbooks.Add
Set xlSheet = xlBook.Worksheets(1)
'Time variables and finding the previous /day/month/year
nowdate = Date
prevdate = DateAdd("d", -1, nowdate)
prevmonth = Format(prevdate, "mmmm")
prevyear = Format(prevdate, "yyyy")
today = Format(prevdate, "d")
RichTextBox1.LoadFile ("E:\perf\" & dirname & "\sar_all.out")
RichTextBox1.SelStart = Ø: RichTextBox1.SelLength =
Len(RichTextBox1.Text)
Clipboard.Clear
Clipboard.SetText RichTextBox1.SelText
'Paste the Clipboard to the active Sheet
ActiveSheet.Paste
Selection.TextToColumns DataType:=xlDelimited,
ConsecutiveDelimiter:=True, Space:=True
'Mark the cells
```

```
xlSheet.Range("A1:E396").Select
'-------------------------------------------------------
'myserver2 Graphics
'-------------------------------------------------------
Set myserver2_rest = xlApp.Charts.Add
With myserver2_rest
.ChartType = xlColumnStacked100
.SetSourceData Source:=xlSheet.Range("A233:E" & 233 + today),
PlotBy:=xlColumns
.Name = "myserver2(18-08)"
.HasTitle = True
.ChartTitle.Text = "ProjectA (myserver2)" & vbCrLf & _
                    prevmonth & " - " & prevyear & " ; (18:00 - 08:00)"
.Axes(xlValue).HasTitle = False
End With
myserver2_rest.SeriesCollection(4).Interior.ColorIndex = 2
'-------------------------------------------------------
Set myserver2_24 = xlApp.Charts.Add
With myserver2_24
.ChartType = xlColumnStacked100
.SetSourceData Source:=xlSheet.Range("A35:E" & 35 + today),
PlotBy:=xlColumns
.Name = "myserver2(00-24)"
.HasTitle = True
.ChartTitle.Text = "ProjectA (myserver2)" & vbCrLf & _
                    prevmonth & " - " & prevyear & " ; (00:00 - 24:00)"
.Axes(xlValue).HasTitle = False
End With
myserver2_24.SeriesCollection(4).Interior.ColorIndex = 2
'-------------------------------------------------------
Set myserver2_8 = xlApp.Charts.Add
With myserver2_8
.ChartType = xlColumnStacked100
.SetSourceData Source:=xlSheet.Range("A134:E" & 134 + today),
PlotBy:=xlColumns
.Name = "myserver2(09-17)"
.HasTitle = True
.ChartTitle.Text = "ProjectA (myserver2)" & vbCrLf & _
                    prevmonth & " - " & prevyear & " ; (09:00 - 17:00)"
.Axes(xlValue).HasTitle = False
End With
myserver2_8.SeriesCollection(4).Interior.ColorIndex = 2
'-------------------------------------------------------
'myserver1 Graphics
'-------------------------------------------------------
Set myserver1_rest = xlApp.Charts.Add
With myserver1_rest
.ChartType = xlColumnStacked100
.SetSourceData Source:=xlSheet.Range("A200:E" & 200 + today),
PlotBy:=xlColumns
```

```
.Name = "myserver1(18-08)"
.HasTitle = True
.ChartTitle.Text = "ProjectA (myserver1)" & vbCrLf & _
                    prevmonth & " - " & prevyear & " ; (18:00 - 08:00)"
.Axes(xlValue).HasTitle = False
End With
myserver1_rest.SeriesCollection(4).Interior.ColorIndex = 2
'-------------------------------------------------------
Set myserver1_24 = xlApp.Charts.Add
With myserver1_24
.ChartType = xlColumnStacked100
.SetSourceData Source:=xlSheet.Range("A2:E" & 2 + today),
PlotBy:=xlColumns
.Name = "myserver1(00-24)"
.HasTitle = True
.ChartTitle.Text = "ProjectA (myserver1)" & vbCrLf & _
                    prevmonth & " - " & prevyear & " ; (00:00 - 24:00)"
.Axes(xlValue).HasTitle = False
End With
myserver1_24.SeriesCollection(4).Interior.ColorIndex = 2
'-------------------------------------------------------
Set myserver1_8 = xlApp.Charts.Add
With myserver1_8
.ChartType = xlColumnStacked100
.SetSourceData Source:=xlSheet.Range("A101:E" & 101 + today),
PlotBy:=xlColumns
.Name = "myserver1(09-17)"
.HasTitle = True
.ChartTitle.Text = "ProjectA (myserver1)" & vbCrLf & _
                    prevmonth & " - " & prevyear & " ; (09:00 - 17:00)"
.Axes(xlValue).HasTitle = False
End With
myserver1_8.SeriesCollection(4).Interior.ColorIndex = 2
'-------------------------------------------------------
'myserver3 Graphics
'-------------------------------------------------------
Set myserver3_rest = xlApp.Charts.Add
With myserver3_rest
.ChartType = xlColumnStacked100
.SetSourceData Source:=xlSheet.Range("A266:E" & 266 + today),
PlotBy:=xlColumns
.Name = "myserver3(18-08)"
.HasTitle = True
.ChartTitle.Text = "ProjectA (myserver3)" & vbCrLf & _
                    prevmonth & " - " & prevyear & " ; (18:00 - 08:00)"
.Axes(xlValue).HasTitle = False
End With
myserver3_rest.SeriesCollection(4).Interior.ColorIndex = 2
'-------------------------------------------------------
Set myserver3_24 = xlApp.Charts.Add
```

```
With myserver3_24
.ChartType = xlColumnStacked100
.SetSourceData Source:=xlSheet.Range("A68:E" & 68 + today),
PlotBy:=xlColumns
.Name = "myserver3(00-24)"
.HasTitle = True
.ChartTitle.Text = "ProjectA (myserver3)" & vbCrLf & _
                    prevmonth & " - " & prevyear & " ; (00:00 - 24:00)"
.Axes(xlValue).HasTitle = False
End With

myserver3_24.SeriesCollection(4).Interior.ColorIndex = 2
'-------------------------------------------------------
Set myserver3_8 = xlApp.Charts.Add
With myserver3_8
.ChartType = xlColumnStacked100
.SetSourceData Source:=xlSheet.Range("A167:E" & 167 + today),
PlotBy:=xlColumns
.Name = "myserver3(09-17)"
.HasTitle = True
.ChartTitle.Text = "ProjectA (myserver3)" & vbCrLf & _
                    prevmonth & " - " & prevyear & " ; (09:00 - 17:00)"
.Axes(xlValue).HasTitle = False
End With
myserver3_8.SeriesCollection(4).Interior.ColorIndex = 2
'-------------------------------------------------------
'Average Graphics
'-------------------------------------------------------
Set average_rest = xlApp.Charts.Add
With average_rest
.ChartType = xlColumnStacked100
.SetSourceData Source:=xlSheet.Range("A365:E" & 365 + today),
PlotBy:=xlColumns
.Name = "Average(18-08)"
.HasTitle = True
.ChartTitle.Text = "ProjectA (Average)" & vbCrLf & _
                    prevmonth & " - " & prevyear & " ; (18:00 - 08:00)"
.Axes(xlValue).HasTitle = False
End With

average_rest.SeriesCollection(4).Interior.ColorIndex = 2
'-------------------------------------------------------
Set average_24 = xlApp.Charts.Add
With average_24
.ChartType = xlColumnStacked100
.SetSourceData Source:=xlSheet.Range("A300:E" & 300 + today),
PlotBy:=xlColumns
.Name = "Average(00-24)"
.HasTitle = True
.ChartTitle.Text = "ProjectA (Average)" & vbCrLf & _
```

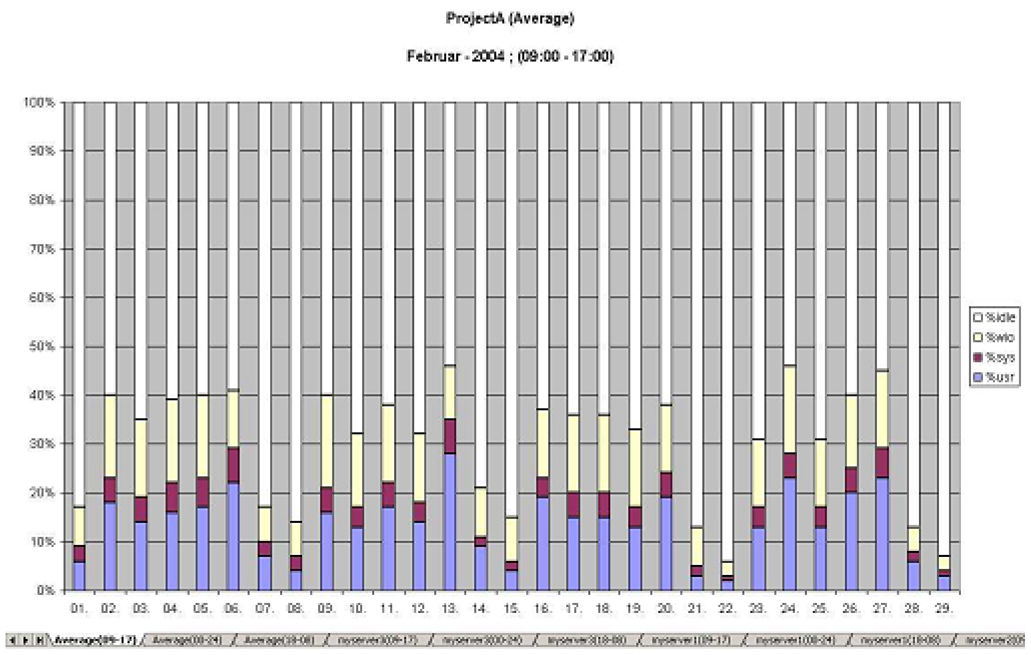

Figure 1: Working hours for ProjectA

```
                    prevmonth & " - " & prevyear & " ; (00:00 - 24:00)"
.Axes(xlValue).HasTitle = False
End With

average_24.SeriesCollection(4).Interior.ColorIndex = 2
'--------------------------------------------------------
Set average_8 = xlApp.Charts.Add
With average_8
.ChartType = xlColumnStacked100
.SetSourceData Source:=xlSheet.Range("A332:E" & 332 + today),
PlotBy:=xlColumns
.Name = "Average(09-17)"
.HasTitle = True
.ChartTitle.Text = "ProjectA (Average)" & vbCrLf & _
                    prevmonth & " - " & prevyear & " ; (09:00 - 17:00)"
.Axes(xlValue).HasTitle = False
End With
average_8.SeriesCollection(4).Interior.ColorIndex = 2
xlApp.ActiveWorkbook.SaveAs FileName:="F:\" & dirname & "\per-" &
dirname & "-" & DatePart("d", nowdate) & "-" & DatePart("m", nowdate) &
"-" & DatePart("yyyy", nowdate) & ".xls"
xlApp.Quit
```

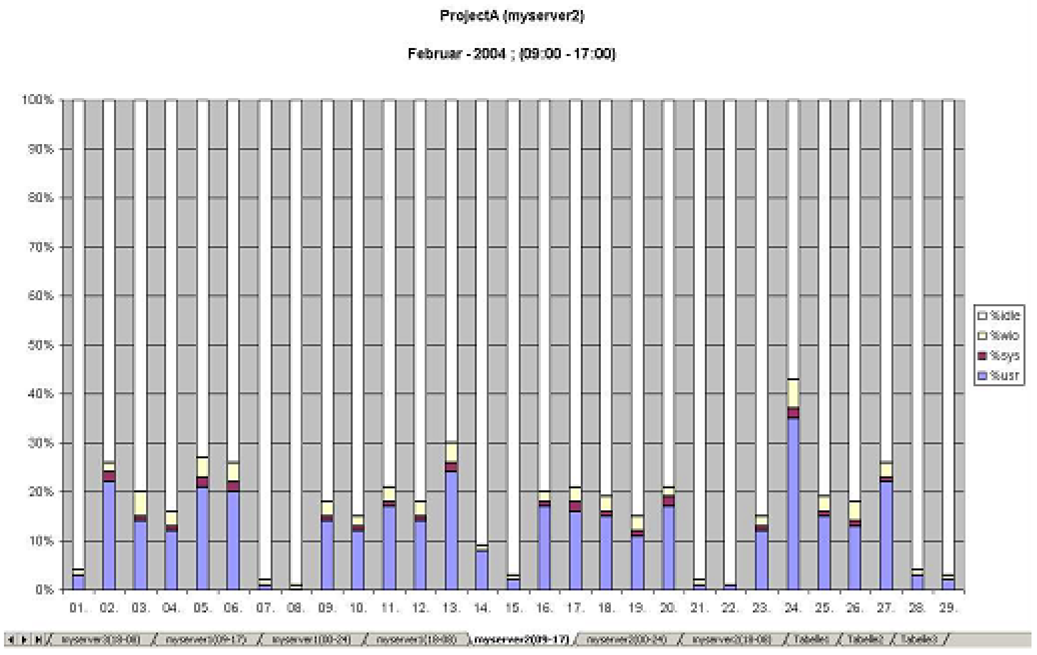

Figure 2: Myserver2 performanceduring the working day

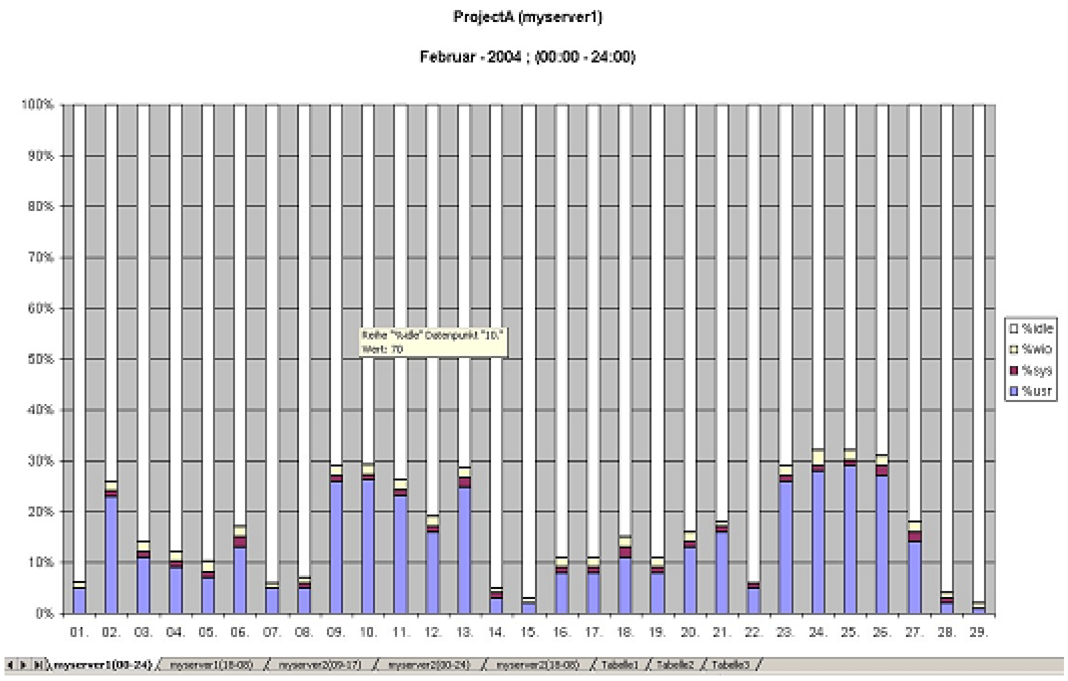

Figure 3: Myserver1 performance over 24-hour period

```
'***** Outlook mail *****
' Creating outlook object
Set myOLApp = CreateObject("Outlook.Application")
Set mynamespace = myOLApp.GetNamespace("MAPI")
Dim myOLItem As Outlook.MailItem
Set myOLItem = myOLApp.CreateItem(olMailItem)
    With myOLItem
        'my own email address
        .To = mynamespace.CurrentUser.Name
        '.CC = "trevore@xephon.com"
        .Subject = "Daily Automatic mail -" & dirname & "- system
performance report"
    End With
' Attaching the performance graphics file
myOLItem.Attachments.Add "F:\" & dirname & "\per-" & dirname & "-" &
DatePart("d", nowdate) & "-" & DatePart("m", nowdate) & "-" &
DatePart("yyyy", nowdate) & ".xls"
' sending the email to the related people
myOLItem.Send
' closing the applications created
Unload Me

End Sub
```

This VB program creates an Excel file consisting of all performance graphics belonging to ProjectA and sends it as an e-mail every month to inform the administrators/managers about the performance of the servers.

The monthly graphics in this Excel Workbook are:

- Three graphs showing overall ProjectA average – work hours, non-work hours, and over 24 hours.
- Three graphs showing myserver1 – work hours, non-work hours, and over 24 hours.
- Three graphs showing myserver2 – work hours, non-work hours, and over 24 hours.
- Three graphs showing myserver3 – work hours, non-work hours, and over 24 hours.

I recommend that you make an .exe file of this Visual Basic program (perf.exe) and schedule it on your Windows PC so that it works every month automatically after the sar_all.out file has arrived from the AIX performance server (myserver1).

An example graph showing the work done during work hours for Project A is shown in Figure 1.

Figure 2 shows performance for myserver2 during the working day.

Figure 3 shows performance over the whole 24-hour period for myserver1.

You can include other groups of servers in your company in the automatic graphing mechanism by simply changing the hostnames and directory names in the scripts.

Adnan Akbas
System Administrator
TURKCELL (Germany)

Tuning and problem determination techniques for AIX networking

PROBLEM DETERMINATION TOOLS

netstat command

One of the most useful tools for network problem determination on AIX is the **netstat** command.

netstat –I displays information about network interfaces configured on the system:

```
root@testhost:/home/root:  netstat -i
Name  Mtu   Network    Address           Ipkts    Ierrs  Opkts Oerrs  Coll
en0   1500  link#2     0.4.ac.de.33.88  20374020    0 42690661     0     0
en0   1500  192.168.5 testhost          20374020    0 42690661     0     0
en1   1500  link#3     0.4.ac.b1.4a.17   5647967   0   4778549     0     0
en1   1500  10.128     testhoste         5647967   0   4778549     0     0
lo0   16896 link#1                          7148   0      7208     0     0
lo0   16896 127        loopback             7148   0      7208     0     0
lo0   16896 ::1                             7148   0      7208     0     0
```

MTU value, MAC address, and IP address for each interfaces are displayed. If columns *Ierrs* or *Oerrs* contain a non-zero value you should check for a disconnected or malfunctioning network cable.

netstat –r displays the system’s routing table:

```
root@testhost:/home/root:  netstat -rn
Routing tables
Destination    Gateway           Flags   Refs     Use  If   PMTU Exp
Groups

Route Tree for Protocol Family 2 (Internet):
default        10.135.255.254    UG       17  4765809  en1     -   -
10.128/13      10.128.3.239      U         7    12163  en1     -   -
10.128.3.239   127.0.0.1         UGHS      0        9  lo0     -   -
127/8          127.0.0.1         U        10      678  lo0     -   -
192.168.5/24   192.168.5.26      U        24 42690510  en0     -   -
192.168.5.26   127.0.0.1         UGHS     16     5560  lo0     -   -

Route Tree for Protocol Family 24 (Internet v6):
::1              ::1             UH        0        0  lo0 16896   -
```

This report shows network gateways and interfaces used to reach a particular destination such as another host or a different network. The *Use* column shows how many times each route has been used.

netstat –m displays information about the memory allocated by networking support:

```
root@testhost:/home/root:  netstat -m
Kernel malloc statistics:
******* CPU 0 *******
By size        inuse     calls failed   delayed    free   hiwat   freed
32                68       368      0         0      60    7440       0
64                71      4552      0         0      57    3720       0
128               34      7184      0         0     126    1860       0
256               47     29200      0         0     289    4464       0
512               52  21918648      0         0     228    4464       0
1024              75     75812      0         0     121    1162       0
2048              29   1007886      0         0      15    1162       0
4096               1   1136645      0         0      38    1395       0
8192               2     71870      0         0       4     116       0
16384              2     17354      0         0     211     279       0
.
.
.
```

```
******* CPU 3 *******
By size        inuse     calls failed   delayed    free   hiwat   freed
32                 1       813      0         0     127    7440       0
64                58      4419      0         0      70    3720       0
128               42      6707      0         0      54    1860       0
256               26     22167      0         0     150    4464       0
512              121  33531847      0         0     143    4464       0
1024              62    108491      0         0      62    1162       0
2048               6   2671642      0         0      16    1162       0
4096               7   2937885      0         0      29    1395       0
8192               4    141536      0         0       4     116       0
16384              1     16281      0         0     212     279       0


***** Allocations greater than 16384 Bytes *****

By size        inuse     calls failed   delayed    free   hiwat   freed
65536              2         2      0         0       0   16384       0
Streams mblk statistic failures:
0 high priority mblk failures
0 medium priority mblk failures
0 low priority mblk failures
```

Check the values in columns headed *failed* – small values don't necessarily indicate a problem. Sometimes a kernel extension will attempt to allocate memory while processing an interrupt, when it may be temporarily unavailable. The column titled *inuse* is helpful for diagnosing memory leaks. If the numbers continually increase over time, and do not decrease again when the network load decreases, a memory leak may exist.

netstat –s displays a summary for networking protocol statistics:

```
root@testhost:/home/root:  netstat -s
udp:
        129874 datagrams received
        0 incomplete headers
        0 bad data length fields
        0 bad checksums
        45883 dropped due to no socket
        3100 broadcast/multicast datagrams dropped due to no socket
        13 socket buffer overflows
        80878 delivered
        9513 datagrams output
tcp:
        47457531 packets sent
                45555844 data packets (3544564882 bytes)
                26502 data packets (1547130б bytes) retransmitted
                1798111 ack-only packets (16387 delayed)
                2 URG only packets
                2363 window probe packets
                64147 window update packets
                10562 control packets
                0 large sends
                0 bytes sent using largesend
                0 bytes is the biggest largesend
        25528969 packets received
                21329791 acks (for 3544566709 bytes)
                241757 duplicate acks
                0 acks for unsent data
                4040477 packets (1924899453 bytes) received in-sequence
                2173 completely duplicate packets (1026395 bytes)
                0 old duplicate packets
                27 packets with some dup. data (5107 bytes duped)
                39354 out-of-order packets (19459419 bytes)
                2736 packets (2736 bytes) of data after window
                2736 window probes
                87943 window update packets
                68 packets received after close
                0 packets with bad hardware assisted checksum
                0 discarded for bad checksums
                0 discarded for bad header offset fields
                0 discarded because packet too short
                10 discarded by listeners
                0 discarded due to listener's queue full
                18973373 ack packet headers correctly predicted
                3767172 data packet headers correctly predicted
.
.
.
```

Look especially at error statistics summaries – large values indicate problems within the network.

netstat –a displays active network connections:

```
.
.
.
tcp4    0    0  *.1008            *.*                     LISTEN
tcp4    0    0  *.writesrv        *.*                     LISTEN
tcp4    0    0  *.tnslsnr         *.*                     LISTEN
tcp4    0    0  *.shilp           *.*                     LISTEN
tcp4    0    0  *.2701            *.*                     LISTEN
tcp4    0    0  loopback.49213    *.*                     LISTEN
```

```
tcp4      0      0  testhoste.32777    amrsp-aw1.amdocs.shilp  ESTABLISHED
tcp4      0      0  testhoste.32778    amrsp-bw2.amdocs.shilp  ESTABLISHED
tcp4      0      0  *.omni               *.*                    LISTEN
tcp4      0      0  testhost.smux      testhost.32784          ESTABLISHED
tcp4      0      0  testhost.32784     testhost.smux           ESTABLISHED
tcp4      0      0  *.6000               *.*                    LISTEN
tcp4      0      0  *.dtspc              *.*                    LISTEN
tcp4      0      0  testhost.tnslsnr   testhost.32785          ESTABLISHED
tcp4      0      0  testhost.32785     testhost.tnslsnr        ESTABLISHED
tcp4      0   1254  testhoste.telnet   10.17.98.143.2775       ESTABLISHED
tcp4      0      0  *.32768              *.*                    LISTEN
tcp4      0      0  *.32769              *.*                    LISTEN
tcp4      0      0  *.32774              *.*                    LISTEN
tcp4      0      0  *.32775              *.*                    LISTEN
tcp4      0      0  *.32776              *.*                    LISTEN
tcp4      0      0  *.32786              *.*                    LISTEN
tcp4      0      0  testhoste.netbios-  asherd05.3636          ESTABLISHED
tcp4      0      0  *.wsmserve             *.*                  LISTEN
tcp4      0      0  testhost.login      host5001e.amdocs..1023 ESTABLISHED
tcp4      0      0  testhost.login      host5003a.1020         ESTABLISHED
tcp4      0      0  testhost.telnet     newhost2.63577         ESTABLISHED
tcp4      0      0  testhost.32769      testhost.32770         ESTABLISHED
tcp4      0      0  testhost.32770      testhost.32769         ESTABLISHED
.
.
.
```

Recv-Q and *Send-Q* columns show whether data is sitting on a socket buffer. Data in *Recv-Q* could indicate that the local application is not reading. Data on the *Send-Q* could indicate that the remote application is not reading.

traceroute command

The **traceroute** command is useful for diagnosing routing problems. Every gateway taken by the network packets to reach the designated destination are displayed. This command is especially handy for identifying routing loops and routes with missing or incorrect settings.

```
root@testhost:/home/root:  traceroute host5001
trying to get source for host5001
source should be 10.128.3.239
traceroute to host5001.amdocs.com (199.221.37.21) from 10.128.3.239
(10.128.3.239), 30 hops max
outgoing MTU = 1500
 1  10.135.255.254 (10.135.255.254)  3 ms  1 ms  1 ms
 2  147.235.224.13 (147.235.224.13)  1 ms  1 ms  1 ms
 3  10.1.1.17 (10.1.1.17)  7 ms *  9 ms
 4  10.5.2.14 (10.5.2.14)  6 ms  6 ms  7 ms
 5  147.235.224.5 (147.235.224.5)  5 ms  6 ms  6 ms
 6  147.235.224.6 (147.235.224.6)  6 ms  6 ms  6 ms
 7  host5001.amdocs.com (199.221.37.21)  8 ms  9 ms  9 ms
```

Network tracing commands – tcpdump, iptrace, and ipreport

The **tcpdump**, **iptrace**, and **ipreport** commands are used to trace and monitor network traffic. They are very useful because every packet transmitted over the network can be captured, formatted, and displayed for inspection and control. **Iptrace** captures the network data and **ipreport** formats it according to specified options.

Tcpdump captures and displays packets according to specified parameters such as source and target host, packet type, etc.

```
root@testhost:/home/root:  tcpdump -I
tcpdump: listening on en0
17:15:28.699836508 0:e0:1e:e5:b2:53 1:80:c2:0:0:0 0026 38:
                       4242 0300 0000 0000 000a 0050 bde3 9802
                       0000 0000 000a 0050 bde3 9802 8144 0000
                       1400 0200 0f00
17:15:30.699804367 0:e0:1e:e5:b2:53 1:80:c2:0:0:0 0026 38:
                       4242 0300 0000 0000 000a 0050 bde3 9802
                       0000 0000 000a 0050 bde3 9802 8144 0000
                       1400 0200 0f00
17:15:32.699820969 0:e0:1e:e5:b2:53 1:80:c2:0:0:0 0026 38:
                       4242 0300 0000 0000 000a 0050 bde3 9802
                       0000 0000 000a 0050 bde3 9802 8144 0000
                       1400 0200 0f00
17:15:34.699846685 0:e0:1e:e5:b2:53 1:80:c2:0:0:0 0026 38:
                       4242 0300 0000 0000 000a 0050 bde3 9802
                       0000 0000 000a 0050 bde3 9802 8144 0000
                       1400 0200 0f00
17:15:36.074808161 hpp704.amdocs.com.12024 > 224.1.1.23.12024: udp 217
17:15:36.076287020 testhost.712 > newhost2.696: udp 88
17:15:36.078829255 newhost2.696 > testhost.712: udp 32
17:15:36.089230002 testhost.712 > newhost2.696: udp 88
17:15:36.089424251 newhost2.696 > testhost.712: udp 32
```

no command

The **no** command is used to display and modify network options. There are more than 100 network options in AIX; most of these are tuned rarely, but a few are commonly changed. In AIX 5.2, a new tuning framework has been introduced to enable persistent setting of the **no** options:

- **no –a** lists all the parameters along with their values.
- **no –L** produces a detailed list of **no** parameters including current, default, minimal, and maximal values possible.

A few of the options are 'reboot' options, meaning that they can be changed only during a reboot of the system:

- The **–p** flag sets an option currently and permanently.
- The **–r** flag sets an option for reboot only.

Interface-specific network options

In order to enable the setting of options only for selective network interfaces, interface-specific network options are supported. To use interface-specific options, the **no** option **use_isno** must be set to **1** (the default value). **Isno** options override the system-wide values set with **no**, but are overridden by values set by an application with the **setsockopt()** system call.

Five **isno** options can be set:

- tcp_sendspace
- tcp_recvspace
- rfc1323
- tcp_mssdflt
- tcp_nodelay

Interface-specific options can be set temporarily (until the next reboot) with the following command:

```
ifconfig en0 tcp_recvspace 65536
```

However, in order to set them permanently, the following command must be used:

```
chdev -l en0 -a tcp_recvspace=65536
```

ARP table size

The ARP table holds the mapping of IP addresses to MAC addresses. On AIX, it is a statically-sized hash table.

You know that your ARP table is too small if the output from the **arp –a** command shows that you have at least one bucket holding more than six entries or **iptrace** output shows repeated ARP requests for the same addresses. The **no** command options **arptab_nb** and **arptab_bsiz** control the size of the ARP table. **Arptab_nb** is the number of buckets in the hash table, and **arptab_bsiz** is the number of entries per bucket. **Arptab_bsiz** should not be changed. Change **arptab_nb** to a prime number so that **arptabnb** times **arptab_bsiz** equals the approximate number of entries needed by you. You need only a single ARP entry for all systems on your local networks; you don't need to count systems accessed over the routers.

Arptab_nb and **arptab_bsiz** are reboot options, which means that you need to reboot the server before the change will take effect.

Socket buffer sizes

Several options are related to socket buffer size. **sb_max** is the largest value that any socket buffer on the system can be set to. This allows the system administrator to limit the amount of memory an application can tie up on a socket buffer.

tcp_sendspace, **udp_sendspace**, **tcp_recvspace**, and **udp_recvspace** limit the amount of memory that can be put into a socket buffer. All of them must be less than or equal to **sb_max**. **tcp_sendspace**, **udp_sendspace**, **tcp_recvspace**, and **udp_recvspace** can be overridden by interface-specific values (set with the **ifconfig** command) or by socket options set by the application. Recommended values for the send and receive space options are at least ten times the MTU of the network adapter. If adapters with different MTUs are present on the system, you should use interface-specific network options to set the socket buffers to appropriate sizes for each interface.

rfc1323

If **tcp_recvspace** is set to a value larger than 65,535, the option **rsfc1323** must be set to **1** in order to take full advantage of the socket buffer size.

The **rfc1323** option enables an enhancement to the TCP protocol that allows TCP to advertise a window size larger than 65,535. **rfc1323** can also be set on an interface-specific basis.

Path MTU discovery

Two **no** options control path MTU discovery:

- **tcp_pmtu_discover** enables it for all TCP connections.
- **udp_pmtu_discover** enables it for all UDP, if the UDP application supports it.

The recommended value for both options is **1** (on) for best performance.

Path MTU, once discovered, is stored in the route to the host. If no specific host route exists, one is cloned from the network route that was used to reach the destination.

Cloned routes expire after **route_expire** minutes of disuse. By default, this is 1 minute.

thewall

In the previous versions of AIX, the option **thewall** specified how much of the system's memory could be used for networking. As of AIX 5.2, **thewall** can no longer be modified. It is always set to the highest possible value.

sockthresh

If the amount of network memory in use reaches **thewall**, the system becomes unusable.

The option **sockthresh** has been introduced to prevent this situation. Once network memory utilization reaches the percentage specified by **sockthresh**, no new sockets can be created. Existing sockets can continue to get memory. This allows administrators to access the system in order to lower the network load and remedy the situation.

extendednetstats

The **extendednetstats** option enables the collection of additional networking memory utilization information, which will be displayed by **netstat –m**.

The additional information consists of *by type* memory allocation statistics – it shows for each network memory (mbuf, socket, etc) how many were allocated. This can be useful when debugging memory leaks.

However, turning this option on will cause a performance degradation – for this reason it should be off unless you suspect that you have a network memory issue.

REFERENCES

1 *IBM Certification Study Guide – pSeries AIX System Support*, IBM Corporation, SG24-6185.

2 *AIX 5L Performance Tools Handbook*, SG24-6039-01.

Alex Polak
System Engineer
APS (Israel)

ASPACE Solutions has ported 4TRESS, its multi-channel banking identification and verification solution to AIX running IBM WebSphere Application Server.

Designed specifically for organizations who combine the need to manage a large number of complex relationships with varying levels of associated risk with the requirement to interact over multiple channels, 4TRESS provides users with a single login procedure via the Internet, call centre, face-to-face, or emerging channels such as mobile phones, digital TV, and interactive voice response systems.

4TRESS supports one-time or reusable passwords as well as stronger authentication methods including smartcards, voice verification, two-factor tokens, and digital certificates. A privilege model can control different access rights for individuals or groups of users depending on the channel used.

With 4TRESS, all authentication, authorization and administration activities are written to a secure audit log. Each log entry is time-stamped, digitally signed, and chained to adjacent entries in a secure tamperproof hardware environment to prevent undetected compromise.

For further information contact:
ASPACE Solutions, Three Tuns House, 109 Borough High Street, London, SE1 1NL, UK.
Tel: (020) 7744 6248.
URL: http://www.aspacesolutions.com/news_article_130404.html.

* * *

Mendocino Software has announced RealTime, its data recovery software that will recover applications such as those from PeopleSoft, Oracle, or SAP, and maintain the integrity of their transactions.

Mendocino's host-based software, which runs on AIX and Solaris servers, lets users instantly recover data and continue their operations uninterrupted. It has a GUI with a time-slide on it that lets users roll back transactions to any point in time. All changes to an application are recorded and time-stamped as they are written to disk. If the application fails, the software can rewind it to the closest minute before the failure. This is unlike traditional back-up and recovery, replication, or snapshot back-ups, which are limited by the frequency a customer uses them to back up the network. RealTime restores data by rewinding the previously queued data until it has moved beyond the system crash point.

For further information contact:
Mendocino Software, 47001 Benicia Street, Fremont, CA 94538, USA.
Tel: (510) 668 1600.
URL: http://www.mendocinosoft.com/pages/products.htm.

* * *

Watercloud has reported a vulnerability in AIX 4.3.3 and AIX 5.1, which can be exploited by malicious, local users to perform certain actions on a system with escalated privileges.

A user invoking 'invscoutd' may specify a logfile as a command line argument. This may reportedly be exploited to create or overwrite files with escalated privileges by supplying the path of an arbitrary file.

Their suggested solution is to grant only trusted users access to affected systems, and to make sure that the latest version of invscoutd is used.

For further information contact:
URL: http://www.xfocus.org/exploits/200403/31.html.