121

AIX update

November 2005

In this issue

A tc PUBLICATION

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

Colin Smith
E-mail: info@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.

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.

Enabling remote login without a password

Distributing SSH keys to each server in a large environment can be painful. There are many tools available to help with keeping track of all the keys, but sometimes, for whatever reason, such tools cannot be installed.

SSH offers the ability to log in remotely without a password. In such cases, even more SSH keys need to be kept track of. The following script can be used to enable remote login without a password if the user exists on both the target and the source system.

Other prerequisites:

1. SSH must be installed on both machines.
2. The user who wants to log in remotely without a password needs to run the following command on the source machine:

   ```
   > ssh-keygen -t dsa
   ```

   This command generates a set of keys, all of which are written to the directory *~[user]/.ssh*.
3. The file */etc/ssh/sshd_config* (or the respective configuration file on older SSH installations) on the remote machine needs to contain the following lines:

   ```
   PubkeyAuthentication yes
   AuthorizedKeysFile     .ssh/authorized_keys
   ```
4. The SSH system must be running on the remote host. If the configuration file was changed, it should be restarted:

   ```
   > stopsrc -s sshd
   > startsrc -s sshd
   ```
5. Run the following script to take care of the rest. The script will do the following:

   – create all the necessary directories on the remote machine.

– copy all the key files.

– create the file *authorized_keys* on the target machine.

– try to log in for the first time.

Please note that for security purposes the script will ask you for the remote user password three times.

Even public keys should never be transferred without encryption; thus the script uses scp.

The script also works as user root, although because of security concerns it is not recommended to allow remote root logins. The target system can be an AIX or a Linux machine.

A typical output sequence when using the script would be:

```
[someone@source]/home/someone/script > ./mkssh 172.16.1.80
        Password: [entered password here]
Does not exist. Creating
        Password: [entered password here]
id_dsa.pub
100%  604     3.0MB/s   00:00
id_rsa.pub
100%  225     1.7MB/s   00:00
identity.pub
100%  333     2.5MB/s   00:00
        Password: [entered password here]
Last login: Fri Aug 27 03:19:44 2004 from source.earth.com
someone@target:~>
```

And here is the script mkssh:

```
#!/usr/bin/ksh

# $me is the name of this script
me=${0##*/}
vUser=$(whoami)

function usage
        {
        clear
        cat <<-EOF
        $me tries to configure ssh on the target machine
        for the user ${vUser}. This user can then log in
        remotely without a password.
        $me algol
EOF
```

```
 }

 [[ -z $1 ]] && usage && exit

 vTarget=$1

# Create the necessary directories on the remote machine.
# The next line is quite long, do not insert a line break.
 ssh ${vTarget} "if [ ! -e  ~${vUser}/.ssh/x ]; then echo 'Does not
exist. Creating'; mkdir ~${vUser}/.ssh/; chown ${vUser} ~${vUser}/.ssh;
mkdir ~${vUser}/.ssh/x; else echo 'Does exist'; fi"

# Copy all the key files
 scp ~${vUser}/.ssh/*pub ${vTarget}:~${vUser}/.ssh/x

# Paste the public key files into the file authorized_keys
 ssh ${vTarget} "cat ~${vUser}/.ssh/x/*pub >> ~${vUser}/.ssh/
authorized_keys"

# Try to log in and see whether it works without a password.
# Delete this line if the script should not log in right away.
 ssh ${vTarget}
```

Robert Kaiser
System Analyst
Bayerischer Rundfunk (Germany)

Back-up and restore using a remote tape drive

This is an interesting topic, which has been posted many times on AIX mailing lists and online groups. Many AIX (or other flavours of Unix) administrators may know one way or another to back up files on a system that has no tape or back-up device, or simply they use an NFS mounted filesystem to a server with a tape drive to perform their back-ups.

This article will explain back-up and restore methods from/to tape devices on remote systems, and you can choose to use any of them that suit your needs. You might be forced to use one method rather than another because you already have a back-up tape in a certain format. For example, you prefer to use **backup** as a preferred method of back-up. However, what if you need to remotely restore a **tar** format tape on a system without a tape drive or a compatible tape drive?

All methods in this article, whether for back-up or restore, will require a trusted login on the source and the destination system as a prerequisite for the successful execution of the commands. This means *$HOME/.rhosts* or */etc/hosts.equiv* must be set up for the users performing the back-up or the restore on the source and the remote systems.

BACK-UP TO A REMOTE TAPE DRIVE

Using rdump

The **rdump** utility is an AIX utility that backs up filesystems only to a remote tape drive and needs to be run as a root user. The command format is:

```
rdump -fremote_host:/dev/rmtx /file_system
```

For example:

```
rdump -fmyhost:/dev/rmtØ /usr
```

The **rdump** utility also supports incremental back-ups using the **-Level** switch. For more details about this utility please refer to the command reference manual of the man pages.

Using tar

Tar is a very popular utility among Unix users. The following is an example of how you can combine **tar**, **rsh**, and **dd** to back up to a remote drive:

```
tar -cvBf - . | rsh hostname "dd of=/dev/rmtØ bs=1Ø24"
```

Using backup

The **backup** utility is an AIX utility that backs up files and filesystems using its own format. Other utilities, like **mksysb** and **savevg**, use the **backup** utility internally. The following is an example of how you can combine **backup**, **rsh**, and **dd** to back up to a remote drive:

```
find ./start_point -print | backup -iqvf- | rsh remote_hostname "dd of=/
dev/rmt1 bs=1Ø24"
```

Using cpio

Backing up to a remote drive using **cpio** as a method is an unusual choice. However, if you end up wanting to use it, I found that the following procedure does work:

- Create a FIFO (named pipeline) special file:

  ```
  rm -f /tmp/pipe
  mknod /tmp/pipe p            (or 'mkfifo /tmp/pipe')
  ```

- Run your **cpio** back-up to the FIFO (named pipeline) file that you created. You can put this operation in the background in order not to tie up your session:

  ```
  find . -print | cpio -ovuC512md > /tmp/pipe &
  ```

- Instead of writing to the FIFO file, redirect it to the remote host tape drive:

  ```
  cat /tmp/pipe | rsh host "dd of=/dev/rmtØ bs=1Ø24"
  ```

Using mksysb

This procedure if very helpful in performing your **mksysb** back-up to a remote drive:

- Change the block size on the remote tape drive:

  ```
  rsh host_name "chdev -l rmtØ -a blocksize=512"
  ```

- Create three dummy sectors on the tape to imitate a **mksysb** tape:

  ```
  echo "dummy" | rsh host_name "dd of=/dev/rmtØ.1 bs=512 conv=sync"
  echo "dummy" | rsh host_name "dd of=/dev/rmtØ.1 bs=512 conv=sync"
  echo "dummy" | rsh host_name "dd of=/dev/rmtØ.1 bs=512 conv=sync"
  ```

- Create a FIFO (named pipeline) special file. For the purposes of this example, I chose to use the **mkfifo** command rather than the **mknode** command to show how we can do things in different ways:

```
rm -f /tmp/pipe
mkfifo /tmp/pipe              (or 'mknod /tmp/pipe p')
```

- Perform the **mksysb** back-up to the pipe file as a device. You can run this operation in the background in order not to tie up your session:

```
mkszfile
mksysb /tmp/pipe &
```

- Redirect writing to the FIFO file to the remote host tape drive:

```
cat /tmp/pipe | rsh host_name "dd of=/dev/rmtØ.1 bs=512 conv=sync"
```

- Note that this **mksysb** tape is not bootable because it was written to a file (FIFO special file). Therefore, for recovery, you need to boot from AIX media and then restore from this **mksysb** tape.

RESTORE FROM A REMOTE TAPE DRIVE

Using rrestore

The **rrestore** utility restores a back-up tape that was originally created using the **rdump** utility or the **backup** utility by I-node. However, unlike **rdump**, you can restore the entire filesystem or individual files. The command format would be:

```
rrestore -i -fremote_host:/dev/rmtx [filesystem or files]
```

For example:

```
rrestore -i -fmyhost:/dev/rmtØ
rrestore -i -fmyhost:/dev/rmtØ /usr
rrestore -i -fmyhost:/dev/rmtØ /usr/file1.*
```

For more details about this utility please refer to the command reference manual of the man pages.

Using tar

The following is an example of how you can combine **tar**, **rsh**, and **dd** to restore a **tar** format back-up tape from a remote drive:

```
    rsh hostname "dd if=/dev/rmt0 bs=1024" | tar -xvf -
```

You can restore specific files from the same **tar** tape using **restore**. For example to restore a file called */home/myuser/myfile*:

```
rsh hostname "dd if=/dev/rmt0 bs=1024" | tar -xvf - ./home/myuser/myfile
```

Using restore

The **restore** utility is an AIX utility that restores a **backup** format tape. The following is an example of how you can combine **restore**, **rsh**, and **dd** to restore from a remote drive:

```
    cd ./start_point
    rsh remote_hostname "dd if=/dev/rmt1 bs=1024" | restore -xqvf -
```

To restore a specific file, for example a file called */home/myuser/myfile*, from the same **backup** tape using **restore** you would use:

```
cd ./home
rsh remote_hostname "dd if=/dev/rmt1 bs=1024" | restore -xqvf - ./
myuser/myfile
```

Using cpio

The following shows how to restore a **cpio** format back-up tape from a remote drive:

- Create a FIFO (named pipeline) special file:

```
    rm -f /tmp/pipe
    mknod /tmp/pipe p             (or 'mkfifo /tmp/pipe')
```

- Run your **cpio** restore from the FIFO (named pipeline) file. You can run this operation in the background in order not to tie up your session:

```
    cpio -ivuC512md < /tmp/pipe  &
```

- Redirect writing from the remote host tape drive to the FIFO (named pipeline) that you created:

```
    rsh host "dd if=/dev/rmtØ bs=1Ø24" | dd of=/tmp/pipe bs=1Ø24
```

Restore from mksysb

This procedure shows how to restore specific files from a **mksysb** tape, even from a remote tape drive. **Mksysb** tapes are written in **backup** format on the fourth sector of the tape. Therefore, basically we can use the **restore** utility on the fourth sector of the tape to extract the files:

```
cd ./location
rsh remote_hostname "tctl -fsf 3; dd if=/dev/rmtØ.1 bs=512" | restore -
xqvf -
```

For example, to extract */etc/passwd* file from a **mksysb** tape on a remote tape drive:

```
cd ./location
rsh myhost "tctl -fsf 3; dd if=/dev/rmtØ.1 bs=512" | restore -xqvf - ./
etc/passwd
```

Basim Chafik
Senior Systems Analyst
IBM Certified Advanced Technical Expert (CATE)
Plexus (Division of BancTec) (Canada)

Client FTP automation

FTP (File Transfer Protocol) is the *de facto* standard when transferring files remotely to other hosts, including via the Internet. The basic ftp protocol has been around since the dawn of time and, though typically unsecured in its transfer methods, will be around for many years to come because of its robustness and popularity, FTP runs under TCP/IP. This article will consider the client ftp part, primarily looking at automation. I will not be focusing on the different types of FTP server like wu-ftp or vs-ftp.

USING NETRC

When operating ftp in batch mode or interactively, it is often better to have a mechanism so that one can login automatically to a remote site. Using the special file *.netrc*, which is located in your *$HOME* directory, allows one to do this. The file permissions of this file should be set to read/write for the owner only (chmod 600). The file may contain many entries for remote sites. When ftp is invoked and there is a *.netrc* file present, if the machine name given on the ftp command line is matched from the file, it will use that entry for the login and password. The format of the file comes in two flavours; I prefer the following format:

```
machine <remote_host> login <login> password <password> password
<account_password>
...
```

where it is one host entry per line; *remote_host* is the remote host one is connecting to; *login* is the login name; *password* is the password one is using to connect to the remote host. *account_password* is useful only if there is another authentication process on the remote host that requires an additional password. Typically, though, only the first three entries are required, each field can be space or tab separated.

Looking at how this works, the *.netrc* file below contains three entries. The first entry is to a Web ftp public server. The login of ‘anonymous’ generally means that the file can be uploaded or downloaded from a publicly accessible folder. Though it is considered good form to use the login of ‘anonymous’ and your e-mail address as the password it is not mandatory. The other entries are hosts contained in an internal network.

```
machine ftp.emea.ibm.com login anonymous password
david.tansley@btinternet.com
machine uk01lx6001          login   dxtans        password        lOopy
machine uk04lx6003          login   dxtans        password        mas123
```

To connect to the host uk01lx6001, all that is needed is:

```
$ ftp uk01lx6001
Connected to uk01lx6001
220 uk01lx6001 FTP server (Version 4.1 Wed Mar 26 16:45:44 CST 2003)
```

```
ready.
331 Password required for dxtans.
230-Last unsuccessful login: Tue Jan 18 12:18:34 GMT 2005 on /dev/pts/0
from uk03lx6009
230-Last login: Tue Mar 29 08:40:07 BST 2005 on ftp from uk03lx6009
230 User dxtans logged in.
ftp>
```

As discussed earlier, when ftp is fired up it will look for the *.netrc* file. In my case I am connecting to the host uk01lx6001. The client part will look for a pattern match in that file. In this case we have one – uk01lx6001. It will then use that entry to parse the login/password across to the remote server.

Before delving deeper into how automation can be carried using ftp in batch mode, let's first look at the most common ftp options. They are:

- -p – you are the client so definitely use passive mode for data transfers. This is the preferred use when behind a firewall because the client tells the server which port to use. This is the default now, and is here only for compatibility. However, some older versions still do not use passive as the default.
- -i – this option turns off interactive prompting when negotiating ftp transfers. If you know what files you need to get or put then this option is invaluable.
- -n – when ftp is fired up, one can stop ftp from checking the *.netrc* file for auto-login if auto-login is enabled. If no entry match is found in the *.netrc* file it will then prompt for the login/password.
- -v – let ftp show what is happening, by being very verbose.

Using a couple of these options we can now put together a rather simple ftp script that connects to a remote host and grabs the */etc/hosts* file:

```
#!/bin/bash
# ftp1
ftp -i -v <<mayday
open uk01lx6001
ascii
```

```
lcd /tmp
cd /etc
get hosts
quit
mayday
```

Looking more closely at the script, ftp is invoked with the interactive mode turned off and verbose turned on. Using the 'here document' approach denoted by the << symbol, everything between the first and second occurrence of the word 'mayday' will be taken as read from standard input. First we open a connection to the host uk01lx6001, we inform ftp that we will be using ASCII for transfer, then the directory on the local side is changed to */tmp*, and the remote directory is changed to */etc*. Next, we get (pull) the *hosts* file across, so that it will reside in */tmp*. We then quit ftp. The second occurrence of the word 'mayday' terminates our standard input as far as the script is concerned and thus, because there are no more lines of code, will exit.

Running the above code gives the output shown below:

```
$ ftp1
Connected to uk01lx6001 (168.14.2.4).
220 uk01lx6001 FTP server (Version 4.1 Wed Mar 26 16:45:44 CST 2003)
ready.
331 Password required for dxtans.
230-Last unsuccessful login: Tue Jan 18 12:18:34 GMT 2005 on /dev/pts/0
230-Last login: Tue Mar 29 18:40:33 BST 2005 on ftp from
::ffff:168.14.2.9
230 User dxtans logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
200 Type set to A; form set to N.
Local directory now /tmp
250 CWD command successful.
local: hosts remote: hosts
227 Entering Passive Mode (162,14,2,4,209,161)
150 Opening data connection for hosts (2370 bytes).
226 Transfer complete.
2443 bytes received in 0.00135 secs (1.8e+03 Kbytes/sec)
221 Goodbye.
```

Notice that, by default, ftp will try to do the transfer in binary. By issuing the **ascii** command it is changed:

```
Using binary mode to transfer files.
Type set to A; form set to N
```

Also notice in the output that we are indeed initiating ftp in passive mode.

Using a ‘here document’, one can also parse a list of files to transfer. However, using this method means that one has to connect and terminate for each transfer. The listing below demonstrates how this can be done:

```
#!/bin/bash
# ftp2
list="hosts hosts.allow hosts.deny"
for files in $list
 do
   ftp -i -v <<mayday
   open ukØ1lx6ØØ1
   ascii
   lcd /tmp
    cd /etc
    get $files
    quit
    mayday
done
```

You might think that you could put a ‘for’ loop inside the ftp code block, but you cannot because you will actually be connected to the remote host and thus in ftp mode, not in the bash shell.

It may be the case that one wishes to check for the presence of the *.netrc* file before initiating an ftp session. The listing below contains a simple test block that checks to see whether the file is present and readable by the script. If it is it then it presents the file to the standard output:

```
#!/bin/bash
# ftp4
netrc_file=$HOME/.netrc
if [ -r "$netrc_file" ]
then
 cat $netrc_file | awk '{print $2,$4}'
else
 echo "$netrc_file not present"
fi
```

Why would one want to check whether *.netrc* is present? It may be necessary for a user to decide whether, if *.netrc* is present, to use entries in the file to initiate automatic connection and to transfer files. If the entries are not present in the file, they would be entered manually via a script prompt. However, automatic mode is still desired, so that script could take a back-up of the *.netrc* and then dynamically create a new one and use that for the ftp session. The point I am trying to put over here is, use *.netrc* whenever you can; do not rely on too much interaction for the user's login/password credentials. And, for heaven's sake, do not hard-code a password into an ftp script.

If you want to offer the option of allowing the user to use one of the *.netrc* entries, it would be user friendly to present the contents of the file to the user, so they can select which record entry to use. One way of achieving this is to **cat** the file to standard output using the **-n** option, which will number each line. The listing below presents a framework of how such an option could be achieved:

```
#!/bin/bash
# ftp5
netrc_file=$HOME/.netrc
if [ -r "$netrc_file" ]
then
max_recs='cat $netrc_file | awk 'END{print NR}''
cat -n $netrc_file | awk '{print $1,": connect to [",$3,"] as
user[",$5,"]"}'

 echo -n " Select record to use [ 1 .. $max_recs ] :"
  read ans
if [$ans -ge 1 ] && [ $ans -le $max_recs ]
then
   host='cat $netrc_file | awk "NR==$ans"|awk '{print $2}''
   user='cat $netrc_file | awk "NR==$ans"|awk '{print $4}''
   password='cat $netrc_file | awk "NR==$ans"|awk '{print $6}''
 else
   echo "invalid choice, needs to be [ 1 .. $max_recs ]"
   exit 1
fi     #  $ans in numeric range
 echo -e "selected info is:\nhost [$host]\nuser [$user]\npassword
[$password]"
else
```

```
 echo " Sorry cannot read $netrc_file"
fi     # netrc present
```

Looking more closely, first we check that *.netrc* is readable and thus present. We need to determine how many records are in the file for selection by the user. This is accomplished using the command:

```
max_recs='cat $netrc_file | awk 'END{print NR}''
```

You may be thinking, why not use the **wc -l** command because it requires less processing? Unfortunately, if we use that command it pumps the variable substitution full of spaces and does not cosmetically look good when displaying it to the standard output. By using the **cat** command, all record entries are displayed to standard output, but only the host and login fields of each record are shown. The listing below shows the output of this code being executed:

```
$ ftp5
1 : connect to [ ftp.emea.ibm.com ] as user[ anonymous ]
2 : connect to [ ukØ1lx6ØØ1 ] as user[ dxtans ]
3 : connect to  [ukØ4lx6ØØ3  ] as user[ dxtans]

Select record to use [ 1 .. 3 ] :2
selected info is:
host [ukØ1lx6ØØ1]
user [dxtans]
password [lOopy]
```

We next prompt the user to enter a number that represents the required record they wish to use, using the $max_recs variable, which holds the total number of records and directs the user to the allowable limit of choices. Once selected, the input number is used to extract the required fields from the record using **awk**'s NR function. The results are then echoed to standard output.

The code listed above demonstrates one way to use a menu-driven interface. I have not included any serious error checking code, apart from a numerical range check, because I want to demonstrate that it does not take too much code with some simple **awk** one-liners to put together a framework that can be menu driven.

Another option, if one does not wish to use the *.netrc* file, is to get the script to source the login/password from an environment file. The listing below contains the necessary authentication information:

```
login=dxtans
password=lOopy
host=ukØ1lx6ØØ1
```

An ftp script could source this file and use that to automatically initiate ftp and file transfer. This could be done by:

```
#!/bin/bash
# ftp5
# source the file
. .ftp_netrc
for files in $list
do
 ftp -i -v  <<mayday
 open $host
 $login
 $password
 ascii
 lcd /tmp
 cd /etc
 get $files
 quit
 mayday
done
```

The source file is sourced, so we now have the variables $host, $login, and $password accessible to the script. The script can now use that information to automate the ftp process. However, I much prefer the *.netrc* method and so should you, because you are not sourcing the password that may be accessible from another session that is looking at the processes running.

CHECKING ERRORS

When an ftp script is running, or, indeed, any script, you will want to check for errors at the end of the run. Quite a few error codes are provided by ftp. A few common return codes that may be considered errors or warnings are shown below:

- 421 – service not available
- 426 – transfer aborted
- 450 – file unavailable
- 500 – syntax error
- 501 – syntax error in arguments
- 503 – user not logged in
- 550 – file unavailable
- 553 – illegal filename
- 666 – file or directory does not exist
- 777 – unknown host
- 999 – invalid command.

One way to check for errors is to **egrep** at the end of a script run. The listing below demonstrates one way to achieve this:

```
#!/bin/bash
# ftp7
log=ftp.log
>$log

list="hosts telnet.conf"
host="uk01lx6001"

echo "Script name [ `basename $0` ]" >>$log
for files in $list
do
 ftp -i -v  >> $log 2>&1 <<mayday
 open $host
 ascii
 lcd /tmp
 cd /etc
 get $files
 quit
 mayday
done
 if egrep "202|421|426|450|500|501|503|550|553|666|777|999"  \
$log > /dev/null 2>&1
  then
   echo "Errors" | tee -a $log
```

```
    exit 1
  else
   echo "OK" | tee -a $log
   exit 0
fi
```

The first thing we need to do is redirect all ftp output into a log file. The following piece of code achieves this:

```
ftp -i -v  >> $log 2>&1 <<mayday
```

It will redirect all output including errors from the ftp session into a log file whose value is held in the variable $log. Once the ftp session has finished we simply use **egrep** to include a list of codes or words we wish to match, with each pattern separated by the bar '|' sign. When using any **grep** within a script it is always a good idea to redirect the output of the pattern match to */dev/null*. It keeps unwanted messages from cluttering the standard output. Like so:

```
if egrep "202|421|426" $log > /dev/null 2>&1
```

If **egrep** returns true with any match then we simply exit with a 1 and echo a message to standard output and the log file. Otherwise we exit OK, with a 0 status. If scripts are run in batch and not interactively from the command line, one should omit echoing anything out to the standard output and should go directly to a log file. If this is the case then we could amend the exit code to:

```
    echo "Errors" >> $log
```

Likewise for a good exit message:

```
    echo "OK" >>$log
```

CONCLUSION

Using ftp interactively is a productive way to transfer files between hosts. Do not be afraid of creating different ftp scripts to fill different administration requirements. However, I do strongly advise using the *.netrc* file within your scripts. So much information about ftp is usually about the different type of ftp servers. In this article I have purposely kept away from that and just dealt with the client side.

David Tansley
Global Operations
ACE Overseas General (UK)

Successful business continuity: volume assignment

This is the third in a series of articles discussing how to implement AIX in an environment dedicated to business continuity (see *AIX Update*, issues 114 and 115, April and May 2005). The topic of this article is the assignment of volume groups, physical volumes, logical volumes, JFS log logical volumes, filesystem mount points, resource groups, and major numbers. The name assignments of each of these entities should be unique, enterprise-wide, and conducive to high-availability and disaster recovery. Much of the content of this article assumes that a SAN environment is available to the AIX systems. In a SAN environment, enterprise-wide unique names allow the AIX system administrator to move a resource group between AIX systems, between clusters, or even between data centres without conflict. The point of ensuring that all resources are addressed by enterprise-wide unique values is to eliminate the conflicts that must be resolved during a high-availability failover or a disaster recovery implementation.

Definition: enterprise-wide unique refers to a parameter that has one distinct value across any or all platforms throughout the entire enterprise.

The purpose of this series of articles is to provide a foundation for business continuity. In support of that purpose, each topic discussed in this article is divided into the following:

- Policies
- Guidelines
- Standards
- Procedures.

Each organization should define its own set of policies, guidelines, standards, and procedures that define their enterprise-wide rules of design and implementation. These rules ensure the ability of an organization to operate on a day-to-day basis as well as in a disaster recovery effort.

On an AIX system the Logical Volume Manager (LVM) provides the system administrator with the ability to allocate storage space for various purposes. The default volume group created when the AIX operating system is installed is called rootvg. In order to facilitate high-availability failovers and disaster recovery, the rootvg volume group is used by many organizations only for the storage of operating system-specific files, programs, etc. All other volume groups are created on a SAN environment for the storage of applications and data.

Many organizations configure their AIX systems with two internal hard drives, both drives being allocated for storage of the operating system in the rootvg volume group, the second drive being a mirror of the first.

An extension of this technique for the creation of boot device(s) is to install AIX on one internal hard drive, mirror the boot disk to a second internal hard drive, and create a second mirror on a SAN disk. Then change the boot list to boot from the SAN disk first, and the internal disks second and third.

This (recommended) technique requires that the system administrator install the operating system onto a single internal disk, mirror the boot disk onto a second internal disk and onto a SAN disk. Example commands are shown where *hdisk0* is the boot disk on which AIX is initially installed, *hdisk1* is the second internal disk, and *hdisk2* is the SAN disk.

```
# mirrorvg rootvg hdisk1
# mirrorvg rootvg hdisk2
# bosboot -ad /dev/hdisk1
# bosboot -ad /dev/hdisk2
# bootlist -m -o hdisk2 hdiskØ hdisk1
```

The result of this technique is a primary boot disk on the SAN with two mirrors of the boot disk stored internally. One of the benefits of this technique is the rootvg can be booted from any AIX system that can access the SAN environment. In the event of a machine failure, its rootvg volume group can simply be booted from another AIX system.

When upgrading AIX to a newer OS level, one of the rootvg mirrored disks can be used to accept the upgrade. This technique, called upgrading an alternate clone, is outside the scope of this discussion, but is well documented on the IBM Web site. Utilizing this technique allows the system administrator to perform a system upgrade to an alternate boot disk while the system is up and running. Once the upgrade is complete the system is rebooted from the alternate boot disk, thus reducing the downtime to the time it takes to perform a single reboot. An added benefit of this technique is that if the upgrade results in an unstable system, it can be rebooted from the original, unchanged boot disk.

In support of a business continuity environment, all application programs, files, and data should be located in non-rootvg volume groups, on disks in the SAN environment. Maintenance, back-ups, failovers, system recovery, and disaster recovery procedures will be made easier, predictable, and repeatable by adhering to this rule.

In a previous article of this series, the concept of resource group was used to define any logical collection of resources, which may include disk, I/O, users, applications, etc. A resource group should be viewed as independent of any machine, server, or data centre. In this context, the resource group name is used as the basis of all other naming structures for all entities whether or not they are controlled by HACMP. The resource group name should be an enterprise-wide unique value in order to eliminate conflicts during manual, automated, or disaster recovery failovers.

Resources, such as disks, volume groups, NFS mounts, etc, needed to support applications should be allocated and provided with a resource group name. These names will be used as the basis for the volume group names. The previous article in this series discussed the creation of resource group names and described a simple technique for defining the resource group names. A more complex technique will be described here for the purpose of integrating the resource group name into the naming structure of volume groups, logical volumes, directory mount points, and other entities to be discussed later.

The resource group naming structure described here is an actual implementation in a service provider's data centre environment, where AIX systems for multiple companies may be simultaneously housed on a single piece of hardware separated by LPARs.

The supporting principles of business continuity provide the basis for defining the standardized naming structures mentioned thus far. Those principles are:

- Policies – those things that shall be done.
- Guidelines – those things that should be done or can be expected.
- Standards – the structure, based on policies and guidelines.
- Procedures – how to implement the structure.

The following policies, guidelines, standards, and procedures describe the rules imposed in an actual data centre for building, supporting, and maintaining a large multi-company, multi-data centre, AIX environment.

POLICIES

All system resources shall be divided into logical resource groups, and each resource group shall have an enterprise-wide unique name. The resource group name shall be the basis of numerous naming structures and policies.

Resource groups shall be defined for stand-alone and high-availability systems. All AIX systems will be designed as though they participate in a high-availability environment, regardless of whether or not they actually do. This does not mean that all AIX systems will have HACMP or other automated high-availability software installed.

A centralized repository (database) shall contain a list of all configured resource groups enterprise-wide. New resource groups shall be entered into this repository.

The boot disk for each AIX system should be located on the SAN environment. Redundant mirrors may be located internally in each machine.

All data and applications will be stored separately from the rootvg volume group. Additional volume groups will be configured for storage of data and applications as necessary.

Each volume group shall have an enterprise-wide unique name for the purpose of eliminating naming conflicts during manual, HACMP, or disaster recovery failovers.

Each volume group volume shall have a cluster-wide unique major number for the purpose of eliminating numbering conflicts during manual, HACMP, or disaster recovery failovers. If possible, make the major number unique enterprise-wide.

Each logical volume shall have an enterprise-wide unique name for the purpose of eliminating naming conflicts during manual, HACMP, or disaster recovery failovers.

Each volume group that contains filesystems will require at least one JFS log. Each JFS log logical volume shall have an enterprise-wide unique name for the purpose of eliminating naming conflicts during manual, HACMP, or disaster recovery failovers.

Each filesystem shall have an enterprise-wide unique mount point directory for the purpose of eliminating directory conflicts during manual, HACMP, or disaster recovery failovers.

GUIDELINES

Each AIX system will host one or more resource group.

There should be a unique DNS entry based on each resource group name. These names shall be aliases pointing to IP names that are assigned to IP addresses.

RG name component	*Number of characters*	*Values*

Application code	3	atl = Atlas
		db2 = DB2
		nim = NIM
		ora = Oracle
		peo = PeopleSoft
		sap = SAP
		tps = Maximo
		vio = Virtual I/O

Environment	1	a = acceptance
		g = pre-production/Gold
		d = test/development
		p = production
		t = test
		x = disaster recovery

Function	1	a = application
		c = combination/multi-purpose
		d = database
		m = management
		u = utility

Company or other identifier	2	ac = Acme
		mx = Mt Xia
		ib = IBM

Sequence ID	1	0-9,A-Z,a-z

Figure 1: Resource group name components

Application volume group, logical volume, JFS log logical volumes, and filesystem mount point names should be enterprise-wide unique values based on the resource group name.

Where possible, create volume groups with an enterprise-wide unique major number. At a bare minimum, the major number should be cluster-wide unique.

More than one JFS log may be configured for a volume group if necessary, dependent upon the filesystem load.

In order to ensure the successful installation of an application into unique mount points, the application team must work closely with the system engineer.

STANDARDS: RESOURCE GROUP NAME

A single standard shall be used for stand-alone, high availability, and disaster recovery environments. This will eliminate naming conflicts in the event of a manual or automated failover, or if multiple instances of an application are running on a single AIX system.

The concept of Resource Group (RG) is used here in a larger scope than it is used in HACMP. In an enterprise environment, a resource group is any logical collection of resources. This may include disk, I/O, users, applications, etc. A resource group should be viewed as being independent of any AIX system or data centre. The resource group name is used as the basis of all other naming structures for all entities whether or not they are controlled by HACMP. The resource group name shall be an enterprise-wide unique value in order to eliminate conflicts during manual, automated, or disaster recovery failovers.

When designing any new system, the first step is to determine the resource group name(s). The names of volume groups, logical volumes, mount points, major numbers, WLM classes, etc, are all derived from the resource group name(s).

The resource group name shall consist of exactly eight characters, with the following structure:

```
ApplicationCode + Environment + Function + Custom + Sequence ID
     3 char     +    1 char   +  1 char  + 2 char +   1 char
```

The detailed information for each component of the resource group name is described in Figure 1.

A single AIX system may contain multiple resource groups, and typically there will be one volume group defined per resource group. However, a resource group may contain several volume groups, depending on the requirements of the application.

RG name component	*VG sequence identifier*	*LV identifier*	*VG name*
db2apmx0	00	vg	db2apmx000vg
db2apmx0	01	vg	db2apmx000vg
db2apmx0	02	vg	db2apmx002vg

Figure 2: Resource group with multiple associated VGs

STANDARDS: VOLUME GROUP NAME

To assign enterprise-wide unique volume group (VG) names, the system administrator must first define the resource group names. Once the resource group names have been defined, then a VG name may be defined based on the resource group name.

In order to facilitate normal maintenance, disaster recovery, and business continuity, it is recommended that each enterprise-wide unique volume group name be assigned an enterprise-wide unique major number.

A single AIX system may contain multiple resource groups, and typically there will be one VG defined per resource group.

However, a resource group may contain several VGs, depending on the requirements of the application.

To define a VG name, obtain the eight-character resource group name. Then add a two-digit volume group sequence number that will uniquely identify the VG, followed by the characters ‘vg’. The VG name will always end with the characters ‘vg’.

The VG name shall consist of exactly 12 characters, with the following structure:

```
ApplicationCode + Environment + Function + Company + Sequence ID
    3 char      + 1 char      + 1 char   + 2 char  +  1 char
+ VG Sequence ID + "vg"
+   2 char       + 2 char
```

Figure 2 shows an example of a resource group named db2apmx0, having multiple associated VGs.

Each VG also requires a server or cluster-wide unique major number. A unique major number can be generated using the following algorithm:

```
VGNAME="db2apmx000vg"
MajorNbr=$( print "${VGNAME}" | sum -o | awk '{ print $1 }' )
print ${MajorNbr}
```

To reiterate, before creating a VG, first establish an enterprise-wide unique resource group name, a VG name, and a major number. Then create the VG.

RG name component	*LV sequence identifier*	*LV identifier*	*LV name*
db2apmx0	db20	lv	db2apmx0db20lv
db2apmx0	db21	lv	db2apmx0db21lv
db2apmx0	db22	lv	db2apmx0db22lv

Figure 3: Volume group and LVs

STANDARDS: LOGICAL VOLUME NAME

To assign enterprise-wide unique logical volume (LV) names, the system administrator must first define the resource group names. Once the resource group names have been defined, a VG must be defined based on the RG name. After the VG has been created, LVs can be assigned. A VG will typically contain several LVs, and each LV will be named based on the resource group to which it is associated.

To define an LV name, obtain the eight-character resource group name, then add a four-digit logical volume sequence identifier that will uniquely identify the LV, followed by the characters 'lv'. The four-digit LV sequence identifier will consist of alpha-numeric characters and must always be exactly four characters in length. The LV name will always end with the characters 'lv'.

The LV name shall consist of exactly 14 characters, with the following structure:

```
ApplicationCode + Environment + Function + Company + Sequence ID
   3 char       + 1 char      + 1 char   + 2 char  + 1 char
+ LV Sequence ID + "lv"
+  4 char         + 2 char
```

As an example, a resource group named db2apmx0 may have a volume group named db2apmx00vg. This volume group may have multiple LVs associated with it; see Figure 3.

JFS filesystems will require a logical volume for the JFS log. This must also have an enterprise-wide unique name.

RG name component	*JFS log LV sequence ID*	*JFS log LV ID*	*JFS log LV name*
db2apmx0	jfs0	lv	db2apmx0jfs0lv
db2apmx0	jfs1	lv	db2apmx0jfs1lv
db2apmx0	jfs2	lv	db2apmx0jfs2lv

Figure 4: VG with multiple JFS Log LVs

STANDARDS: JFS LOG LOGICAL VOLUME NAME

To assign enterprise-wide unique JFS log LV names, the system administrator must first define the resource group names. Once the resource group names have been defined, a VG must be defined based on the RG name. After the VG has been created, JFS log LVs can be assigned. A VG will typically contain one JFS log LV; however, multiple JFS log LVs can exist.

To define a JFS Log LV name, obtain the eight-character resource group name, then add the four-digit logical volume sequence identifier that will uniquely identify the JFS log LV, followed by the characters ‘lv’. The four-digit JFS log LV sequence identifier will consist of the characters ‘jfs’ followed by a single digit to uniquely identify the JFS log LV. The JFS Log LV name will always end with the characters ‘lv’.

The JFS log LV name shall consist of exactly 14 characters, with the following structure:

RG name component	*Optional LV sequence ID*	*Optional subdirectories*	*Filesystem mount point*
db2apmx0		db2_08_01/bin	/db2apmx0/db2_08_01/bin
db2apmx0		db2_08_01/data	/db2apmx0/db2_08_01/data
db2apmx1	mq01	lv	/db2apmx1mq01
db2apmx1	mq02	lv	/db2apmx1mq02
db2apmx1	mq03	lv	/db2apmx1mq03

Figure 5: Resource group name and filesystems

```
ApplicationCode + Environment + Function + Company + Sequence ID
   3 char       +  1 char     + 1 char   + 2 char  + 1 char
+  "jfs"  + JFS Log Sequence ID + "lv"
+ 3 char  +   1 char            + 2 char
```

As an example, a resource group named db2apmx0 may have a volume group named db2apmx00vg. This volume group may have multiple JFS Log LVs associated with it, as shown in Figure 4.

JFS filesystems will require a logical volume for the JFS log. This must also have an enterprise wide unique name.

STANDARDS: FILESYSTEM DIRECTORY MOUNT POINTS

To assign enterprise-wide unique LV names, the system administrator must first define the resource groups, volume groups, and logical volumes. Once these have been defined, the filesystem mount point directory names can be assigned. Typically a filesystem mount point is required for each logical volume, therefore the mount point can usually be based on the logical volume name, or, at a minimum, the resource group name.

To define a filesystem mount point directory name, obtain the eight-character resource group name, then, depending on the application's filesystem requirements, use the RG name as the mount point, or add subdirectories to make it enterprise-wide unique.

The filesystem mount point directory name shall consist of at least eight characters, but may be of a variable length:

```
/ + ApplicationCode + Environment + Function + Company + Sequence ID
   3 char           + 1 char      + 1 char   + 2 char  + 1 char
+ ( LV Sequence ID or Directory Structure )
+   4 or more characters
```

As an example, a resource group named db2apmx0 may have multiple filesystems associated with it; see Figure 5.

PROCEDURES

The procedures associated with implementing the standards described here require strict attention to detail when creating volume groups, logical volumes, and filesystems. The system administrator must plan the implementation and define a resource group to associate with the naming structures. For example, the steps in defining a new filesystem are as follows:

1 Define an enterprise-wide unique resource group name.

2 Define an enterprise-wide unique volume group name.

3 Define a cluster-wide unique (enterprise-wide unique if possible) major number for the new volume group.

4 Define an enterprise-wide unique JFS log logical volume name.

5 Define an enterprise-wide unique logical volume name.

6 Define an enterprise-wide unique mount point directory name.

7 Create the volume group using the enterprise-wide unique name and major number.

8 Create the JFS log logical volume using the enterprise-wide unique name.

9 Create the logical volume using the enterprise-wide unique name.

10 Create the filesystem on the logical volume using the enterprise-wide unique mount point directory and JFS log.

System design and implementation in a business continuity environment should be regarded as a ‘non-standard’ design. Normally a ‘standard’ implementation does not account for the ability to perform failovers from any AIX system to any other AIX system, as described here. When installing applications in this ‘business continuity’ environment, recognize that the application vendor will almost always provide ‘standard’ installation instructions, not ‘business continuity’ installation instructions. For the system and application administrators, there is a learning curve associated with moving from a ‘standard’ environment to a ‘business continuity’ environment because the ‘standard’ installation instructions for applications must usually be modified to fit this enhanced environment.

The procedures for each application to be installed in the 'business continuity' environment will be specific to each environment and will likely be different from the documentation provided by the application vendor. It will be the job of the system and application administrators to create a procedure to fit the environment.

The next article in this series will discuss enterprise-level techniques for defining resource group start/stop scripts, MQ Series queue names and aliases, error logging, and error notification.

Dana French
President
Mt Xia (tel: 615-556-0456) (USA)

AIX and DB2 tuning for DB2 performance – a case study

The essence of knowledge is, having it, to apply it – Confucius

OVERVIEW

In a previous article ('AIX and DB2 Tuning Essentials and Best Practices for DB2 Performance' in *AIX Update*, issue 120, October 2005) we talked about performance issues and methods of tuning in AIX and DB2. We discussed the important configuration parameters and registry variables to improve CPU, memory, and I/O performance. We discussed best practices for creating buffer pools, table spaces, tables, and indexes, etc.

In this article we'll cover tuning based on AIX tools and monitor output in detail. I'll show you types of monitor switches and how to use snapshot monitoring. We will discuss in detail a case study for a system (P Series 610 server with AIX5L 5.2

and DB2 UDB V8.2 ESE) having high CPU utilization, high memory usage, and high I/O wait times. We will discuss various AIX commands to understand the system bottleneck and how to tune various DB2 configuration parameters and registry variables to overcome the problem. The case study allows you to better understand the relationship between the tools, their output, and problems related to CPU, memory and I/O.

The article is intended for those with an intermediate skill in AIX and DB2 database administration.

MONITOR SNAPSHOTS

What are snapshots?

DB2 snapshots are point-in-time views into what DB2 is doing and how it is performing. They are used in the performance tuning of instances, databases, and applications running on DB2. The information returned from a snapshot is a combination of cumulative information and data that applies only to a single moment. So why should I use them? Snapshots are the only method available to view a large amount of the behaviour of DB2. There are tools built into V8.1 that allow you to track the memory usage of the instances and databases but these tools often do not give sufficiently detailed information. Snapshots are actually used by all the third-party analysis tools that run on DB2. The tools are simply taking snapshots constantly and parsing the data into an easily viewed format.

What types of snapshot are there?

Each type of snapshot captures a different set of data about DB2:

- Database
- Table space
- Tables

- Buffer pools
- Locks
- Database manager
- Applications.

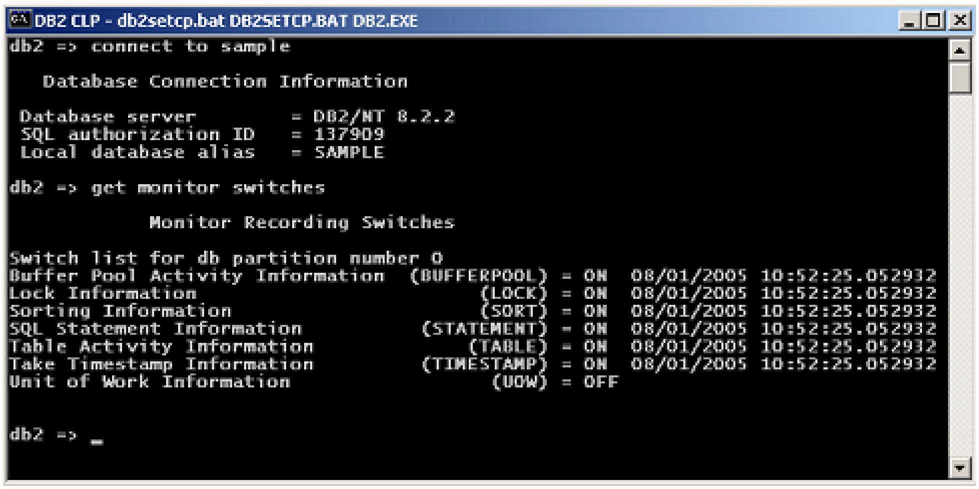
Turning snapshots on and off

The monitoring of DB2 is not turned on by default, but instead has to be set at the connection or instance level. There is a series of monitor switches that determine whether a type of data element is monitored. There is also a memory heap set aside for containing the information stored for monitoring.

Method 1: Setting monitor switches at the instance level

Setting the monitor switches at the instance level will affect all users connected to any databases in that instance. Here are the monitor switches:

- DFT_MON_STMT – statement monitor (used for dynamic SQL)
- DFT_MON_TABLE – table monitors



```
db2 => connect to sample

   Database Connection Information

 Database server        = DB2/NT 8.2.2
 SQL authorization ID   = 137909
 Local database alias   = SAMPLE

db2 => get monitor switches

            Monitor Recording Switches

Switch list for db partition number 0
Buffer Pool Activity Information  (BUFFERPOOL) = ON  08/01/2005 10:52:25.052932
Lock Information                        (LOCK) = ON  08/01/2005 10:52:25.052932
Sorting Information                     (SORT) = ON  08/01/2005 10:52:25.052932
SQL Statement Information          (STATEMENT) = ON  08/01/2005 10:52:25.052932
Table Activity Information             (TABLE) = ON  08/01/2005 10:52:25.052932
Take Timestamp Information         (TIMESTAMP) = ON  08/01/2005 10:52:25.052932
Unit of Work Information                 (UOW) = OFF


db2 => _
```

Figure 1: Database manager configuration parameters

- DFT_MON_LOCK – lock monitor
- DFT_MON_BUFPOOL – buffer pool monitors
- DFT_MON_SORT –sort monitoring
- DFT_MON_UOW – unit of work information
- DFT_MON_TIMESTAMP – tracks timestamp information.

The values of these switches are stored in the database manager configuration information. Figure 1 shows the database manager configuration parameters.

The monitor switches are set just like any other instance configuration parameter:

```
db2 update DBM cfg using monitorSwitch [ON | OFF]
```

Here’s an example:

```
db2 update DBM cfg using DFT_MON_SORT ON
```

Method 2: Setting the monitor switches at an application level

The monitor switches can also be set at an application level; setting the switches in this way will apply to only a particular application. If you use the method listed below at a command prompt or command window, the changes will be applicable to only that particular prompt window.

Here are the switch names:

- Bufferpool
- Lock
- Sort
- Statement
- Table
- Timestamp
- UOW.

Use this command to list the switches:

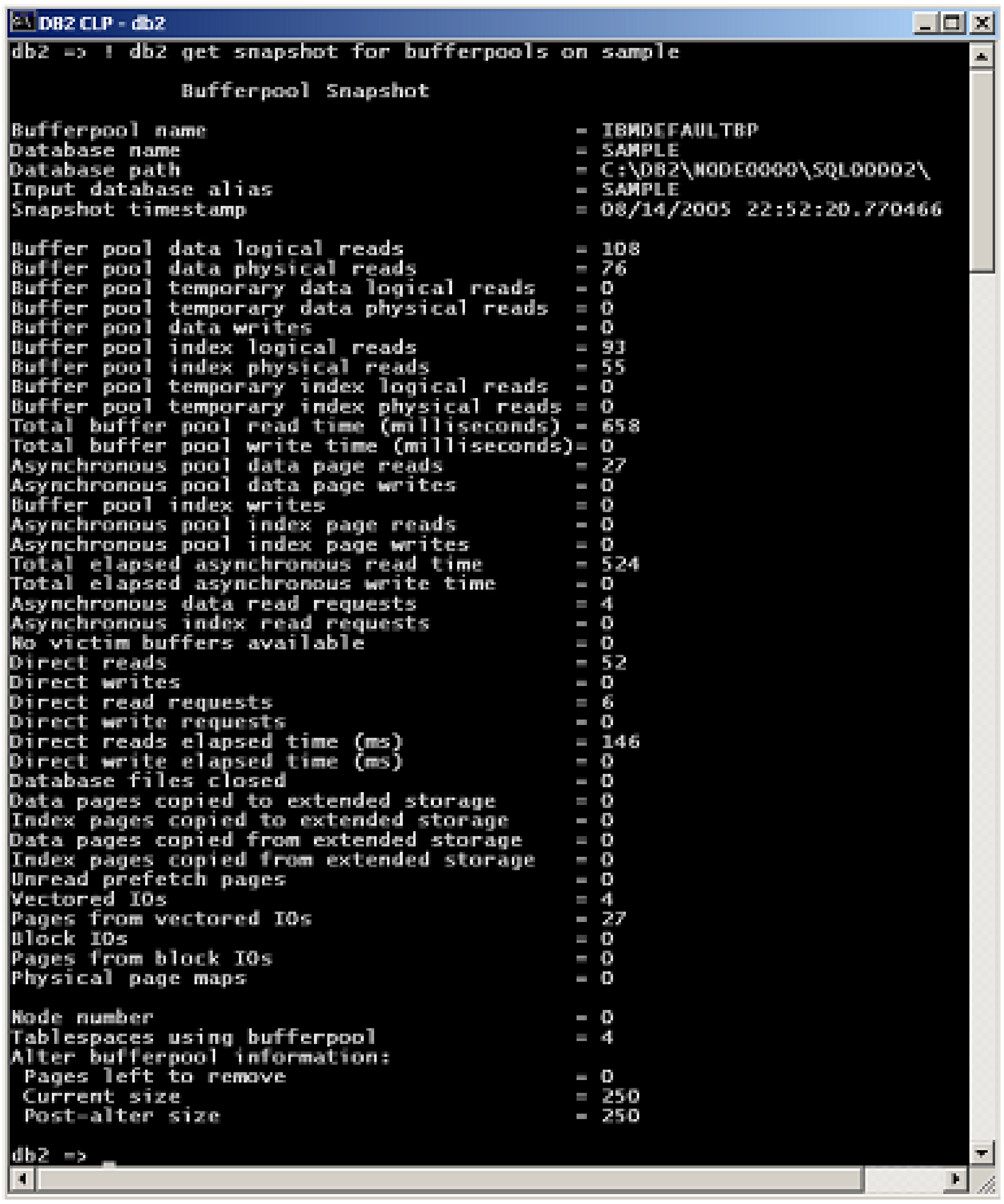
```
db2 get monitor switches
```

And use this command to set the switches:

```
db2 update monitor switches using switchName [ON | OFF]
```

Scope of the snapshots

It is important to understand what information is being returned



```
db2 => ! db2 get snapshot for bufferpools on sample

             Bufferpool Snapshot

Bufferpool name                                = IBMDEFAULTBP
Database name                                  = SAMPLE
Database path                                  = C:\DB2\NODE0000\SQL00002\
Input database alias                           = SAMPLE
Snapshot timestamp                             = 08/14/2005 22:52:20.770466

Buffer pool data logical reads                 = 108
Buffer pool data physical reads                = 76
Buffer pool temporary data logical reads       = 0
Buffer pool temporary data physical reads      = 0
Buffer pool data writes                        = 0
Buffer pool index logical reads                = 93
Buffer pool index physical reads               = 55
Buffer pool temporary index logical reads      = 0
Buffer pool temporary index physical reads     = 0
Total buffer pool read time (milliseconds)     = 658
Total buffer pool write time (milliseconds)    = 0
Asynchronous pool data page reads              = 27
Asynchronous pool data page writes             = 0
Buffer pool index writes                       = 0
Asynchronous pool index page reads             = 0
Asynchronous pool index page writes            = 0
Total elapsed asynchronous read time           = 524
Total elapsed asynchronous write time          = 0
Asynchronous data read requests                = 4
Asynchronous index read requests               = 0
No victim buffers available                    = 0
Direct reads                                   = 52
Direct writes                                  = 0
Direct read requests                           = 6
Direct write requests                          = 0
Direct reads elapsed time (ms)                 = 146
Direct write elapsed time (ms)                 = 0
Database files closed                          = 0
Data pages copied to extended storage          = 0
Index pages copied to extended storage         = 0
Data pages copied from extended storage        = 0
Index pages copied from extended storage       = 0
Unread prefetch pages                          = 0
Vectored IOs                                   = 4
Pages from vectored IOs                        = 27
Block IOs                                      = 0
Pages from block IOs                           = 0
Physical page maps                             = 0

Node number                                    = 0
Tablespaces using bufferpool                   = 4
Alter bufferpool information:
 Pages left to remove                          = 0
 Current size                                  = 250
 Post-alter size                               = 250

db2 => _
```

Figure 2: Generating a snapshot for each buffer pool

to you with the snapshots and when the monitoring begins. The switches can be set dynamically at both instance and application levels, and both affect the monitoring of connections.

If the monitor switches are turned on after an application has already connected, the information will be captured only for all actions after the switch has been turned on.

Resetting the switches

The monitor switches can be reset to null or 0 for all values by executing the following command:

```
db2 reset monitor [ALL | for database databaseName] [at dbpartitionnum
partitionNum]
```

where *databaseName* is the name of the database and *partitionNum* is the node number.

Here's an example:

```
db2 reset monitor for database drew_db
```

Use snapshot monitoring to identify the behaviour of a database over a period of time, showing things such as how memory is being utilized and how locks are being acquired. Monitoring is the approach used to fine-tune configuration and identify problems such as long statement execution time.

Using **get snapshot for bufferpools on dbname** will generate one snapshot for each buffer pool on the database. Figure 2 shows such a snapshot.

CASE STUDY

An example explains a thousand concepts – Anonymous

The following case study allows you to better understand the relationship between the tools, their output, and problems related to CPU, memory, and I/O that you may need to solve.

The performance assessment of the environment has been conducted in the following dimensions:

- CPU performance
- I/O performance
- Memory performance.

Machine configuration:

- IBM P690
- 2 x PowerPC Power4 1300MHz processor
- 2GB memory
- L2 cache 1440KB
- AIX5L 5.2 ML02 64-bit operating system.

Storage:

- FAStT900 – 20 disks (5 x 4+p RAID arrays).

DB2 configuration:

- DB2 V8.1.1.64 ESE, FixPak 7
- Workload – OLAP, 20GB, 18 queries
- Partitions – 2 logical nodes.

CPU PERFORMANCE

First we discuss the CPU bound performance problem with conclusions based on output from commands discussed in a previous article.

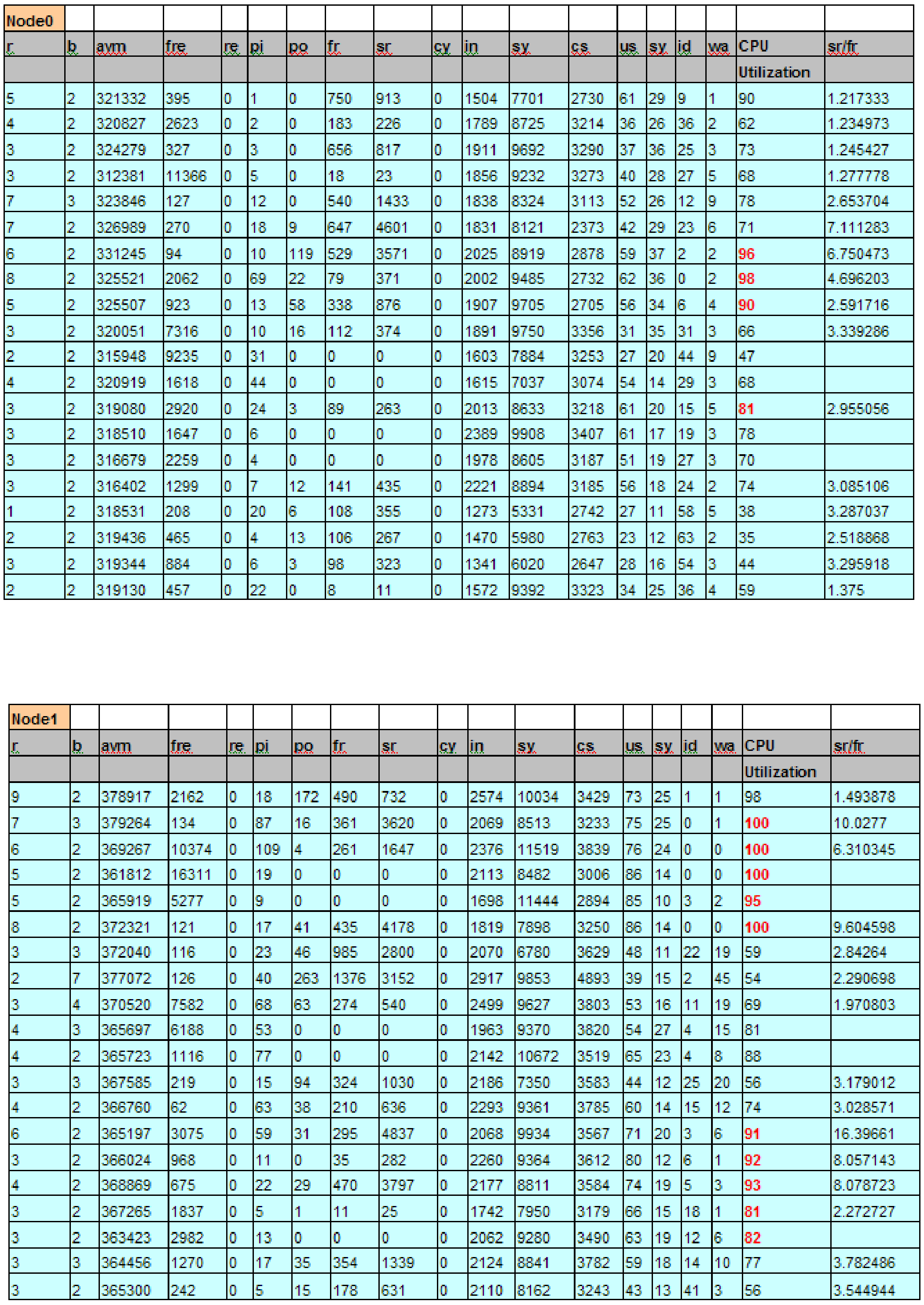
Data collection

The **vmstat** output is shown below for node0 and node1:

```
# vmstat 2 40
```

Or:

```
# rsh <Node Name> 'vmstat 1 10'
```



Node0																		
r	b	avm	fre	re	pi	po	fr	sr	cy	in	sy	cs	us	sy	id	wa	CPU	sr/fr
																	Utilization	
5	2	321332	395	0	1	0	750	913	0	1504	7701	2730	61	29	9	1	90	1.217333
4	2	320827	2623	0	2	0	183	226	0	1789	8725	3214	36	26	36	2	62	1.234973
3	2	324279	327	0	3	0	656	817	0	1911	9692	3290	37	36	25	3	73	1.245427
3	2	312381	11366	0	5	0	18	23	0	1856	9232	3273	40	28	27	5	68	1.277778
7	3	323846	127	0	12	0	540	1433	0	1838	8324	3113	52	26	12	9	78	2.653704
7	2	326989	270	0	18	9	647	4601	0	1831	8121	2373	42	29	23	6	71	7.111283
6	2	331245	94	0	10	119	529	3571	0	2025	8919	2878	59	37	2	2	96	6.750473
8	2	325521	2062	0	69	22	79	371	0	2002	9485	2732	62	36	0	2	98	4.696203
5	2	325507	923	0	13	58	338	876	0	1907	9705	2705	56	34	6	4	90	2.591716
3	2	320051	7316	0	10	16	112	374	0	1891	9750	3356	31	35	31	3	66	3.339286
2	2	315948	9235	0	31	0	0	0	0	1603	7884	3253	27	20	44	9	47	
4	2	320919	1618	0	44	0	0	0	0	1615	7037	3074	54	14	29	3	68	
3	2	319080	2920	0	24	3	89	263	0	2013	8633	3218	61	20	15	5	81	2.955056
3	2	318510	1647	0	6	0	0	0	0	2389	9908	3407	61	17	19	3	78	
3	2	316679	2259	0	4	0	0	0	0	1978	8605	3187	51	19	27	3	70	
3	2	316402	1299	0	7	12	141	435	0	2221	8894	3185	56	18	24	2	74	3.085106
1	2	318531	208	0	20	6	108	355	0	1273	5331	2742	27	11	58	5	38	3.287037
2	2	319436	465	0	4	13	106	267	0	1470	5980	2763	23	12	63	2	35	2.518868
3	2	319344	884	0	6	3	98	323	0	1341	6020	2647	28	16	54	3	44	3.295918
2	2	319130	457	0	22	0	8	11	0	1572	9392	3323	34	25	36	4	59	1.375

Node1																		
r	b	avm	fre	re	pi	po	fr	sr	cy	in	sy	cs	us	sy	id	wa	CPU	sr/fr
																	Utilization	
9	2	378917	2162	0	18	172	490	732	0	2574	10034	3429	73	25	1	1	98	1.493878
7	3	379264	134	0	87	16	361	3620	0	2069	8513	3233	75	25	0	1	100	10.0277
6	2	369267	10374	0	109	4	261	1647	0	2376	11519	3839	76	24	0	0	100	6.310345
5	2	361812	16311	0	19	0	0	0	0	2113	8482	3006	86	14	0	0	100	
5	2	365919	5277	0	9	0	0	0	0	1698	11444	2894	85	10	3	2	95	
8	2	372321	121	0	17	41	435	4178	0	1819	7898	3250	86	14	0	0	100	9.604598
3	3	372040	116	0	23	46	985	2800	0	2070	6780	3629	48	11	22	19	59	2.84264
2	7	377072	126	0	40	263	1376	3152	0	2917	9853	4893	39	15	2	45	54	2.290698
3	4	370520	7582	0	68	63	274	540	0	2499	9627	3803	53	16	11	19	69	1.970803
4	3	365697	6188	0	53	0	0	0	0	1963	9370	3820	54	27	4	15	81	
4	2	365723	1116	0	77	0	0	0	0	2142	10672	3519	65	23	4	8	88	
3	3	367585	219	0	15	94	324	1030	0	2186	7350	3583	44	12	25	20	56	3.179012
4	2	366760	62	0	63	38	210	636	0	2293	9361	3785	60	14	15	12	74	3.028571
6	2	365197	3075	0	59	31	295	4837	0	2068	9934	3567	71	20	3	6	91	16.39661
3	2	366024	968	0	11	0	35	282	0	2260	9364	3612	80	12	6	1	92	8.057143
4	2	368869	675	0	22	29	470	3797	0	2177	8811	3584	74	19	5	3	93	8.078723
3	2	367265	1837	0	5	1	11	25	0	1742	7950	3179	66	15	18	1	81	2.272727
3	2	363423	2982	0	13	0	0	0	0	2062	9280	3490	63	19	12	6	82	
3	3	364456	1270	0	17	35	354	1339	0	2124	8841	3782	59	18	14	10	77	3.782486
3	2	365300	242	0	5	15	178	631	0	2110	8162	3243	43	13	41	3	56	3.544944

Figure 3: Example data collection output

The ouput is shown in Figure 3.

Data analysis

In Figure 3 we can see very high CPU utilization, often touching 100% (shown in bold). The major contributor to this percentage usage of CPU time is us, which is 61% for node0 and 71% for node1.

Inferences:

- High CPU utilization can sometimes be attributed to excessive table scans or index scans on large tables. We have to look for indexing opportunities by analysing SQL statements which have very high rows read values (in SQL snapshots).
- The reason can be because database statistics haven't been gathered. You should use the **db2 runstats** command to ensure that the optimizer has the correct statistics.

Problems and recommendations:

- Set MAXAGENTS = 500 (a little more than MAXAPPLS).

  The MAXAGENTS parameter indicates the maximum number of database manager agents that are available at any given time to accept application requests. The value of MAXAGENTS should be at least the sum of the values for MAXAPPLS (maximum concurrent applications) in each database to be accessed concurrently. In our case we have only one database.

  Configuration parameter:

  – MAXAPPLS = 500

  – MAXAGENTS = 200

  – NUM_POOLAGENTS = 200

  – NUM_INITAGENTS = 0

  – MAX_COORDAGENTS =MAXAGENTS - NUM_

    INITAGENTS

    – MAX_CONNECTIONS = MAX_COORDAGENTS.

- Set NUM_POOLAGENTS => 500 (closer to MAXAGENTS).

  The NUM_POOLAGENTS parameter is a guideline for how large you want the agent pool to grow. If more agents are created than is indicated by the value of this parameter, they will be terminated when they finish executing their current request, rather than be returned to the pool. In an OLTP environment in which many applications are concurrently connected, the value of NUM_POOLAGENTS should be closer to the value of MAXAGENTS. CPU resource is utilized to initiate agents.

- Where a large number of sorts are observed, the sort process is very highly CPU intensive.

- A large number of sort overflows are observed. The sort overflow should ideally be less than 10. This causes high I/O. Statistics are not current, hence DB2 asks for smaller sort heaps leading to overflows.

  Configuration parameter:

  – SORTHEAP = 512

  – SHEAPTHRES = 10000.

  Database snapshot:

  – total sort heap allocated = 38778

  – total sorts = 12108

  – total sort time (ms) = 1789800

  – sort overflows = 2054

  – active sorts = 19.

- SORTHEAP is not sufficient. Increase SORTHEAP size. SHEAPTHRES should be a multiple of SORTHEAP.

- Use appropriate indexes to avoid sorts.
- More use of static SQL through stored procedures will prevent dynamic SQL compilation, which consumes high CPU.
- Looking at the database we find tables are very highly normalized. Use association tables and summary tables to avoid SQL with complex JOINS.
- Very high row reads suggest tablespace scan or index scan with very poor selectivity leading to high CPU utilization.

  Dynamic SQL snapshot:

  – number of executions = 21
  – number of compilations = 1
  – worst preparation time (ms) = 10578
  – best preparation time (ms) = 10578
  – internal rows deleted = 0
  – internal rows inserted = 0
  – rows read = 10163072
  – internal rows updated = 0
  – rows written = 1208
  – statement sorts = 0
  – total execution time (sec.ms) = 1084.435432
  – total user cpu time (sec.ms) = 190.33
  – total system cpu time (sec.ms) = 85.39.
- The reason can be because database statistics haven't been gathered. You should use the **db2 runstats** command to ensure that the optimizer has the correct statistics after large updates, inserts, deletes, and system-level change.

Notes

VMSTAT command

The **vmstat** command reports statistics about kernel threads, virtual memory, disks, traps, and CPU activity. Reports generated by the **vmstat** command can be used to balance system load activity. These system-wide statistics (among all processors) are calculated as averages for values expressed as percentages, and as sums otherwise.

The **vmstat** output column headings and their descriptions are:

- kthr – kernel thread state changes per second over the sampling interval:
  - r – number of kernel threads placed in run queue.
  - b – number of kernel threads placed in wait queue (awaiting resource, awaiting input/output).
- Memory – information about the usage of virtual and real memory. Virtual pages are considered active if they have been accessed. A page is 4096 bytes:
  - avm – active virtual pages.
  - fre – size of the free list.

  Note: a large portion of real memory is utilized as a cache for filesystem data. It is not unusual for the size of the free list to remain small.
- Page – information about page faults and paging activity. These are averaged over the interval and given in units per second:
  - re – pager input/output list.
  - pi – pages paged in from paging space.
  - po – pages paged out to paging space.
  - fr – pages freed (page replacement).

- sr – pages scanned by page-replacement algorithm.
- cy – clock cycles by page-replacement algorithm.

- Faults – trap and interrupt rate averages per second over the sampling interval:
  - in – device interrupts.
  - sy – system calls.
  - cs – kernel thread context switches.
- CPU – breakdown of percentage usage of CPU time:
  - us – user time.
  - sy – system time.
  - id – CPU idle time.
  - wa – CPU cycles to determine that the current process is wait and there is pending disk input/output.

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

T S Laxminarayan (ts_laxminarayan@yahoo.com)
System Programmer (India)

AIX news

Innovation Data Processing has announced a new version of UPSTREAM Reservoir. The product provides automated centralized storage management and control for back-up and recovery and archival for all systems, databases, and enterprise servers. This version now offers a choice of platforms used as the host storage back-up server – AIX, Windows, Sun Solaris, x86 Linux, and Linux for zSeries.

The UPSTREAM Reservoir back-up server can use any tape drive such as 9840, Magstar, DLT, LTO, AIT, Ultrium, or library. In addition, UPSTREAM Reservoir provides the ability to back-up to disk or tape, and provides the option for data to be backed up to disk, staged from disk, and automatically moved to tape.

For further information contact:
URL: www.innovationdp.fdr.com/products/upstream/upsreservoir.cfm.

* * *

IBM has launched Lotus Notes and Domino 7. IBM said that it has upgraded the Lotus Domino 7 toolset to include a new design element that positions Lotus Domino as a Web services host. Additionally, Lotus Notes and Domino 7 provide developers with the option of using either traditional NSF storage features or DB2 as the foundation for new and existing applications. This new capability gives developers the option of leveraging open-standard SQL. The release also includes new IT administration tools for increasing system performance and scalability, IBM claims.

Lotus Notes 7 includes more than 100 new features, including visual indicators that can help users organize and manage their in-box by highlighting high-priority messages, as well as differentiating between group e-mails and messages targeted for specific users. New memory functions can automatically save and return to open documents and applications upon shut down and restart. Instant messaging and presence technology, already integrated in the Lotus Notes client, has been expanded across the platform, including e-mails and calendar items that rapidly connect users with experts and key contacts.

Lotus Domino 7 is currently available for AIX 5.2 or 5.3 as well as iSeries, zSeries, Windows 2000 and 2003, Solaris 9, and Linux (x86).

For further information contact:
URL: www.lotus.com/products/product4.nsf/wdocs/notesdomino.

* * *

Interestingly, when Oracle announced a new record-breaking TPC-H 3TB data warehousing benchmark for Oracle Database 10g Release 2, it was running on an IBM eServer p5 595 system with 64 1.9GHz POWER5 processors using AIX 5L V5.3.

For further information contact:
URL: www.tpc.org.

* * *

IBM has announced Version 8.0 of WebSphere DataStage TX, a data integration suite. The suite will run on WebSphere and is used primarily for iSeries boxes running AIX and Linux, though it can also be deployed for the zSeries for batch, CICS, and USS usage.

For further information contact:
URL: www.ibm.com/software/data/integration/datastagetx.