108

AIX update

October 2004

In this issue

A itc PUBLICATION

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.

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.

Printed in England.

A simple and low cost performance monitor

INTRODUCTION

How many times has your boss, after having refused to buy a commercial performance management suite, rushed into your office asking about the state of all the AIX systems?

In order to address this problem I wrote a simple C daemon based on the AIX perfstat API. The daemon gets performance data at a given frequency and sends (a subset of) it to a specified server on a specified port using the UDP protocol.

On the receiver server you are free to decide how to handle all that data. In my opinion the best choice is to store it on RRD archives and then to use one on the many RRD tool front-ends (http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/index.html).

In this article I'm going to show you the piece of code I'm using to read system performance data and send it to a central location. I'm going to show you a (sample) simple collector daemon that receives all the performance data and stores it in RRD archives too.

THE PERFSTAT API

As I said in the introduction, in order to get system performance data from my AIX machines, I used the perfstat API. But what is it?

The perfstat API is a collection of C programming language subroutines that execute in user space, and they use the perfstat kernel extension to extract various AIX performance metrics. System component information is also retrieved from the Object Data Manager (ODM) and returned with the performance metrics.

The perfstat API is both a 32-bit and a 64-bit API, is thread-

safe, and does not require root authority. The API supports extensions, so binary compatibility is maintained across all releases of AIX. This frees the user from version dependencies.

The perfstat API subroutines reside in the *libperfstat.a* library and are part of the *bos.perf.libperfstat* fileset, which is installable from the AIX base installation media and requires that the *bos.perf.perfstat* fileset be installed. The latter contains the kernVÃ extension and is automatically installed with AIX.

The */usr/include/libperfstat.h* file contains the interface declarations and type definitions of the data structures to use when calling the interface. This file is also part of the *bos.perf.libperfstat* fileset.

Two types of API are available. Global types return global metrics related to a set of components, while individual types return metrics related to individual components. Both types of interface have similar signatures, but slightly different behaviour. Global interfaces report metrics related to a set of components on a system (such as processors, disks, or memory). Component-specific interfaces report metrics related to individual components on a system (such as a processor, disk, network interface, or paging space).

The daemon described in this article makes use of a subset of the global metrics.

All the interfaces return raw data; that is, values of running counters. Multiple calls must be made at regular intervals to calculate rates.

Several interfaces return data retrieved from the ODM (Object Data Manager) database. This information is automatically cached into a dictionary that is assumed to be ‘frozen’ after it is loaded. The perfstat_reset subroutine must be called to clear the dictionary whenever the machine configuration has changed.

Most types returned are unsigned long long – that is, unsigned 64-bit data. This provides complete kernel independence.

Some kernel internal metrics are in fact 32-bits wide in the 32-bit kernel, and 64-bits wide in the 64-bit kernel. The corresponding libperfstat API's data type is always unsigned 64-bit.

For a more detailed description of the perfstat API please have a look at the official AIX 5.*x* documentation (http://www.ibm.com/aix).

THE PERFORMANCE SENDER DAEMON

The daemon that gets and sends all the performance data from any AIX machine to the collector is named aixpsend. It must (obviously) be run on the AIX machine you want to monitor and gets some input parameters – the 'collector' hostname (that is the machine running the collector), the UDP port the collector is listening on, and the sampling interval. If you don't specify any parameters, a small help file is shown. The daemon uses the syslogd facility and must be run by root. There should be no security issues because the daemon doesn't open listening sockets.

The source code for the aixpsend daemon has been split into four different files – *aixperf.c*, *totperf.c*, *commchannel.c*, and *aixpsend.c*.

The *aixpsend.c* file contains the main function. It basically makes up some initialization steps – detachment from terminal, stdout, stdin, and stderr closure, UDP socket opening, etc – and then enters the get/prepare/send data loop. The *commchannel.c* file contains all the code needed to manage the UDP channel used to send data to the collector machine. The *totperf.c* file contains all the code needed to manage the packets sent to the collector machine. The *aixperf.c* file contains all the code needed to retrieve the performance data from the AIX machine by using the perfstat API.

In order to compile the aixpsend application, you must copy all the files contained into the aixpsend.zip file into a directory, edit the Makefile in order to be sure you are using your own

C/C++ compiler as well as your desired compilation options, and issue the classic **make** command. The executable file, aixpsend, will be produced.

Let's have a closer look at the code. It may, in fact, be customized in order to send different performance data. For example, if you want to modify which data among the global metrics exposed by the perfstat API is to be sent to the central collector, the subroutines that need to be customized are buildcpudata, builddskdata, buildnetdata, and buildmemdata (contained in the *aixperf.c* file). Inside each of these, a string is prepared containing all the data you want to send. It's the same if you want to send some other 'per resource' performance metric exposed by the perfstat API. You may want to modify the collecttotperfdata routine inside the *aixperf.c file*, and so on. Please remember also to modify the collector's code.

The actual transmission is performed inside the sendtotperfpacket routine contained in the *totperf.c* file. This routine uses the UDP channel that has been opened before entering the main get/prepare/send data loop. Moreover this routine adds to the packet a CRC code – so the receiver will be able to understand whether the received packet is a good one or not.

You are free to choose which data is to be sent to the central collector from all the data made available through the perfstat API. I've decided to send (almost) all of the global data exposed by the perfstat API about the CPU, MEM, DSK, and NET categories. In fact all of them may be useful in order to monitor a machine.

Here is an example of execution on the daemon. In this example, all the data is sent to the syslogd daemon running on the same machine (localhost) with a sampling interval of 10 seconds:

```
aixpsend -d localhost -p 514 -i 10
```

You need to execute the daemon with root permission.

If, for example, the syslogd configuration file contains a line

similar to the following:

```
*.debug                 /tmp/syslog.out     rotate size 100k files 4
```

all the performance data will be written to the */tmp/syslog.out* file. By redirecting performance data collected on different machines to a single syslogd daemon, you may build up a single repository containing all the data. Obviously you may redirect data to a different UDP listener.

A VERY SIMPLE PERFORMANCE DATA COLLECTOR

The file *perfmngd.tar* contains the code for a really very simple collector for the performance data sent by the AIX performance daemon. It runs on the Linux platform and requires the RRD tool to be installed and enabled, as well as the (x)inetd daemon to be up and running. In fact, it has been implemented as a new (x)inetd service.

In order to build the perfmngd executable, you must explode the perfmngd.tar file into a directory on your Linux collect station and issue the classic **make** command.

The source of the collecting server has been split into four files. Some of them are shared with aixpsend and have been described above. The new ones are *perfmngd.c* and *rrddb.c*. The first one contains the main function. It reads data from the standard input, decodes and validates the received packet, and then writes down the data to the suitable RRD archive. The second one contains all the code needed to interact with the RRD archives.

In order to run the new service inside an (x)inetd environment, you should add something into its configuration file (please refer to the documentation about your inetd super-daemon). For example, to add it onto a SUSE 9.0 GNU/Linux distribution I've added the following file into the directory */etc/xinetd.d*:

```
service yamp
{
      socket_type     = dgram
      protocol        = udp
```

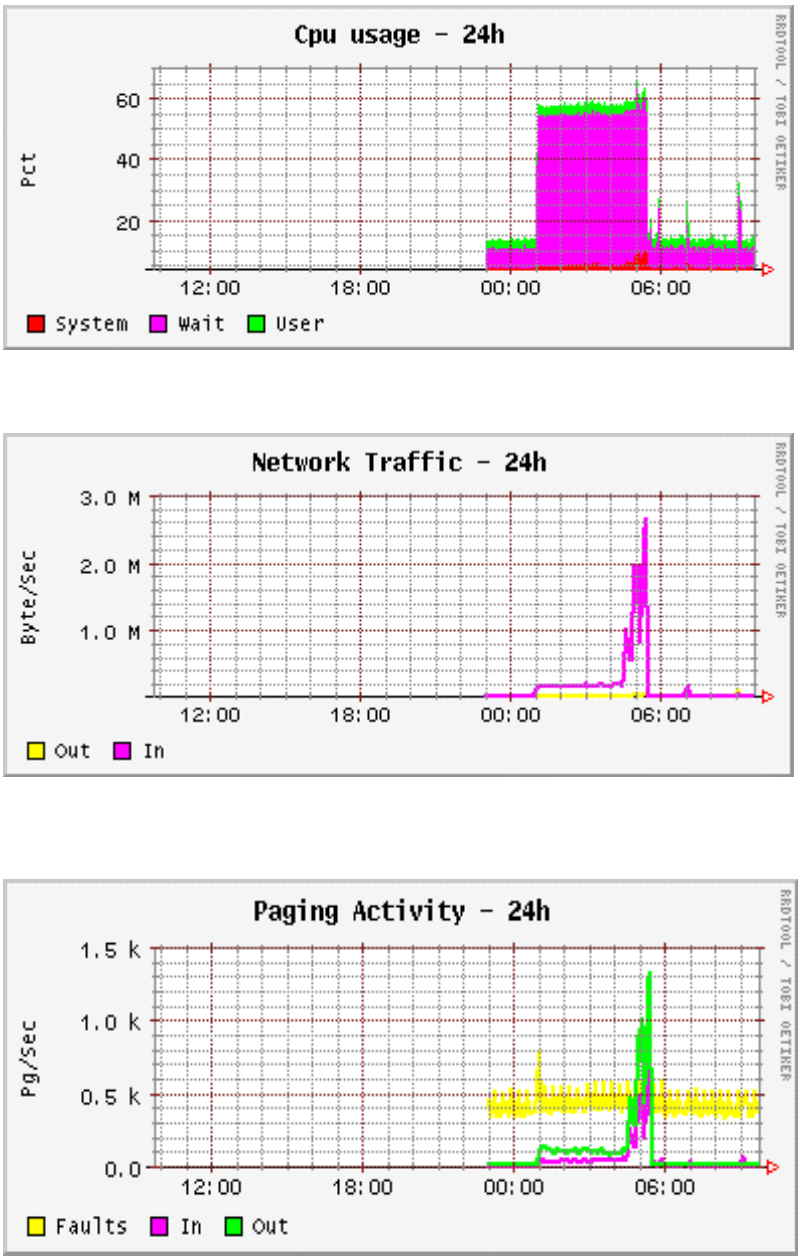

Figure 1: Output examples

```
        wait            = no
        user            = yamp
        server          = /home/yamp/collector/perfmngd
        server_args     = -w /home/yamp/rrdarchive
        id              = yamp
        type            = UNLISTED
        port            = 1765
}
```

The perfmngd executable asks for an input parameter – the directory where all the RRD archives are placed. When adding a new AIX machine to the monitored set, you must run the shell script that creates and initializes all the RRD archives related to that machine. The script is contained in the perfmngd.tar file and is named *createrrd.sh*. Please be sure that the parameter samplint set here is the same as the one used by the corresponding aixpsend on the monitored machine.

After you have been feeding your RRD archives for some time, you may want to manipulate the data received, for example to display graphs. Figure 1 shows examples of what you can obtain; but remember – the only limit is your imagination!

The files aixpsend.tar and perfmngd.tar are available for download from www.xephon.com/extras/aixpsend.tar and /extras/perfmngd.tar.

Marco Pirini
IT Architect (Italy)

AIX storage benchmark tool

There will always be a need to find the optimum disk layout for each application that will allow it to perform better. There are lots of tools available to find the best settings for these applications. However, the performance of a disk may vary depending on the internal architecture of the storage hardware used for the disk storage. The following script is written with this in mind, to help the AIX administrator when deciding what kind of AIX filesystem will perform best for his/her server.

PREREQUISITES

In order to run the following script, the AIX administrator has to create three filesystems as follows:

1 A concatenated filesystem, which might even be a single physical volume. Create the volume as */dev/concat* and mount it on the */concat* directory.

2 A striped filesystem with a stripe size of 64MB. This filesystem should have a minimum of two physical volumes. Create one volume as */dev/strip64* and mount it on the */strip64* directory.

3 A striped filesystem with a logical volume striping of 128KB. This filesystem should also have a minimum of two physical volumes. Create the volume as */dev/lvm128kb* and mount it on the */lvm128kb* directory.

Note: each filesystem should have a minimum of 4 gigabytes of storage.

An administrator may create different kinds of filesystem and may use them with only slight modification to the following scripts.

We are using IOZONE toll, which is freeware to drive the I/O on the storage disks. Therefore, download this AIX tool, and put it in the */usr/sbin* directory. To collect I/O statistics, we use the AIX built-in filemon tool.

DESCRIPTION

This script runs with different I/O block sizes on each filesystem for write and read operations. We are using random write and sequential read in our example. This can be changed according to your needs by passing a different parameter in the IOZONE command.

This script generates filemon output for each operation. For I/O block sizes of 8KB on the concatenated volume for a write operation, it creates filemon output in the current directory. Similarly, it creates filemon output for all the operations. By carefully looking at all the filemon output, an administrator can decide which size is best for his/her application.

#

```
# Storage Performance Benchmark Tool
#
# Muthukumar,
# Washington, DC
#
CONCAT_VOLUME="/dev/concat"
CONCAT_FILESYS="/concat"
STRIPED_64MB_VOLUME="/dev/strip64"
STRIPED_64MB_FILESYS="/strip64"
LVM_STRIP_128KB_VOLUME="/dev/lvm128kb"
LVM_STRIP_128KB_FILESYS="/lvm128kb"
IOSIZE="8 16 24 32 64 128 256 512 1Ø24"
FILE_SIZE="4Ø96m"
TESTTYPE="Storage disk benchmark test for AIX for concatenated
filesystems"
for iosize in $IOSIZE
do
#
# Unmount the filesystem and mount it again to clear the I/O statistics
#
umount $CONCAT_FILESYS
mount $CONCAT_VOLUME $CONCAT_FILESYS
rm -f $ CONCAT_FILESYS/x
write_filename="concat-iosize-$IOSIZE-write"
filemon -o $write_filename -O pv,lv
/usr/sbin/iozone -i Ø -i 2 -s $FILE_SIZE -r $IOSIZE -f $FILESYS/x -w
echo "iocount = 1 blocksize=$iosize KB iterations = $NREPS
$TESTTYPE"
        echo "   "
  trcstop
# Unmount the filesystem and mount it after 3Ø seconds, which helps to
# flush file content from memory - so that we can get true Disk I/O
# performance
umount $CONCAT_FILESYS
#
# Sleep for 3Ø seconds to flush the file content from the memory.
#
sleep 3Ø
mount $CONCAT_VOLUME $CONCAT_FILESYS
read_filename="concat-iosize-$IOSIZE-read"
filemon -o $read_filename -O pv,lv
       /usr/sbin/iozone -i 1 -s $FILE_SIZE -r $IOSIZE -f $FILESYS/x
trcstop
done

TESTTYPE="Storage disk benchmark test for AIX for striped 64MB
filesystems"
for iosize in $IOSIZE
do
#
```

```
# Unmount the filesystem and mount it again to clear the I/O statistics
#
umount $STRIPED_64MB_FILESYS
mount $STRIPED_64MB_VOLUME $STRIPED_64MB_FILESYS
rm -f $STRIPED_64MB_FILESYS/x
write_filename="striped-iosize-$IOSIZE-write"
filemon -o $write_filename -O pv,lv
/usr/sbin/iozone -i 0 -i 2 -s $FILE_SIZE -r $IOSIZE -f
$STRIPED_64MB_FILESYS/x -w
echo "iocount = 1 blocksize=$iosize KB $TESTTYPE"
        echo "   "
  trcstop
# Unmount the filesystem and mount it after 30 seconds, which helps to
# flush file content from memory - so that we can get true Disk I/O
# performance
umount $STRIPED_64MB_FILESYS
#
# Sleep for 30 seconds to flush the file content from the memory.
#
sleep 30
mount $STRIPED_64MB_VOLUME $STRIPED_64MB_FILESYS
read_filename="striped-iosize-$IOSIZE-read"
filemon -o $read_filename -O pv,lv
       /usr/sbin/iozone -i 1 -s $FILE_SIZE -r $IOSIZE -f
$STRIPED_64MB_FILESYS/x
trcstop
done

TESTTYPE="Storage disk benchmark test for AIX for LVM stripe 128KB
systems"
for iosize in $IOSIZE
do
#
# Unmount the filesystem and mount it again to clear the I/O statistics
#
umount $LVM_STRIPE_128KB_FILESYS
mount $LVM_STRIPE_128KB_VOLUME $LVM_STRIPE_128KB_FILESYS
rm -f $LVM_STRIPE_128KB_FILESYS/x
write_filename="lvm_striped-iosize-$IOSIZE-write"
filemon -o $write_filename -O pv,lv
/usr/sbin/iozone -i 0 -i 2 -s $FILE_SIZE -r $IOSIZE -f
$LVM_STRIPE_128KB_FILESYS/x -w
echo "iocount = 1 blocksize=$iosize KB $TESTTYPE"
        echo "   "
  trcstop
# Unmount the filesystem and mount it after 30 seconds, which helps to
# flush file content from memory - so that we can get true Disk I/O
# performance
umount $LVM_STRIPE_128KB_FILESYS
#
```

```
# Sleep for 30 seconds to flush the file content from the memory.
#
sleep 30
mount $LVM_STRIPE_128KB_VOLUME $LVM_STRIPE_128KB_FILESYS
read_filename="lvm-stripe-iosize-$IOSIZE-read"
filemon -o $read_filename -O pv,lv
       /usr/sbin/iozone -i 1 -s $FILE_SIZE -r $IOSIZE -f
$LVM_STRIPE_128KB_FILESYS/x
trcstop
done
```

Muthukumar Kannaiyan
Storage Administrator
IORMYX (USA)

MySQL

MySQL is one of the fastest RDBMS (Relational Database Servers) in use today. MySQL can be downloaded from www.mysql.com and compiled from source (the currently stable version is 4.2.0). Alternatively, you can get the binaries from www.bullfreeware.com. The only current release they offer is 3.23. Be sure to download both the client and the server packages. Either way, be sure to check out the AIX notes in the MySQL installation guide for specific advice about compiling.

A good feature of the database server is the use of index compression on string indexes, which saves space on the tables that hold this type of index. Up to 32 non-clustered indexes per table are allowed and up to 16 columns may be allocated to an index.

When creating tables, the DBA can specify Transaction Safety Tables (TST) of type InnoDB. This enables rollback on all un-committed transactions. Unfortunately, using this type of table does create a slight overhead in performance and, because of its function, will use more disk space to hold the transaction logging. This is in comparison with ordinary table types (ie non transaction safe) like HEAP, ISAM, MERGE, and MyISAM (where MyISAM is the default table type).

MEMORY AND MYSQL

As with any RDBMS, the more memory your throw at MySQL the better. It will start up with a basic set of memory configurations. For a dedicated server running MySQL, the three main areas a DBA should be thinking of changing from day 1 are:

- key_buffer
- sort buffer

- query cache.

The key_buffer is used for index cacheing on queries that use the indexes. This saves MySQL from looking through the index during a transaction. Think of adding at least 10% of the total server memory to this configuration. If the database server involves a lot of sorting during its processing, then change the sort_buffer value so that all or most of the sorting is carried out in cache – give it, say, 15% of the total memory. The query cache is probably the most important configuration with regard to performance. It allows the most frequent queries to be cached, and setting this correctly is a bit of trial and error. There are three settings involved – query_cache_limit, query_cache_size, and query_cache_type. To enable the query cacheing, set the variable query_cache_type to 1; this sets it to enable. The cache_limit denotes that queries that are larger than 1MB (the default) should not be cached. Try increasing this value, in increments of 1MB, to a maximum of 40MB, and monitor performance. The cache size is by default disabled (ie 0). If you are in a production environment, you will need this on. It is best to start off small and experiment with running queries and see how the performance improves. Use the SHOW STATUS command to monitor cache performance.

To display the current parameter values set within MySQL, issue the SHOW VARIABLES command. This command will return all variables and their current values, which can in turn be used to set new configuration values. To assign or change a value, use the SET command. For example, to set the sort buffer size to 150MB, the following would be used:

```
SET sort_buffer_size=150M
```

All the server variables and command line options used by MySQL can be put into a global config file called */etc/my.cnf*. Local users can also create their own local start-up file located in their $HOME directory called *.my.cnf*. This local file is read when a user invokes mysql.

The following entry in a local *.my.cnf* file, would connect the

user dxtans with the password of mayday to the host bumper, and automatically change into the database karate:

```
[client]
  host = bumper
  user = dxtans
  password = mayday
[mysql]
 database = karate
```

The main configuration settings file, */etc/my.cnf*, is read by MySQL when starting up. This file can be edited directly to change configuration settings – be sure to make a copy first. Restarting MySQL will force the database server to read its new configuration setting. I strongly advise that all configuration settings for MySQL that affect performance should have an entry in this file. Keeping it all in a file makes making any changes simpler, and it is easier to undo any suspect changes a DBA may have made.

USING MYSQL

Creating and maintaining users are two of the main tasks of a DBA. By using the GRANT/REVOKE command, user access can be maintained:

```
mysql>grant all privileges on karate.* to
-> dxtans@"localhost" identified by 'mayday';
```

In the above example, user dxtans is granted full privileges on the database karate including all tables (.*) from the local host with a password of mayday. The user can change their password later using the SET command once they have access to MySQL. The following statement changes the user's own password to master:

```
mysql> set password = password ('master');
```

In a large company environment, a DBA would want to allow access from users within the company domain, because this is easier to maintain and administer. For example, to grant access to user pxlong from any host within the domain company.co.uk, with the password secret, but allow only

select, insert, and update permissions to the database accounts:

```
Mysql>grant select, insert, update privileges on accounts.* to
pxlong@"%.company.co.uk" identified by 'secret';
```

The %.company.co.uk (%. is a wildcard) will match all hosts that belong to the domain company.co.uk.

To change a user's authority, simply use the REVOKE command. To take away the update and insert privilege from user pxlong on all tables in the accounts database, thus leaving user pxlong with only select access, we could use:

```
mysql> revoke insert, update on accounts.*
-> from user pxlong@"%.company.co.uk";
```

For the changes to take effect, the privileges table must be flushed:

```
mysql> flush privileges;
```

Connecting or logging into MySQL is generally done via the mysql prompt on the command line, though there are other client applications around that offer GUI interfaces. Supply the username and password. If a user is connecting from another server, you must also supply the hostname as well. Alternatively, if you have set up your *.my.cnf* file to hold your user connection details, simply type mysql, and the parameter values will be parsed to mysql on connection:

```
$ mysql -Udxtans -Pmayday
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 3.23.36
Type 'help;' or '\h' for help. Type '\c' to clear the buffer
mysql>
```

All commands or queries with MySQL are by default terminated by issuing a semi-colon (;). When MySQL sees this semi-colon, it will execute the command.

As a new user, be sure to change your password at the earliest opportunity.

By default, MySQL will always display output surrounded by

hashes, and the time taken to run the command. To disable this feature, start MySQL with the silent parameter:

```
mysql -s
```

To see who you are currently logged in as, issue the USER command:

```
mysql> select user();
+------------------+
| user()           |
+------------------+
| dxtans@localhost |
+------------------+
```

To display all the current databases in MySQL:

```
mysql> show databases;
+--------------+
| Database     |
+--------------+
| mysql        |
| test         |
| gotcha       |
| holding      |
| karate       |
| accounts     |
+--------------+
```

Before data can be inserted into a table, a database must first be created to hold the tables. The following example creates a database called karate:

```
mysql> create database karate;
```

Next, change to that working database.

```
mysql> use karate;
Database changed
```

The following example creates a table called karate_students. Notice the column name student_ID – MySQL is being instructed that this column is an integer and NOT NULL. This prevents null values being entered into that column. The column has also been created as an AUTO_INCREMENT. This means that for each row inserted, this column will increment by 1. Because the column's contents will always be unique, it makes sense to initially specify this as the primary

key for the table.

```
mysql> CREATE TABLE karate_students (
    -> name VARCHAR(20),
    -> belt VARCHAR(10),
    -> sex CHAR(1),
    -> student_type CHAR(1),
    -> student_ID INT NOT NULL AUTO_INCREMENT,
    -> PRIMARY KEY (student_ID));
```

To view all the current tables in a database:

```
mysql> show tables;
+-----------------+
| Tables_in_karate |
+-----------------+
| karate_students  |
| karate_grading   |
| karate_events    |
| karate_lookups   |
+-----------------+
```

To view the structure of a table:

```
mysql> describe karate_students;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| name         | varchar(20) | YES  |     | NULL    |                |
| belt         | varchar(10) | YES  |     | NULL    |                |
| sex          | char(1)     | YES  |     | NULL    |                |
| student_type | char(1)     | YES  |     | NULL    |                |
| student_ID   | int(11)     |      | PRI | NULL    | auto_increment |
+--------------+-------------+------+-----+---------+----------------+
```

This output is quite handy when you want to know quickly what your columns and data types are.

Inevitably, once a table has been created, it is usually the case that the table needs to be tweaked a bit regarding its structure – for example adding or deleting some columns. To make amendments to a table's attribute use the ALTER TABLE command. The following command will rename the table karate_events to karate_comps:

```
mysql> alter table karate_events rename karate_comps;
```

One of the most common changes to a table's structure is the addition and deletion of columns. The following two examples

will first add a new column called telephone of type int(11). The second drops the column contact from the table karate_events:

```
mysql> alter table karate_events add telephone int(10);

mysql> alter table karate_events drop column contact;
```

To drop a database or table use the DROP command:

```
drop database <database_name>;
drop table <table_name>;
```

CHANGING AND LOADING DATA

Once the tables are set up to your satisfaction, the next task will be to insert some data. Typically, after table creation, this would be achieved by exporting the data from some other source into a text file, ready for import into MySQL.

Using the LOAD DATA command, data can be inserted *en masse* into a table. By default, the columns in the text file are <tab> separated, and the end of a record (or row) is terminated by a new line. These can be changed using the LOAD DATA command options. The following example uses a prepared text file to load (data) rows into the table karate_students. The use of the word LOCAL means the load will take place on the local server.

```
mysql> load data local infile "/tmp/karate_students.in.txt"
-> into table karate_students;
```

Inserting individual rows is achieved via the INSERT command. When inserting data into text fields, the text is surrounded with quotes. Do not surround an integer or real number with quotes if it is inbound to an int or real column type because MySQL will interpret this value as text and not a number. Also, if a table has an AUTO_INCREMENT column, just put a pair of empty quotes in its place; MySQL will take care of the incrementing values.

```
mysql>insert into karate_students values
-> ('P.Somename','Green','M','J','');
```

Use the UPDATE command to update a record in a table. The

following will change the field value (of the column named belt) to Blue. When making updates be sure the updating is taking place on the correct row. Using a SELECT statement, select some columns that will retrieve your desired row, like a person's name. Then use the ID of that row to change the record contents, like so:

```
mysql > select name, belt, student_ID from karate_students
where name="A.Foxtrot";
+-----------+--------+------------+
| name      | belt   | student_ID |
+-----------+--------+------------+
| A.Foxtrot | Yellow |          6 |
+-----------+--------+------------+
```

Now do the change:

```
mysql> update karate_students set belt="Blue"
-> where student_ID=6;
```

To delete a user, make sure the correct record is being retrieved based on a SELECT query first. Once the correct row has been retrieved, the row or record can then be deleted:

```
mysql> delete from karate_students where ID=14;
```

To delete all data from a table, thus leaving just the table structure intact (a common task during the development cycle of a database), use the delete or truncate command:

```
mysql> delete from <table_name>;
mysql> truncate table <table_name>
```

A database is worthless if the data is not secured – by which I mean backed up. The **msqldump** utility, which is run from the command line, backs up individual tables and their structure (in SQL format), or the whole database can be dumped out to disk. Once the database(s) have been dumped, the file should be archived to tape. To back up a database use:

```
$ mysqldump -l <database_name>   >   <database_name>.sql
```

The -l will first lock all tables in read-only mode before dumping. Also notice that the output is redirected into another file, ideally called .sql, but not necessarily so. This becomes apparent when you wish to restore the whole database, by

using command redirection whilst invoking mysql. The following reloads the whole database structure plus all the data held within it:

```
$ mysql <database_name>   <   <database_name.sql>
```

If disk space is at a premium, zip it on-the-fly:

```
$ mysqldump -l <databse_name> | gzip >  <database_name>.gz
```

To restore a zipped file on-the-fly, use:

```
$ gunzip <   <database_name>.gz | mysql
```

ADMINISTRATION TOOLS

Administering any RDBMS from the command line is slow and the DBA is open to creating numerous typos. Although MySQL's tool **mysqladmin** does a good job, a client tool is essential for managing MySQL on a production or development server. The *de facto* Web-based package tool, phpMyAdmin, is essential if one is providing full on-line support; however, you will need PHP installed to run it.

David Tansley
Global Operations
ACE Overseas General (UK)

IBM pSeries and AIX systems installation and maintenance recommendations

During my long experience working with the IBM pSeries line of servers and AIX OS, I have accumulated valuable knowledge in configuration, set-up, and installation of such systems.

This article will present some general set-up guidelines that are considered by me to present a summary of my experience and could benefit other users.

HARDWARE CONFIGURATION AND SET UP

When a new server is acquired, it is essential to make sure that, apart from being able to run the desired workload effectively, the system is configured with all available features that are supported by the chosen server model. The price of these features, if they are purchased at the time of server acquisition, is usually a small fraction of the total price. However, they can significantly enhance the RAS (Reliability, Availability, and Serviceability) of the whole system.

The following list summarizes some of the typical features:

- Three internal disks: one for the active OS installation, one for a mirror of the active system disk, and one for an alternative system disk back-up.
- Two network adapters: one used for the server client's access, and the other one for administration tasks such as back-up, management, monitoring, NFS servicing, etc.
- One CD for OS and third-party software installation.
- One tape drive for system back-up as well as small user and application data back-up and restore applications. In the case of partitions, it is possible to install one tape drive and one CD in an external enclosure and share the devices between partitions.
- Redundant power supply, if available for a given system.
- One graphic adapter or connectivity to HMC for server console access.
- Two Fibre Channel storage adapters for SAN connectivity. Typically these adapters can be set to use third-party software or drivers to support increased performance and availability by simultaneously connecting both of them to the storage device. Typically software is unique for different storage hardware vendors – Storage Manager for FastT, SDD for ESS Shark, PowerPath for EMC, AutoPath for HP, and DLM for HDS.

Slot placement should be in accordance with PCI Adapter Placement Reference http://publib16.boulder.ibm.com/pseries/en_US/infocenter/base/HW_pci_adp_pl.htm.

BASIC OS INSTALLATION

When the AIX OS is installed on a server, some basic characteristics have to be selected. Some of these characteristics cannot be altered afterwards. The most important one is whether to use 64-bit kernel and JFS2 for the system disk. My recommendation is to select this option, despite some issues that have been previously reported for JFS2. In my view, enhanced scalability and improved performance of JFS2 are compelling reasons to use it.

As for 64-bit kernel, it does support 32-bit programs and drivers as well, while providing enhanced scalability features and support for 64-bit operations.

Another irreversible option is whether to install the Trusted Computer Base (TCB). This feature should be installed only if required by data security regulation at the site because it cannot be uninstalled and, in my view, could have a negative effect on the performance of the system.

In general my recommendation is to install all software provided on the AIX OS CDs.

Presently, the size of the smallest internal disk provided with IBM pSeries is 36GB. So installing all the OS components leaves a lot of available space.

At least the following filesets should be installed for more space- conscious sites:

- bos.utils – DOS filesets.
- bos.adt.samples (pre 5.2) or bos.perf.tune at AIX 5.2 and above.
- bos.contents_list.

- man pages.

Install Netscape Navigator, Adobe Acrobat Reader, and IBM HTTP Server, which are available on the bonus pack CD.

Look through the Linux Toolbox CD and install the tools that are used at your site.

Additional tools to consider are nmon and nmon analyzer. These unsupported tools are available free of charge by downloading from http://www106.ibm.com/developerworks/eserver/articles/analyze_aix/.

Create a CD-ROM filesystem with a mount point of */cdrom*.

Enlarge the size of filesystems to keep their sizes no less than:

- / – 512MB.
- /local – 3048MB.
- /local64 – 1024MB (optional).
- /opt – 3062MB.
- /tmp – 512MB.
- /usr – 2048MB.
- /var – 2048MB.

Define a swap space size equal to the size of memory, divided into several equal-sized parts, each one located on a separate physical disk.

When AIX is first installed, run the **/usr/lib/ras/dumpcheck –p** command to make sure that adequate dump space has been allocated. Because the dump space requirement tends to grow as the system gets busier, configure dumpcheck to run regularly at a time when the system is likely to be fairly heavily loaded. For systems with large memory, the compression feature of system dump should be enabled in order to cut down dump space requirements:

```
/usr/bin/sysdumpdev -C
```

SYSTEM DISK BACK-UP

System disk availability and integrity is essential for the successful operation of the AIX OS, therefore it is imperative to protect it from hardware as well software or human errors.

In order to prevent outages caused by disk hardware, the system disk should always be mirrored, and the root volume group quorum turned off (in a single disk mirroring configuration).

The **alt_disk_install** command should be utilized to clone a running, optimally laid out system after the testing of the initial installation has been completed. A third internal system disk should be used for this purpose.

In addition, a tape back-up produced using the **mksysb** command should be prepared after the initial install is completed. Ensure that the prompt field is set to NO in *bos.inst.images*. Label the tape and make it write protected. Store the tape in a fireproof safe, preferably offsite.

It is imperative to perform testing of the validity of both the alternative OS back-up disk as well as of the mksysb tape.

Subsequently **alt_disk_install**-based OS disk cloning as well as **mksysb**-based tape back-up should be performed periodically, preferably on a weekly basis.

NAMING CONVENTIONS

Recommended hostname naming conventions are shown below:

```
<Platform>      <model><count><partition>

ibm63001  - ibm + 630  +  01            - Model p630 without partitions.

ibm65003a - ibm + 650  +  03 +    a     - partition of Model p650.
```

To allow volume groups transfer between the systems there should be a unique name for VG. One possible method to create one is:

```
            vg  <host><partition><count>

vg653aØ1 - vg + 653  +    a     + Ø1
```

For the same reason, logical volume names should be unique as well. One of the methods to create one is:

```
              lv  <host><partition><vg><count>

lv653aØ1Ø1 - lv + 563 +       a   +Ø1 + Ø1
```

Logical volumes residing on protected (RAID) storage should be striped with the maximal stripe size equal to 128KB in order to improve I/O performance.

For improved I/O performance, all defined filesystems should use the JFS2 filesystem type.

OS PARAMETERS

Use the following parameters:

- Increase the number of logins up to 32,767:

  ```
  chlicense -u 32767 -I
  ```

- Increase the maximal number of processes per user:

  ```
  chdev  -l sysØ -a maxuproc=5ØØØ
  ```

- Keep I/O history, enlarge the size of NCARGS:

  ```
  chdev -l sysØ -a iostat=true -a fullcore=true -a ncargs=128
  ```

- Enable legacy asynchronous I/O:

  ```
  chdev -l aioØ -P -a autoconfig=available; mkdev -l aioØ
  ```

- Define process resource limits for users.

  Set the following stanza ‘defaults’ in */etc/security/limit*:

  – fsize = -1

  – core = -1

  – cpu = -1

  – data = 4194302

- data_hard = -1
- rss = 4194302
- rss_hard = 4194302
- stack = 4194302
- stack_hard = 4194302
- nofiles = -1.

TUNING VIRTUAL MEMORY PARAMETERS

The tuning of virtual memory parameters is one of the most important factors affecting the performance of your system. The following is a description of virtual memory tuning tools and the parameters affected by them.

vmtune is one of the most important tools to load on your system. It is contained in *bos.adt.samples* and is installed in */usr/samples/kernel*. This is the tool that allows you to tune the virtual memory manager and some aspects of I/O.

With the AIX V5.2 release, vmtune was replaced by the tuning parameters vmo and ioo. schedtune was replaced by the tuning parameter schedtune. All of these parameters (along with noo and nfso) support the ability to retain tuning parameters in the files residing under the */etc/tunables* directory.

Although vmtune and schedtune can still be run, the appropriate vmo, ioo, or schedtune command should be utilized. vmtune and schedtune are still available for backward compatibility but have limited functionality.

Below are some of the advantages of new commands:

- Command consistency.
- Options for display or change.
- Ability to control changes now, next boot, all.
- Ability to return to defaults, check consistency, save, or propagate.

- Commands supported from SMIT or WSM.

The following summarizes some of the common flags of the commands vmo, ioo, schedo, no, and nfso:

- -a – displays values for all tunable parameters, one per line value.
- -h – displays command help or displays help about tunables.
- -d – resets tunables to default values.
- -D – resets all tunables to their default value.
- -o – tunable=value, sets tunable to a specified value.
- -p – makes changes apply to both current and reboot values; modifying the */etc/tunables/nextboot* file in addition to updating the current value.
- -r – makes changes apply to reboot values only. Modifies */etc/tunables/nextboot.fil* only.
- -L – prints header and characteristics of one or all tunables, one tunable per line.

We will look at two examples of Virtual Memory Tuning:

- maxperm/minperm – to control what types (file pages or computational pages) of page are stolen first.
- maxfree/minfree – to control at what levels the page replacement algorithm will begin or stop stealing pages.

TUNING MINPERM, MAXPERM, AND MAXCLIENT

AIX provides a mechanism for you to loosely control the ratio of page frames used for files versus those used for computational (working or program text) segments by adjusting the minperm and maxperm values according to the following guidelines:

- If the percentage of real memory occupied by file pages

falls below the minperm value, the page-replacement algorithm steals both file and computational pages, regardless of repage rates.

- If the percentage of real memory occupied by file pages rises above the maxperm value, the page-replacement algorithm steals both file and computational pages.
- If the percentage of real memory occupied by file pages is between the minperm and maxperm parameter values, the Virtual Memory Manager (VMM) normally steals only file pages, but if the repaging rate for file pages is higher then the repaging rate for computational pages, the computational pages are stolen as well.

If the load on the system is relatively unknown, using the values below could be considered as a good starting point:

- minperm = 15
- maxperm = 60.

Examine the values for numperm during valid production periods (disregard the back-up period), you can use the nmon tool to record numperm value:

- Set maxperm 5% to 10% lower than the numperm value.
- Set minperm to 5% less than the maxperm value.
- Set maxclient to be equal to maxperm.

TUNING MINFREE AND MAXFREE

If memory demand continues after the minfree value is reached, then processes may be suspended or killed. When the number of free pages is equal to or smaller than maxfree, the algorithm no longer frees pages. There will be insufficient pages relative to the total system memory to satisfy demand.

On a system with large memory or SMP, the defaults of 120 and 128 are a very small percentage of the real memory available.

In order to calculate the required values for minfree and maxfree the number of available memory pools should be found. This number can be determined by execution of **vmtune –a** or **vmstat –v** (for AIX 5.2 and above).

minfree = (120 + 4) * # of memory pools.

maxfree = minfree + (maxpgahead (or j2maxpgahead) * # of memory pools).

I/O TUNING

If you are using Oracle databases, you should use the asynchronous I/O features of AIX. You should set the value of the AIO parameters to the following initial values:

- minservers = 80
- maxservers = 200
- maxrequests = 8192.

If performance monitoring programs such as vmstat, topas, and nmon are reporting that over 35% of the time your system is busy doing I/O wait, you should tune the I/O-related parameters. Some areas to address are:

- Recent technology disks will support higher LTG numbers.
- **lvmstat** (must be enabled prior to usage) provides detailed information for I/O contention.
- **filemon** is an excellent I/O tool (trace – ensure you turn it off).

I/O TUNING PARAMETERS

numfsbufs (**vmtune –b**) specifies the number of filesystem buffer structures. This value is critical because VMM will put a process on the wait list if there are insufficient free buffer structures. A good initial value for this parameter is 186.

Run **vmtune –a** (pre 5.2) or **vmstat –v** (5.2 and above) and

monitor fsbufwaitcnt. This is incremented each time an I/O operation has to wait for filesystem buffer structures.

A general technique is to double the numfsbufs value (up to a maximum of 512) until fsbufwaitcount no longer increases. The setting of this value, because it is dynamic, should be re-executed at boot time, prior to any **mount all** command.

hd_pbuf_cnt (**vmtune –B**) determines the number of pbufs assigned to LVM. pbufs are pinned memory buffers used to hold pending I/O requests. Use the following formula to set the initial value of this parameter:

hd_pbuf_cnt = (# of disks attached to the server (physical or LUNs) + 4) times 120.

Again, examine **vmtune –a** and review the psbufwaitcnt. If it is increasing, multiply the current hd_pbuf_cnt by two until the psbufwaitcnt stops incrementing. Because the hd_pbuf_cnt can only be reduced via a reboot (this is pinned memory) – be frugal when increasing this value.

NETWORK TUNING

Check thewall (add at the end of */etc/rc.net*):

```
 no -a | grep thewall
```

If thewall is less than 1GB (1,048,576 KB), increase it dynamically:

```
no -o thewall=1048576
```

Also check that the maxmbuf attribute on sys0 is zero:

```
lsattr -El sys0 | grep maxmbuf
```

If not, set it:

```
chdev -l sys0 -a maxmbuf=0
```

Check sb_max (add at the end of */etc/rc.net*):

```
no -a | grep sb_max
```

If sb_max is less than 1MB (1,048,576 bytes), increase it dynamically

```
no -o sb_max=1048576
```

Check tcp_sendspace, tcp_recvspace:

```
no -a | grep tcp_
```

Increase them dynamically from 16384 to 262144:

```
no -o tcp_sendspace=262144
no -o tcp_recvspace=262144
```

Enable rfc1323 support dynamically (add at the end of */etc/rc.net*):

```
no -o rfc1323=1
```

ISNO NETWORK TUNING

You should check the value of Interface-Specific Network Options (ISNO) by executing the following commands:

```
lsattr -El en0 (and en1, en2, etc)
ifconfig en0 (and en1, en2, etc)
```

If these interface-specific network options are inconsistent with what you set previously, change them:

```
chdev -l en0 -a tcp_sendspace=262144
chdev -l en0 -a tcp_recvspace=262144
chdev -l en0 -a rfc1323=1
ifconfig en0 tcp_sendspace 262144
ifconfig en0 tcp_recvspace=262144
ifconfig en0 rfc1323 1
```

MISCELLANEOUS NETWORK TUNING

Comment out (by using #) the stanza at the end of */etc/rc.net* that turns off extended netstats. We want them left on after the reboot.

Check for Receive Pool Buffer Errors. These are pre-allocated buffers in memory for each adapter:

```
entstat -d ent0 |pg
```

(repeat for ent1, ent2, etc.)

Increase the value from the default of 384 if errors are non-zero:

```
chdev -l ent0 -a rxbuf_pool_size=768 -P
```

then reboot.

Review the pmtu_discover parameter. This is turned on by default and may not be suitable for certain environments.

Check for transmit queue overflows:

```
entstat -d ent0
```

(repeat for ent1, ent2, etc.)

If there have been overflows, or if max packets on s/w transmit queue is close to the transmit queue size (**lsattr -El ent0 | grep tx_que_size**), increase the transmit queue on the adapter:

```
chdev -l ent0 -a tx_que_size=1024 -P then reboot
```

The 10/100 Mbps Ethernet media speed defaults to Auto_Negotiation, and you will probably want to set it explicitly to 100_Full_Duplex:

```
chdev -l ent0 -a media_speed=100_Full_Duplex -P
```

then reboot.

GIGABIT ETHERNET

Check that Gigabit Ethernet is enabled for jumbo frames (make sure that your network equipment ports have been set accordingly!):

```
lsattr -El ent1 |grep jumbo_frames
```

If it is not, enable jumbo frames thus:

```
chdev -l ent0 -a jumbo_frames=yes -P then reboot
```

Set MTU on the Gigabit Ethernet IP interface to 9000:

```
chdev -l en1 -a mtu=9000 -P then reboot
```

tcp_sendspace and tcp_recvspace should be set (general

rule-of-thumb) to 10 times the adapter's MTU size.

Because of the higher speed of gigabit, using a larger tcp_sendspace value results in better performance.

Certain combinations of tcp send and receive space will result in very low throughput (1Mbit or less). To avoid this problem, set the tcp_sendspace to a minimum of three times the MTU size, or equal to or larger than the receiver's tcp_recvspace.

PATCH MANAGEMENT

One of the main chores of system administration is the installation of OS patches.

The first step of the process is to set notification of new patches.

Recommend the categories to which customer sysadmins should subscribe on the AIX notification facility (at http://techsupport.services.ibm.com/server/listserv) so that they are informed of HIPER (high impact, pervasive) fixes as they become available.

Recommend that AIX maintenance be downloaded from the new Support for pSeries products site (at http://www-1.ibm.com/servers/eserver/support/pseries/).

Each patch should be reviewed and its relevance for a particular environment should be determined. For instance, if your site is not utilizing LoadLeveler, there is no need to bother with the installation of patches related to it. I would suggest applying patches at least a week after they have been announced by IBM, simply because some of the PTFs have been known to contain errors as well. If after a week no errors have been reported, the PTF can be applied to one of your systems. At this stage you should perform testing in order to verify that your system functionality and performance have not been affected by patch installation. If everything is OK, you can commit the patch and roll out installation to the rest of your servers. You should use the **compare_report** command to

verify that the levels of installed filsets are at the same level on all your systems.

Twice per year IBM issues an OS maintenance level, which is a collection of PTFs fixed since the previous level was issued. The PTFs contained in the maintenance level are installed and tracked collectively, enabling the establishment of similar levels of operating system across all AIX systems deployed in the organization.

After installing any new optional components, always reapply the AIX maintenance level. When an AIX maintenance level is applied, maintenance is applied only to filesets installed on the system when the maintenance level application occurs. When new optional components are installed, they are installed at the maintenance level available on the base AIX installation media, which is probably below the AIX maintenance level at which the system is currently running.

The version of AIX may be determined as follows:

- **oslevel –rq** – shows a list of known maintenance levels.
- **oslevel -r** – reports the highest maintenance level that has everything installed.
- **oslevel -rg <name>** – shows the filesets that are above the specified maintenance level.
- **instfix -ik <name>_AIX_ML** – indicates whether all filesets are installed for the specified maintenance level.
- **instfix -ciqk <name>_AIX_ML | grep ":=:"** – shows any filesets that are missing from the specified maintenance level.

REFERENCES

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

2 *AIX 5L Workload Manager WLM*, SG24-5977.

3 *Managing AIX Server Farms*, SG24-6606-00.

4 *AIX 5L Differences Guide Version 5.2 Edition*, SG24-5765-02.

5 *AIX Version 4.3 to 5L Migration Guide*, SG24-6924-00.

6 *RS/6000 SP System Performance Tuning Update*, SG24-5340.

7 *Understanding IBM pSeries Performance & Sizing*, SG24-4810.

8 Performance Tuning Guide, http://www.austin.ibm.com/doc_link/en_US/a_doc_lib/aixbman/prftungd/toc.htm.

9 AIX 5 Documentation, http://publibn.boulder.ibm.com/cgi-bin/ds_form?lang=en_US&viewset=AIX.

10 Business Partners Internet Site, http://www.developer.ibm.com/welcome/technical1.html.

11 Public Domain Software Library, http://aixpdslib.seas.ucla.edu/aixpdslib.html.

Alex Polyak
System Engineer
APS (Israel)

More teach me DB2 on AIX!

This month we continue our series of articles looking at DB2 UDB running on AIX and comparing it with DB2 on mainframes.

POT POURRI COMMAND EXAMPLES AND MISCELLANEOUS ITEMS

Here are some relevant comparative command examples between DB2 for OS/390 and DB2 UDB on AIX.

DISPLAY and CANCEL THREAD:

- OS/390:

```
-display thread (*) type(*)
-cancel thread (token id)
```

- UDB (first connect to a specific database):

```
$DB2 "CONNECT TO <database name>"
$DB2 "LIST APPLICATIONS SHOW DETAIL"
$DB2 "LIST DCS APPLICATIONS EXTENDED" | MORE
$DB2 "FORCE APPLICATION <application handle>"
```

DISPLAY DATABASE TS and its restrictions:

- OS/390:

```
-display DATABASE (*) spacenam(*) limit(*) restrict
-display DATABASE (*) spacenam(*) use locks
```

- UDB (first connect to a specific database):

```
$DB2 "CONNECT TO <database name>"
$DB2 "LIST TABLESPACES SHOW DETAIL"
$DB2 "GET SNAPSHOT FOR LOCKS ON <database name>"
/*but before one issues the snapshot command, */
/* one needs to turn on the various monitor switches as follows:*/
$DB2 "UPDATE MONITOR SWITCHES USING BUFFERPOOL ON LOCK ON UOW ON
                                    TABLE ON STATEMENT ON SORT ON"
/* After one get the snapshot, do not forget to turn off the */
/*    switches as in the next example */
$DB2 "RESET MONITOR FOR DATABASE <database name>"
```

DISPLAY and TERM utility:

- OS/390:

```
-display utility(*)
-term utility (utility id)
```

- UDB (first connect to a specific database):

```
$DB2 "CONNECT TO <database name>"
$DB2 "LIST TABLESPACES SHOW DETAIL"
$DB2 "FORCE APPLICATION <application handle>"
/*see also my section on utilities for UDB for */
/* terminating a UDB load utility*/
```

Display buffer pools:

- OS/390:

```
-display BUFFERPOOL (*) LIST(*) DBNAME(*)
```

- UDB (first connect to a specific database):

```
$DB2 "GET SNAPSHOT FOR BUFFERPOOL ON <database name>"
/*but before one issues the snapshot command; one needs to turn */
/*on the various monitor switches as follows:*/
$DB2 "UPDATE MONITOR SWITCHES USING BUFFERPOOL ON
                    LOCK ON UOW ON TABLE ON STATEMENT ON SORT ON"
/* After one gets the snapshot, do not forget to turn off the */
/* switches by the following command */
$DB2 "RESET MONITOR FOR DATABASE <database name>"
/* Remember that turning on or reseting the monitor switches for */
/* a database is a view per user. Each user can turn on the   */
/* monitors and reset them without affecting the other users. */
/*If one wants to affect all users, one needs to turn on the  */
/* monitors by updating them in the dbm cfg file.*/
```

DISPLAY DDF:

- OS/390:

```
-display DDF
```

- UDB:

  There is no equivalent to DDF in UDB. Rough equivalence may be DB2 Connect …! Basically make sure that the DB2 instance of DB2 Connect is up by using the following command:

```
$ps -ef|grep db2
```

  Look for a process called DB2SYSC. This is the heart engine of the instance. If this process is up, then DB2 Connect is up and running; otherwise issue:

```
$DB2 "DB2START"
```

  or you can issue:

```
$DB2 "DB2STOP"
```

  then:

```
$DB2 "DB2START"
```

  One may need to log on using the instance owner id to be

able to issue these commands.

Connecting to a mainframe from DB2 CONNECT on AIX:

```
$DB2 "CONNECT TO <db2 alias> USER xxxxx USING yyyyy"
```

where *xxxxx* is your mainframe id and *yyyyy* is your mainframe password.

```
Select * FROM SYSIBM.SYSTABLES;
```

Help for syntax in AIX:

```
$ ? <cmd>
```

Example:

```
$ ? describe
```

The **man** Unix command is a syntax help facility:

```
$man date
$man passwd
$man who
$man crontab
```

CA7 scheduling product:

- CA7 is a mainframe product from CA that allows the user to control batch mainframe jobs.
- The equivalent of CA7 on AIX is the crontab file. Every executable name entry in this crontab file is preceded by five positions informing AIX when to execute the executable program. One can edit this crontab file as follows:

  ```
  $crontab -e /*one needs to be root to edit it*/
  ```

  For both products see the relevant manuals.

DSNZPARMS on OS/390 versus DB CFG file versus DBM CFG file:

- DSNZPARM is a DB2 OS/390 module created by assembling source code and linking it into SDSNEXIT. It contains major control parameters for the DB2 subsystem.
- The source code is in the DSNTIJUZ member created by the installation CLIST.

- The source code can be seen by browsing DSNTIJUZ, but a more accurate view, reflecting the current settings, is done by executing IBM package DSN8ED7. This package actually calls a stored procedure to dump the current DSNZPARM values.
- The equivalent of the DSNZPARM in UDB is the dbm cfg file. It can be browsed by the following command:

  ```
  $DB2 "GET DBM CFG"|MORE
  ```

  One can update this file with the following command:

  ```
  $DB2 "UPDATE DBM CFG USING <name of key value>"
  ```
- There is yet another cfg file that is equivalent to the DSNZPARM of OS/390 and the dbm cfg on AIX. It is a cfg file for a particular database.
- One can browse its values using this command:

  ```
  $DB2 "GET DB CFG FOR DATABASE <database name>"|more
  ```

  One can update this db cfg file with the following command:

  ```
  $DB2 "UPDATE DB CFG FOR <database name> USING <name_of_ key value>"
  ```

MVS log and DSNMSTR log versus DB2DIAG.LOG file:

- For problem and troubleshooting in OS/390 one needs to look in the MVS log or DSNMSTR log.
- In UDB, however, one needs to look in the DB2DIAG.LOG file. This file can be found in the DIAGPATH parameter in the dbm cfg file. To see the value of the path issue:

  ```
  $db2 "get dbm cfg" | more
  ```

  Once one finds the DB2DIAG.LOG file one can **vi** it. Here is an example of that:

  ```
  /sys2/home/db2devp1/sqllib/db2dump/$vi  db2diag.log
  ```
- Here are useful **vi** commands that could be used once you are in **vi** mode:

  ```
  :$ <enter> means go to last line in file.
  ```

This is nice because the error is appended to the db2diag.log at the bottom:

- :1 <enter> means go to first line. No need to do that, but just in case.
- :set nu <enter> means show me the line numbers (I do not know the command to 'unshow' the line numbers but :q! will get rid of them at the end).
- Ctrl +b <enter> means move the display up one screen. This is nice because you will be going one page at a time looking for the latest errors.
- Ctrl +f <enter> means move the display down one screen.
- :q! <enter> means quit and no write, ie no save.

RENEGING ON MY PROMISE

I know I have promised that datasharing on the mainframe and EEE UDB partitioning are outside the scope of this article. But I could not resist the temptation of discussing at least a couple of items. Please forgive me.

OS/390:

- It is a mistake to think that UDB EEE (which is now called Enterprise Server Edition (ESE) with Database Partitioning Feature (DPF)) is implementing the datasharing concept of OS/390.
- It appears at face value that UDB EEE is implementing datasharing like OS/390. In fact it is not. The confusion stems from the similar benefits each solution can offer, such as scalability and parallelism.
- Datasharing in an OS/390 environment allows DB2 applications to run on one or more DB2 subsystems that are participating in Parallel Sysplex. These DB2 subsystems reside on different OS/390 images.

- These OS/390 images, which are called members in the Sysplex system, communicate and cooperate with each other via separate and specialized hardware called a Coupling Facility and an independent Sysplex timer.
- In OS/390 datasharing, one must have shared DASD.
- All DB2 members in the Syplex share the same DB2 directory and catalog. There is no individual catalog and directory for every DB2 subsystem member.

  However, every member should have its own logs, BSDS, IRLM, temporary database, and work files database (by the way only one DB2 subsystem member can call its work file database DSNDB07; the rest have to give it a different name).
- The main purpose of the Sysplex is datasharing, parallelism, scalability, and availability.

UDB:

- In UDB EEE, on the other hand, we have one instance, not many instances. We can have one or more databases in an instance. A database has TSs. A TS can span several machines (called nodes). Consequently the table that resides in this spanned TS will be partitioned, ie portions of this table will be stored in one machine and other portions will be stored in other machines.
- In contrast, a partitioned TS in OS/390 contains portions of only one table and all these partitions have to be on one machine.
- The collection of nodes (machines) that a partitioned TS/ table resides on is called a nodegroup. When a partitioned TS is defined via DDL, one specifies in which nodegroup one wants this TS to be spread.
- Note that the catalog TS of our particular database and all its tables cannot be spanned over nodes. The catalog TS and all its tables must reside in one machine (node).

- Remember every instance should have a TCP port reserved for it in the */etc/services* file. However, in the case of UDB EEE, one port is needed for every database partition to be in the */etc/services* file.
- Here are some important cfg files for EEE:
  - DBM cfg file, one per instance.
  - DB2_NODES.cfg, one per instance. This tells us how many hosts are in the nodegroup.
  - .rhosts and RAHHOSTFILE. These two files allow users to communicate across nodes without passwords.
  - DB.cfg file, one per partition.
  - every partition has its own log file.
- Here are a couple of commands that are specifically used and useful in an EEE environment:
  - the **rah** command, which is used to send a command to all physical nodes (host names) that are mentioned in the DB2_NODES.cfg file.
  - the **db2_all** command, which is used to send a command to all database partitions listed in the DB2_NODES.cfg.
- Just a quick point on partitioning:
  - in OS/390 we partition by the partitioning key value in each record. So all the records that fall within this key range values will be in one OS/390 partition.
  - in UDB, on the other hand, the rows are not placed in a partition because their partitioning keys fall within a certain range. In UDB the value of the partitioning key column is put through a hashing routine and the result coming from the hashing routine will be the partition number where that row goes.

November 2001 – October 2004 index

Items below are references to articles that have appeared in *AIX Update* since issue 73, November 2001. References show the issue number followed by the page number(s). Subscribers can download copies of all issues in Acrobat PDF format from Xephon's Web site.

BMC Software has announced SmartDBA Recovery Management, which, it claims, is the first policy-based database back-up and recovery solution. The product extends the functionality of BMC's back-up and recovery solution, SQL-BackTrack, to include policy-based and centralized management.

SmartDBA Recovery Management is compatible with AIX, HP HP-UX, Sun Solaris, Linux (Red Hat and SuSE), and Windows operating systems, and currently provides database support for Oracle and Sybase. DB2 UDB and Microsoft SQL Server support are in development.

SmartDBA Recovery Management enables the planning, monitoring, and management of database back-up and recovery across the enterprise from a centralized console.

For further information contact:
BMC Software, 2101 City West Blvd, Houston, TX 77042, USA.
Tel: (713) 918 8800.
URL: http://www.bmc.com/corporate/nr2004/081604_1.html.

* * *

SAS Institute has updated Enterprise Miner 5.1 for SAS9, which is its data-mining product. Enterprise Miner now extends deeper into predictive analytics with new methods and improvements, including tools that accelerate the building of neural networks. The new version uses multithreaded execution to distribute its work over multiple processors and to perform simultaneous calculations on multiple models in the client.

As well as AIX 5.1, the client runs on Windows NT 4 Workstation, 2000 Professional, XP Professional; HP-UX 11i; Solaris 8 or 9. The server runs on AIX 5.1 and Windows NT 4, 2000, 2003 Server; HP-UX 11i; Linux for Intel; Red Hat Linux 8.0 and Advanced Server 2.1; SuSE Linux, Enterprise Server 8; Solaris 8 or 9; Tru64 Unix 5.1A or 5.1B.

For further information contact:
SAS Institute, 100 SAS Campus Drive, Cary, NC 27513-2414, USA.
Tel: (919) 677 8000.
URL: http://www.sas.com/technologies/analytics/datamining/miner.

* * *

Micro Focus has announced that its 'Lift and Shift' approach for migrating legacy applications from older mainframes to contemporary platforms is now available for the IBM eServer zSeries, xSeries, and pSeries lines. Micro Focus Enterprise Server allows customers to take legacy applications and move them to the latest eServer platforms running AIX or Linux, without the risks or costs associated with rewriting entire mission-critical applications.

With the mainframe transaction option within Enterprise Server, Micro Focus and its partners, they claim, offer the most rapid and lowest-risk way of re-hosting mainframe applications on IBM's latest eServer platforms running AIX and Linux.

For further information contact:
Micro Focus, 9420 Key West Avenue, Rockville, MD 20850, USA.
Tel: (301) 838 5000.
URL: http://www.microfocus.com/press/releases/20040817.asp.