120

AIX update

October 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.

Migrating TTY and printer devices between AIX systems

As promised in a previous edition of *AIX Update* (see ‘Migrating print queues between AIX systems’ in issue 118), the following article will describe a procedure that allows you to move local TTY and printer devices from one AIX system to another. Local TTY and printer devices are those connected directly to a serial or parallel port on the server. Use this unsupported procedure at your own risk – and make sure you have a proper **mksysb** back-up before starting. This procedure has been used since the early days of AIX 3.2, when local TTY and printer devices were widely used, and it is still valid on the latest releases of AIX 5L.

The install scripts use an undocumented program called */usr/lpp/bosinst/rda*, which is part of the *bos.rte.bosinst* fileset. This program is used within the preservation install process to preserve the local device definitions, but it could be adapted for use outside the preservation install process. Basically what **rda** does is compare an old ODM with the current one and create a shell script consisting of a group of **mkdev** and **chdev** commands to create the devices.

Below is the usage information for **rda**:

```
rda -a attr_file_name -c prefix -d dvc_file_name -p object path -s
scriptname  -eRrx
```

where:

- -a specifies the name of the output attribute stanza file. The default is */etc/objrepos/old_ca_stanzas*.
- -c – the prefix to search for in the old CuDv. The default is ‘*’.
- -d specifies the name of the output device stanza file. The default is */etc/objrepos/old_cd_stanzas*.
- -p specifies the directory pathname containing the old

object classes. The default is */etc/objrepos*.

- -e – only extract from the old database and do not recover device or configuration information, and do not execute the recovery script.
- -R – relax the criteria for determining which device to configure. If this flag is set, the location code will not be checked.
- -r – only recover device and configuration information from the stanza files. Do not extract information from the old database and do not execute the recovery script.
- -s – the name of the shell script file to use for preserving data.
- -x – execute only the script containing recovery config methods.

The following is the procedure step by step:

1 Copy the ODM customized device configuration files on the source system. The files are:

```
/etc/objrepos/CuAt
/etc/objrepos/CuDep
/etc/objrepos/CuDvDr
/etc/objrepos/CuDv
```

2 Make sure you perform a system back-up (**mksysb**) on the destination system.

3 Once you have booted the destination system with a minimal configuration, restore the ODM customized device configuration files from your previous source system into a temporary directory, eg */tmp/odm_old*. Place the following files in this directory:

```
CuAt
CuDep
CuDvDr
CuDv
```

4 Save the current ODM database on the destination system:

```
rm -rf /tmp/odm.save
mkdir /tmp/odm.save
cp /etc/objrepos/C* /tmp/odm.save
```

5 Get the configuration stanzas from the current database by using the **rda** command:

```
/usr/lpp/bosinst/rda -e -p /tmp/odm_old -d /tmp/odm_old/CuDv.save \
  -a /tmp/odm_old/CuAt.save
```

6 Execute the **rda** command to compare the old ODM files with your current device configuration, recreate the necessary devices and re-apply the necessary attributes. Run it twice to ensure all parent devices are configured before you attempt to configure the child devices:

```
/usr/lpp/bosinst/rda -R -d /tmp/odm_old/CuDv.save \
-a /tmp/odm_old/CuAt.save -s /tmp/conf1
/usr/lpp/bosinst/rda -R -d /tmp/odm_old/CuDv.save \
-a /tmp/odm_old/CuAt.save -s /tmp/conf2
```

You may receive error messages during the last two **rda** executions. They can usually be ignored because the system is trying to configure devices whose parent device or adapter does not exist. If you look at the files */tmp/conf1* and */tmp/conf2*, you will see all the configuration commands used to create and change the customized devices.

7 Shut down and reboot:

```
shutdown -Fr
```

Note that this process must be used only for adding customized ODM data to a currently uncustomized base operating system. The procedure is best used immediately after a fresh install because there will be no customized ODM configuration data on the destination system. If a customized set of ODM files is used on the destination system, the system may fail to boot properly afterwards. Of course, you always have a saved copy of the old ODM database on the destination system in the */tmp/odm.save* directory, and you have the **mksysb** back-up that you can restore to fix the problem.

Finally, after moving the TTY devices, they are ready for use. However, the local printers require the print queues and the virtual printers' configuration to be migrated from the source system. The procedure to migrate the print queues from one system to another can be found in 'Migrating print queues between AIX systems' in *AIX Update*, issue 118.

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

Solve problems during data migration from SSA to SAN storage

SAN storage systems are more flexible than SSA subsystems. Data can be migrated from SSA to a SAN storage system in several ways. To migrate data residing on an old machine with SSA disks to a new machine with SAN storage, probably the fastest approach is the following:

- Create back-ups of all systems involved.
- Define SAN storage on the new system so it's exactly the same size as the old SSA system. Do not create volume groups or logical volumes. The new system should see only a new hdisk.
- Save the old SSA-based volume group using */usr/bin/savevg* – read the man pages for the correct syntax in your environment. Store the saved volume group on tape or in a temporary filesystem.
- Mount the tape or the temporary filesystem on the new system. Then use */usr/bin/restvg* to restore the volume groups along with all the logical volumes and data on the newly-defined SAN disk.

If the SSA subsystem is now disconnected from the old machine without the old volume group and logical volume information being deleted, your system still knows about the old volume group names and logical volume names, but the data can no longer be accessed. Even worse, the old volume group names cannot be deleted using smitty.

The following script helps get rid of exactly such information – old volume group and logical volume name entries in the ODM. Smitty will then be able to reuse the old names for volume groups and logical volumes.

```
#!/bin/ksh
# This script deletes persistent logical volume and volume group
# information from the ODM.
# This script is known to work on AIX 5.1 ML Ø8 and has been tested
# on AIX 5.2 ML Ø6.
# Use this script only in case of emergency. An emergency
# is any of the following:
# 1. A hard drive was removed from an SSA subsystem without
#    properly unconfiguring it.
# 2. A savevg and subsequent restvg didn't work properly
#    because the disk image is broken. Some information is left
#    on the system and smitty refuses to use the same volume group
#    name for creating a new volume group.
# 3. smitty refuses to delete, change, or create a volume group
#    with a certain name, but the volume group name doesn't
#    exist on the system anymore. There is old information in
#    the ODM that cannot be removed using smitty.
# The ODM will be saved before any changes are performed.
# Run this script only when you understand the implications.
# Before you run the script set the following two variables:

# vVg is the volume group name to be deleted from the ODM
# Work on only one volume group at a time!
vVg=fnvg

# vLVname is the logical volume name to be deleted from the ODM
# The variable can consist of just one entry or several blank
# separated entries.
vLVnames="lvfn lvfn_db lvfnora fn_sec_dbØ"

# This is just a date string
vDate=$(date +"%d.%m.%Y.%H.%M.%S")

vLog="/tmp/odmdel.log-${vDate}"
```

```
vODMBackupDir="/etc/objrepos/"

clear
print "CAUTION:"
print "This script can cause potential damage to the following logical
volumes:"
print "\t${vLVnames} \nand the following volume group:\n\t${vVg}"
print "by deleting all information from the ODM."
print "\nType\n\tYes\nand hit the enter key if you want to go ahead."
read vIknow

if [ "${vIknow}X" = "YesX" ]
then
        print "OK, going ahead.  ${vLVnames} and ${vVG} will be
permanently removed from the ODM"
else
        print "Did not make any changes. Exiting."
        exit
fi

# Before editing the ODM create a backup.

cp /etc/objrepos/CuAt ${vODMBackupDir}/CuAt.${vDate} >${vLog}
cp /etc/objrepos/CuDep ${vODMBackupDir}/CuDep.${vDate} >>${vLog}
cp /etc/objrepos/CuDv ${vODMBackupDir}/CuDv.${vDate} >>${vLog}
cp /etc/objrepos/CuDvDr ${vODMBackupDir}/CuDvDr.${vDate} >>${vLog}

# Loop over all logical volumes and delete them in the ODM
for vLVname in ${vLVnames}
        do
             odmdelete -q name=${vLVname} -o CuDv >>${vLog}  2>&1
            odmdelete -q dependency = ${vLVname} -o CuDep >>${vLog} 2>&1
             odmdelete -q name=${vLVname} -o CuAt >>${vLog} 2>&1
             odmdelete -q value3=${vLVname} -o CuDvDr >>${vLog} 2>&1
        done

# Delete the volume group name from the ODM
             odmdelete -q name=$vVg -o CuDep  >>${vLog} 2>&1
             odmdelete -q name=$vVg -o CuAt  >>${vLog} 2>&1
             odmdelete -q parent=$vVg -o CuDv  >>${vLog} 2>&1
             odmdelete -q dependency=$vVg -o CuDep >>${vLog} 2>&1
             odmdelete -q name=$vVg -o CuDv  >>${vLog} 2>&1

        if [ "$vVg" = "rootvg" ]
        then
             odmdelete -q "value1 = 10" -o CuDvDr >>${vLog} 2>&1
        else
             odmdelete -q "value1 = $vVg" -o CuDvDr  >>${vLog} 2>&1
        fi
        odmdelete -q value3=$vVg -o CuDvDr savebase >>${vLog} 2>&1

print "\n\nPlease have a look at the following log file: ${vLog}"
```

If something goes badly wrong with the script, look at the log file and see what the file names of the saved ODM are.

You can restore the ODM to its previous state by copying the four files back to their original names, for example:

```
> cd /etc/objrepos
> cp CuAt.29.11.2005.11.00.35 CuAt
> cp CuDep.29.11.2005.11.00.35 CuDep
> cp CuDv.29.11.2005.11.00.35 CuDv
> cp CuDvDr.29.11.2005.11.00.35 CuDvDr
```

Robert Kaiser
System Analyst
Bayerischer Rundfunk (Germany)

Synchronizing servers using rsync over secure shell

INTRODUCTION

There are several ways to synchronize the directories on two servers. When using tools such as **rcp**, **tar**, or **cpio** it can be tedious to keep an exact copy of the source server's directory on the destination server. When I want to synchronize multiple servers, I use a very useful utility called **rsync**. Recent versions of **rsync** may be run over a secure channel such as **ssh**, and if you are using private/public key combinations **rsync** may even run passwordless.

Rsync is a program that behaves in much the same way that **rcp** does, but has many more options and uses the **rsync** remote-update protocol to greatly speed up file transfers when the destination file already exists. The **rsync** remote-update protocol allows **rsync** to transfer just the differences between two sets of files across the network connection, using an efficient checksum-search algorithm.

For remote transfers, **rsync** may use **ssh** for its communications. In this case, you would need to have **rsync** and **ssh** installed on both the source and destination servers.

Because **rsync** has a command line interface, it is very easy to integrate into scripts.

In this article, I will describe how to install and use **rsync** to synchronize two Web servers running AIX 5L. A sample script is included to simplify the task.

DOWNLOADING RSYNC

Rsync can be found and downloaded from IBM's AIX Toolbox Download page at http://www-1.ibm.com/servers/aix/products/aixos/linux/download.html.

All software found on IBM's AIX Toolbox Download page is in RPM (Redhat Package Manager) format. So, you will need to have the *rpm.rte* LPP installed before you can install the RPMs. The *rpm.rte* AIX installp format can be found at ftp://ftp.software.ibm.com/aix/freeSoftware/aixtoolbox/INSTALLP/ppc/rpm.rte.

You will need the following RPMs to use **rsync**:

- popt-1.7-1.aix4.3.ppc.rpm.
- rsync-2.6.2-1.aix5.1.ppc.rpm.

INSTALLING RSYNC

To install the RPMs you have just downloaded, simply type these commands:

```
# rpm -i popt-1.7-1.aix4.3.ppc.rpm
# rpm -i rsync-2.6.2-1.aix5.1.ppc.rpm
```

DOWNLOADING OPENSSH

OpenSSH can be found on the Bull Freeware site at http://www.bullfreeware.com.

INSTALLING OPENSSH

OpenSSH is the open-source version of secure shell. You can download it from Bull Freeware for any version of AIX.

To install the package from Bull Freeware, which comes in *exe* format, simply follow the instructions found on Bull's site at http://www.bullfreeware.com/install_down.html.

Once the package is installed, you will have to generate the server's keys using the command:

```
# ssh-keygen -t rsa1 -f /usr/local/ssh/etc/ssh_host_key -N  ""
# ssh-keygen -t rsa -f /usr/local/ssh/etc/ssh_host_rsa_key -N  ""
# ssh-keygen -t dsa -f /usr/local/ssh/etc/ssh_host_dsa_key -N  ""
```

You may have to change the path where your server's keys will be stored (ie */usr/local/ssh/etc*).

SETTING UP PASSWORDLESS SSH

In order to set up passwordless **ssh**, you must first create a user on both your servers that will be used to execute **rsync**. You may optionally use an existing user or even the root account (at your risk).

For my example, I chose to create a special user 'webuser' and group 'webgroup' to execute **rsync**.

First, create a new AIX group that will contain only the passwordless SSH user, webuser. I have chosen to name the group webgroup with a gid of 522:

```
    # mkgroup -'A' id='522' webgroup
```

Next, create the user webuser with a uid of 673 and a shell of ksh:

```
# mkuser id='673' pgrp='webgroup' groups='webgroup' shell='/usr/bin/ksh'
home='/home/webuser' \
gecos='web site user' webuser
```

Note: do not give this user a password in AIX. Authentication will be done using SSH.

In order to allow the user webuser to log in to all servers without a password, you must generate the private/public key pair.

First, as root, change user to webuser and in the home directory type:

```
# su - webuser
# ssh-keygen -t dsa -f .ssh/id_dsa
```

A password will be asked for – do not enter a password, just press *Enter*.

Now, go the *.ssh* directory, and you will find two new files – *id_dsa* and *id_dsa.pub*.

The second one is the public key. Rename it thus:

```
# cd .ssh
# mv id_dsa.pub authorized_keys2
# chmod 600 authorized_keys2
```

Now, try to log in to the same server using **ssh** to verify that things have worked.

Note 1: the first time you log in to a server using **ssh**, even though it is passwordless, it will ask you if you want to permanently add that host to the list of known hosts. You must answer ‘yes’.

The next time you log in to the remote server, no password will be requested or questions asked.

Note 2: this system will work as long as none of the machines changes its IP address. If this server changes its IP address, you will have to regenerate the public/private keys. If any of the other servers change their IP address, you will have to manually delete them from the *known_hosts* file in the *.ssh* directory and answer ‘yes’ to permanently adding that host to the list of known hosts again.

You will have to run this procedure on both the source and destination server. However, you will not need to recreate the private/public key pair on every server.

After you have created the group and user on the other server, simply copy the file from the first server to the new server:

```
     /home/webuser/.ssh/authorized_keys
```

Now, change the permissions for the local webuser account thus:

```
# chown -R webuser:webgroup /home/webuser
# chmod -R 700 /home/webuser
# chmod 600 /home/webuser/.ssh/authorized_keys
```

Once again, try to log in to this new server using **ssh** to verify that things have worked correctly.

SAMPLE SCRIPT USING RSYNC

Here is a sample Korn shell script that I use to synchronize two Web servers. The source server is a development box that is mirrored onto the production Web server.

```
#!/bin/ksh
#*************************************************************************
# Script: rsync.sh
# By    : Elvio Prattico
# Date  : August 1, 2005
#
# This script synchronizes a directory on a source server to a
# destination server using rsync over secure shell (ssh).
#
#*************************************************************************

# Change the values of these variables to match your environment

BASENAME='/usr/bin/basename $0'                # The name of this script
HOSTNAME='/usr/bin/hostname'                   # The name of this server
RSYNC=/usr/bin/rsync                     # Full path to rsync executable
SRC="aixdev"                                    # Name of source server
DEST="aixprod"                              # Name of destination server
SRCDIR="/dev/www/"                       # Full path of source directory
DESTDIR="/prod/www/"                # Full path to destination directory

# DO NOT CHANGE ANYTHING AFTER THIS UNLESS YOU KNOW WHAT YOU ARE DOING!!
```

```
OPT="-ave ssh –delete –delete-after"          # rsync options
OPT_TEST="-nave ssh –delete –delete-after"
                                          # rsync options in test mode


#***********************************************
# Verify rsync is installed and available
#***********************************************
if [ ! -x $RSYNC ]
then
        echo ""
        echo "   ERROR : rsync is not installed or executable."
        echo ""
        exit 1
fi

#***********************************************
# Verify we are on the source server
#***********************************************
if [ $HOSTNAME != $SRC ];
then
        echo ""
        echo "   ERROR : You must run this script on the source server."
        echo "           SRC variable configured = $SRC"
        echo ""
        exit 1
fi

#***********************************************
#
# Main
#
#***********************************************

if [[ $1 = "test" ]];
then
        echo "\n\n*** Running $BASENAME in TEST mode ****\n\n"
        $RSYNC $OPT_TEST $SRCDIR $DEST:$DESTDIR
        exit Ø
else
        # Verify we are not already running

      RUNNING='ps -ef|grep $BASENAME|grep -v grep|wc -l|sed -e 's/ //g''

        if [ $RUNNING -ge 2 ]
        then
         echo "\nERROR : the script $BASENAME is currently running...\n"
                ps -ef|grep $BASENAME|grep -v grep
                echo ""
                exit 1
        fi

        $RSYNC $OPT $SRCDIR $DEST:$DESTDIR

        if [[ $? -ne Ø ]];
        then
                echo "\n\n***** An error occurred during
synchronization. Read messages above carefully !!! *****\n\n"
                exit 1
        fi

        exit Ø
fi
```

Note: you will have to change the values of the variables at the top of the script to match your environment.

RUNNING THE SAMPLE SCRIPT

To run the sample script in test mode (nothing is actually done):

```
# su - webuser
# rsync.sh test
```

To run the script:

```
# su - webuser
# rsync.sh
```

Verify the output to ensure the transfer completed without any errors.

MORE INFORMATION ON RSYNC

I have only scratched the surface of what you can do with **rsync**. To get more information on it, visit http://rsync.samba.org.

To get a listing of all the **rsync**’s options, type:

```
# rsync —help
```

Elvio Prattico
Consultant
PRATTICO Consulting (Canada)

AIX and DB2 tuning essentials and best practices for DB2 performance

INTRODUCTION

Performance problems are either related to a system resource issue or an application issue (or both). When it comes to system resource usage, AIX provides performance-tuning parameters to achieve optimal resource usage for your business objectives. From an application perspective, DB2 is considered middle tier, yet at the same time runs within the application process space. Like AIX, it can monitor performance from both an application and a system perspective, providing a wealth of tuning data. This article helps you analyse and tune performance from both an AIX and a DB2 perspective.

Performance tuning is a vast topic and so I have divided the article into two parts. In the first part we'll talk about performance issues and methods of tuning AIX and DB2. We'll discuss the important configuration parameters, which you need to adjust from their initial settings to better support your application. I will discuss the most important configuration parameters that will bring major improvement to CPU, memory, and I/O performance. There are many best practices for creating buffer pools, table spaces, tables, and indexes so we'll take a look at those as well.

In the second part of the article we'll cover tuning based on monitor output in detail. I'll show you how to use snapshot monitoring to help tune your SQL, buffer pools, and various database manager and database configuration parameters. 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.

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

PERFORMANCE FUNDAMENTALS

Performance is vital to the success of your on-demand applications, is the way a computer system behaves given a particular workload and is measured in terms of system response time, throughput, and availability. Performance is affected by:

- The resources available in your system.
- How well those resources are used and shared.

In general, you tune your system to improve its cost-benefit ratio. Specific goals could include:

- Processing a larger, or more demanding, workload without increasing processing costs – for example increasing the workload without buying new hardware or using more processor time.
- Obtaining faster system response times, or higher throughput, without increasing processing costs.
- Reducing processing costs without degrading service to your users.

DB2 UDB environments range from stand-alone systems to complex combinations of database servers and clients. In any type of system, for successful applications the common key is performance. When you plan, design, and build your database system, you need to understand several considerations about logical and physical database design, application design, and the configuration parameters of DB2, so that your database system can meet the performance requirements of your application.

While its performance may initially be good, as time goes on, your system may need to serve more users, store more data, and process more complex queries. Consequently, the increased load level of your database server will affect its performance. This could be the time to upgrade to more powerful equipment. However, before investing in equipment, you may be able to improve performance by simply tuning the database and the operating system.

CAUSES OF PERFORMANCE PROBLEMS

There are many possible causes of performance problems. The most frequently encountered problems are:

- Poor application design.

  In many cases performance problems stem from poor application design and inefficient programs. The database itself may not have any problems at all. For example, SQL statements written inappropriately can degrade overall performance, even though your database is well designed and tuned.

- Poor system and database design.

  Poor design of your system and/or databases can also be a reason for performance problems. Inefficient disk storage layout or failing to consider performance when designing and implementing your database will certainly degrade performance, and can be very difficult to fix once your production system has been started.

- System resource shortages.

  System resource shortages can cause bottlenecks in your database system:

  – CPU – too many users, or running too many applications on a CPU, may cause system degradation.

  – Memory – every process will use some physical memory. If you have insufficient memory, you may find that applications will fail, or your system will start thrashing. From an operating system perspective, excessive paging generally indicates an insufficient amount of memory in the system. For most cases the easiest fix is to add more memory.

- – Disk I/O – I/O performance can play an important part in a database system. Too much activity on a single disk or I/O bus may cause performance problems. As far as poor disk I/O goes, there are several things you can do within AIX to improve performance, which is what makes AIX unique compared with other operating systems. The I/O settings you are able to adjust within AIX for performance will complement the performance gains you will be able to obtain from DB2.

GENERAL TUNING RULES

So, where should a DBA start? From a database perspective, performance problems typically arise from deficiencies in one or more of the following:

- System (environment) configuration
- Instance configuration
- Database configuration
- Database design
- Application design.

Here are the top ten things you can do to get the most performance out of your database. Usually, you will find that about 90% of maximum performance is achieved using about 10% of possible configuration changes.

1 Ensure that you have enough disks (6–10 per CPU is a good start). Each table space's containers should span all available disks. Some table spaces, such as SYSCATSPACE and those with a small number of tables, do not need to be spread across all disks, while those with large usage or temporary tables should be.

2 Buffer pools should make use of about 75% (OLTP) or 50% (OLAP) of available memory.

3 Runstats should be performed on all tables, including the system catalog tables.

4 Use the Design Advisor to recommend and review indexes for SQL workloads.

5 Use the Configuration Advisor to configure the database manager and database for your application environment.

6 Logging should occur on a separate high-speed disk, identified by the NEWLOGPATH database configuration parameter.

7 Committing can often increase concurrency.

8 SORTHEAP should be increased to avoid sort overflows.

9 The table space type should be SMS for the system catalog table space, and temporary table spaces and DMS raw (device) or file for the rest. Run db2empfa to enable multi-page file allocation for the SMS table spaces; this will allow SMS table spaces to grow one extent at a time (instead of one page), which can speed up heavy insert operations and sorts that spill to disk.

10 Use parameter markers for repeated statements.

AIX PERFORMANCE

One of the most important decisions a database administrator has to make when designing a database is how the data will be laid out across physical partitions. Overlooking this important step can drastically affect I/O performance. From an operating system layer, DB2 is only as good as the filesystem on which it runs. In AIX, the Logical Volume Manager (LVM) controls that filesystem. LVM is a subsystem within AIX that controls disk resources and provides a logical mapping between storage and physical disks. To achieve the most from LVM in I/O performance, it is important that you properly plan your database layout across physical disks.

First we will discuss some recommended filesystem layouts and accompanying tuning parameters that will help you get maximum performance from LVM and DB2. Then we will discuss various AIX commands for monitoring CPU, memory, and I/O activity.

Operating system configuration parameters

The following settings are recommended to allow maximum resource usage of AIX by DB2:

- Maxuproc (maximum number of processes per user) – set the AIX maxuproc device attribute to 4096. The default value set by AIX (500) has been found to be too low for some large-scale database environments, and causes DB2 to generate an SQL error message 'SQL1402N – Unable to authenticate user due to unexpected system error'. For a DB2 partitioned server environment this value must be set.
- Ulimit and Large file enabled – to create large files on AIX greater than 2GB, 'Large file enabled' must be set for the file system, and the fsize for the DB2 instance owner must be set to unlimited (or –1).
- Striping – also known as RAID 0, striping is a technology that was developed to achieve maximum I/O performance. The basic concept is that data is written and read from the filesystem in chunks across the width of the physical disk layout in parallel. The width represents the number of disks within the layout, and striping represents the contiguous layout of data across separate disks. The following are some general recommendations that would increase performance in a striped layout:
  - max_coalesce – this represents the maximum number of bytes the SSA disk device driver attempts to transfer to and from an SSA logical disk in one operation. Set max_coalesce equal to or a multiple of the stripe unit size, but not smaller. This value is available only for SSA configurations.
  - queue_depth – this represents the maximum number of commands that the SSA disk device driver

dispatched for a single disk drive for an hdisk. For an N+P array, set this value to 2*N or even 3*N. By altering this value, we maximize the chance of reading data from each component of the array in parallel. The syntax for changing this value is the same as max_coalesce.

CPU-related monitoring in AIX

The **sar** command (system activity report) collects, reports, or saves system activity information. The **sar** command can be used in two ways to collect data: one is to view system data in real time, and the other is to view data previously captured. The **sar** command is one of the first performance monitors to be used by an administrator.

Examples:

```
sar -o <filename> 60 10 >/dev/null 2>&1 &
```

will sample the CPU activity every minute for 10 minutes and save the output to a file.

If user+nice+system = 100% then you have a CPU bottleneck.

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.

Example:

```
vmstat 10 500
```

will display system CPU usage every 10 seconds.

If id=0 or us+sy>100 this indicates that the system is CPU-bound.

The breakdown of percentage usage of CPU time can be obtained from the following columns:

- us – user time.
- sy – system time.
- id – CPU idle time.
- wa – CPU cycles to determine that the current process is waiting and there is pending disk input/output.

Memory-related monitoring in AIX

The **svmon** command displays information about the current state of memory. The memory consumption is reported using the *inuse*, *free*, *pin*, *virtual*, and *paging space* counters:

- The *inuse* counter represents the number of used frames.
- The *free* counter represents the number of free frames from all memory pools.
- The *pin* counter represents the number of pinned frames, that is, frames that cannot be swapped.
- The *virtual* counter represents the number of pages allocated in the system virtual space.
- The *paging space* counter represents the number of pages reserved or used on paging spaces.

The **svmon** command generates seven types of report:

- Global
- User
- Process
- Segment
- Detailed segment
- Command
- Workload management class.

To run each of these reports, a report indicator flag needs to be used.

svmon –P will sort all processes by the amount of memory they are consuming.

The *inuse* column indicates the amount of memory being used by each process in 4K pages.

The **lsps** command displays characteristics such as the paging-space name, physical-volume name, volume-group name, size, percentage of the paging space used, whether the space is active or inactive, and whether the paging space is set to automatic.

lsps –s will display the total paging space and the percentage that is used.

The **bootinfo** command determines and displays various boot information, including boot device type and boot device name. This command is not a user-level command and is not supported in AIX Version 4.2 or later.

bootinfo –r (root) or **lsattr –E –l sys0 –a realmem** will display actual physical memory.

Disk-related monitoring in AIX

The **iostat** command is a useful tool providing a first indication of I/O-related performance problems. The **iostat** command is capable of reporting CPU statistics, terminal I/O statistics, and disk I/O statistics, which help identify the I/O load on individual components, such as hard disks. The information from **iostat** reports can be used to modify system configurations to improve I/O load distribution between physical disks. The **iostat** command extracts data from AIX kernel I/O counters in the kernel address space, which are updated at every clock tick (1/100 second) for TTY as well as CPU and I/O subsystem activity.

iostat –d 10 will report statistics for all attached disks every 10 seconds. Look for the disks with the highest *%tm_act* (percentage of time disk was active) and high Kbps.

There are various factors that affect logical volume (LV) performance, for example the allocation position on the disk or the mirroring options. To obtain information about the logical volume, you can use the LVM **lslv** command, which provides information on:

- LV attributes – list of the current logical volume settings.
- LV allocation – placement map of the allocation of blocks on the disk.
- LV fragmentation – fragmentation of the LV blocks.

The **filemon** command monitors and presents trace data on the following four levels of filesystem utilization:

- Logical filesystem – the monitored operations include all read, write, open, and lseek system calls, which may or may not result in actual physical I/O, depending on whether or not the files are already buffered in memory. I/O statistics are kept on a per-file basis.
- Virtual memory system – at this level, operations (that is, paging) between segments and their images on disk are monitored. I/O statistics are kept on a per-segment basis.
- Logical volumes – I/O statistics are kept on a per-logical volume basis.
- Physical volumes – at this level, physical resource utilizations are obtained. I/O statistics are kept on a per-physical volume basis.

The **filemon** command is based on the AIX trace facility to monitor I/O activity during a certain time interval.

DB2 PERFORMANCE

Configuration parameters

DB2 was designed with tuning and configuration parameters that fall into two general categories:

1 Database manager configuration parameters (DBM).

Each instance of the database manager has a set of the database manager configuration parameters (also called database manager parameters). These affect the amount of system resources that will be allocated to a single instance of the database manager.

2 Database configuration parameters (DB).

Each database has a set of the database configuration parameters (also called database parameters). These affect the amount of system resources that will be allocated to that database.

I shall discuss some of the important parameters whose tuning will result in significant performance benefits. I have divided the parameters into groups related to CPU, memory, and I/O. Normally it's not necessary to go through the whole list to achieve your performance goal. You can just try several of them at the top of the list to see whether there's any performance improvement.

CPU-related parameters

Here we discuss the main configuration parameters that affect CPU usage:

- INTRA_PARALLEL (DBM) – this parameter specifies whether the database manager can use intra-partition parallelism. The default of NO is best for high concurrent connections (mainly OLTP), while YES works best for few concurrent connections and more complex SQL (OLAP/DSS). Mixed workloads typically benefit from NO.

  When enabled, it causes sort memory to be allocated from shared memory. Additionally, it can cause excessive system overhead if the level of concurrency increases significantly. If the system is non-OLTP, there is a 4:1 ratio of CPUs to partitions, and the CPU busy rate runs less than 50% on average, INTRA_PARALLEL would likely improve performance.

- MAX_QUERYDEGREE (DBM) – this parameter specifies the maximum degree of intra-partition parallelism that is used for any SQL statement executing on this instance of the database manager. An SQL statement will not use more than this number of parallel operations within a partition when the statement is executed. The *intra_parallel* configuration parameter must be set to YES to enable the database partition to use intra-partition parallelism.

  The default value for this configuration parameter is -1. With this value the system uses the degree of parallelism determined by the optimizer; otherwise, the user-specified value is used.

- DFT_QUERYOPT (DB) – this is used to specify a default level of optimization to use when compiling SQL queries. Try the default of 5 or 3 for mixed OLTP/OLAP, lower for OLTP, higher for OLAP. For simple SELECTS or short run-time queries (which generally take less than one second to complete) 1 and 0 may be appropriate. Try a class of 1 or 2 if you have many tables with many join predicates on the same column. Try using a class of 7 for longer running queries, that take more than 30 seconds to complete, or if you are inserting to a UNION ALL VIEW (added in FixPak4). You should avoid using a class of 9 in most circumstances.

- CPUSPEED (DBM) – the CPU speed (milliseconds per instruction) is used by the SQL optimizer to estimate the cost of performing certain operations. You can explicitly set this value to model a production environment on your test system or to assess the impact of upgrading hardware. By setting it to -1, *cpuspeed* will be re-computed. The CPU speed is used by the optimizer in determining access paths. You should consider rebinding applications (using the **REBIND PACKAGE** command) after changing this parameter.

Memory-related parameters

Many of the configuration parameters available in DB2 affect memory usage on the system. We discuss some of them, which have a high impact on the database performance:

- MAXAGENTS (DBM) – this is the maximum number of database manager agents available to accept application requests for all databases within the instance. This parameter can be useful in memory-constrained environments to limit the total memory usage of the database manager, because each additional agent requires additional memory.
- NUM_INITAGENTS and NUM_POOLAGENTS (DBM).

  NUM_INITAGENTS specifies the number of idle agents that are created in the pool at db2start and helps speed up connections at the beginning of database use.

  NUM_INITAGENTS and NUM_POOLAGENTS should be set to the average number of expected concurrent instance-level connections, which is usually low for OLAP and higher for OLTP. For performance benchmarks where there is a substantial ramp up of connections, set NUM_INITAGENTS to the number of expected connections (this will significantly reduce time required to ramp up connections by reducing resource contention).
- NUMDB (DBM) – this parameter specifies the maximum number of concurrent active databases that different applications can use. Because each database has its own global memory area, the amount of memory that might be allocated increases if you increase the value of this parameter. When you set *numdb* to *automatic*, you can create any number of databases and memory consumption grows accordingly.
- SORTHEAP (DB) – this parameter specifies the maximum number of private memory pages to be used for private sorts or the maximum number of shared memory pages to be used for shared sorts. Each sort has a separate sort heap that is allocated, as needed, by the database manager. It is often well understood that sort overflows

occur when the amount of memory needed for a sort exceeds SORTHEAP. Less well understood is that if your statistics are out of date, or if data is skewed and you have not collected distribution statistics, an overflow can occur if DB2 requests too small a sort heap, and the actual sort operation exceeds the requested amount. Therefore, it is very important to keep statistics up-to-date. Additionally, make sure that the sort is not the result of a missing index. A good starting point is 128 for OLTP and between 4096 – 8192 for OLAP. If there are a lot of 'Sort overflows' (double-digit), you probably need to increase SORTHEAP. If the 'Number of hash join overflows' is not 0, increase SORTHEAP by increments of 256 until it is. If 'Number of small hash join overflows' is not 0, increase SORTHEAP by 10% increments until it is.

- SHEAPTHRES (DBM) – this is the total amount of memory that concurrent private sorts may consume for all databases in the instance. Additional incoming sorts will be given a smaller amount of memory to use.
- CHNGPGS_THRESH (DB) – use this to specify the percentage of changed pages in the buffer pool, which, when reached, causes the asynchronous page cleaners to be started to write the changes to disk so there is room for new data in the buffer pool. Page cleaners are not used in a read-only environment. In OLTP, a value of 20–40 should improve performance (20 in the case of very heavy update activity), because lowering the value makes the I/O cleaners more aggressive in their writing out of dirty buffer pool pages, but with less work each time. If there are not a lot of INSERTs or UPDATEs, the default of 60 should be fine for OLTP and OLAP.
- LOGBUFSZ (DB) – this parameter allows you to specify the amount of the database heap (DBHEAP) to use as a buffer for log records before writing to disk. The log records are written to disk when a transaction commits or the log buffer is full. Buffering the log records will result in

log records being written to disk less frequently with more log records being written each time.

- PKGCACHESZ (DB) – the package cache is used for cacheing sections for static and dynamic SQL statements. Cacheing packages allows the database manager to reduce its internal overhead by eliminating the need to access the system catalogs when reloading a package; or, in the case of dynamic SQL, eliminating the need for recompilation.
- CATALOGCACHE_SZ (DB) – this is used to cache system catalog information, such as SYSTABLE, authorization, and SYSROUTINES information. Cacheing catalog information is very important, especially when DPF is used, where internal overhead is reduced by eliminating the need to access the system catalogs (catalog partition) to obtain information that has been previously retrieved.
- UTIL_HEAP_SZ (DB) – this parameter specifies the maximum amount of memory that can be used simultaneously by the BACKUP, RESTORE, and LOAD utilities. If you are using LOAD, set UTIL_HEAP_SZ to at least 10,000 pages per CPU.
- FCM_NUM_BUFFERS (DBM) – in a partitioned database environment, communication between database partitions is handled by the Fast Communications Manager (FCM). This parameter specifies the number of 4KB buffers that are used for internal communications (messages) both among and within database servers.

  If you are using multiple logical nodes, on non-AIX systems, one pool of *fcm_num_buffers* buffers is shared by all the multiple logical nodes on the same machine, while on AIX:

  – if there is enough room in the general memory that is used by the database manager, the FCM buffer heap will be allocated from there. In this situation, each database partition server will have *fcm_num_buffers*

buffers of its own; the database partition servers will not share a pool of FCM buffers (this was new in DB2 Version 5).

  - if there is not enough room in the general memory that is used by the database manager, the FCM buffer heap will be allocated from a separate memory area (AIX shared memory set) that is shared by all the multiple logical nodes on the same machine. One pool of *fcm_num_buffers* will be shared by all the multiple logical nodes on the same machine. This is the default configuration for all non-AIX platforms.

Disk I/O-related parameters

Hardware components that make up your system can influence its performance. You have to balance the demand for disk I/O across several disk storage devices and controllers. These can affect the speed at which the database manager can retrieve information from disks:

- BUFFPAGE (DB) – each database has one buffer pool, IBMDEFAULTBP, which is created when the database is created, and can have more. All buffer pools reside in global memory, which is available to all applications using the database. If the buffer pools are large enough to keep the required data in memory, less disk activity will occur. Conversely, if the buffer pools are not large enough, the overall performance of the database can be severely curtailed and the database manager can become I/O-bound as a result of the high amount of disk activity (I/O) required to process the data your application requires.

  The buffpage parameter controls the size of a buffer pool when the CREATE BUFFERPOOL or ALTER BUFFERPOOL statement is run with NPAGES -1; otherwise, the buffpage parameter is ignored and the buffer pool will be created with the number of pages specified by the NPAGES parameter.

- NUM_IOCLEANERS (DB) – this parameter allows you to specify the number of asynchronous page cleaners for a database. These page cleaners write changed pages from the buffer pool to disk before a database agent needs the space in the buffer pool. As a result, database agents should not have to wait to use the space in the buffer pool. If the applications for a database primarily consist of transactions that update data, an increase in the number of cleaners will speed up performance. Increasing the page cleaners will also decrease the time for recovery from soft failures, such as power outages, because the contents of the database on disk will be more up to date at any given time.
- NUM_IOSERVERS (DB) – I/O servers are used on behalf of the database agents to perform prefetch I/O and asynchronous I/O by utilities such as **backup** and **restore**. This parameter specifies the number of I/O servers for a database. No more than this number of I/Os for prefetching and utilities can be in progress for a database at any time.

  In order to fully exploit all the I/O devices in the system, a good value to use is generally one or two more than the number of physical devices on which the database resides. It is better to configure additional I/O servers, since there is minimal overhead associated with each one and any that are unused will remain idle.
- LOGPRIMARY, LOGSECOND, and LOGFILSZ (DB).

  LOGPRIMARY specifies the number of primary log files to be pre-allocated, while LOGSECOND are allocated on an as-needed basis. LOGFILSIZ defines the size of each log file.

  If there is a high number of 'Secondary logs allocated currently', you will want to increase LOGFILSIZ or LOGPRIMARY (while making sure that LOGPRIMARY + LOGSECOND does not exceed 256). Log file size has an implication on disaster recovery configurations where log shipping is used. A large log file will have better performance, but potentially increase the degree of lost transactions. When the primary system goes down, the last log file and its transactions may never be sent to the secondary because the file had not been closed before failure. The larger the log file, the greater the degree of lost transactions because of lost log files.

Registry parameters

The following are eleven of the important registry variables from the 214 registry variables available that impact performance:

- DB2_APM_PERFORMANCE – this registry variable specifies whether or not performance-related changes in the Access Plan Manager (APM) that will affect the behaviour of the SQL cache (package cache) are to be made. It also specifies whether the global SQL cache will operate without the use of package locks, which are internal system locks preventing cached package entries from inadvertently being removed. This registry variable should be set to ON only in a non-production environment. When ON, you may see 'Out of package' cache errors, and memory usage may increase. Also PRECOMPILE, BIND, and REBIND operations can't be performed, nor can operations that invalidate packages or make them inoperable.
- DB2_PARALLEL_IO – possible values include * and null (the default) for this registry variable, which specifies whether or not DB2 can use parallel I/O when reading or writing data to and from tablespace containers (even in situations where the tablespace contains only one container).
- DB2_STRIPED_CONTAINERS – this variable is set to ON or null (the default) to specify whether or not the tablespace container ID tag will take up a partial or full RAID disk stripe. When using RAID devices, the tablespace

should be created with an extent size that is equal to, or a multiple of, the RAID stripe size. However, because of the one-page container tag, the extents will not line up with the RAID stripes. It may be necessary to access more physical disks than would be optimal during an I/O request unless this registry variable is set to ON.

- DB2_AVOID_PREFETCH – this specifies whether or not prefetching should be performed during crash recovery. The default is OFF; if set to ON, prefetching is not performed.
- DB2_ENABLE_BUFPD – the default value is OFF for this registry variable, which specifies whether or not DB2 is to use intermediate buffering to improve query performance.
- DB2_EXTENDED_OPTIMIZATION – this specifies whether or not the query optimizer will use optimization extensions to improve query performance; the default is OFF.
- DB2MAXFSCRSEARCH – this variable can be set to -1, or a value from 1 to 33554 in order to specify the number of free-space control records to search when adding a record to a table. It allows you to balance insert speed with space reuse (small values optimize for insert speed, large values optimize for space reuse). If the registry variable is set to -1, the DB2 Database Manager will search all free-space control records. The default value is 5.
- DB2_OVERRIDE_BPF – this registry variable, which can be set to a positive number of 4K pages, specifies the size of the buffer pool (in pages) that will be created at database activation or the first time a connection is established. DB2_OVERRIDE_BPF is useful when failures resulting from memory constraints occur during database activation or the first time a connection is established. Such a memory constraint could arise either because of a real memory shortage (which is rare) or because of an attempt by the DB2 Database Manager to allocate large

inaccurately-configured buffer pools. The default value is null.

- DB2PRIORITIES – the values for this registry variable are platform-dependent. DB2PRIORITIES controls the priorities of DB2 processes and threads.
- DB2_SORT_AFTER_TQ – this specifies how the DB2 optimizer works with directed table queues in a partitioned database environment when the receiving end requires the data to be sorted and the number of receiving nodes is equal to the number of sending nodes. When set to NO (which is the default), the DB2 optimizer tends to sort at the sending end and merge the rows at the receiving end. When set to YES, the optimizer transmits the unsorted rows and, when they have all been received, sorts them at the receiving end.
- DB2_HASH_JOIN – a YES or NO value specifies whether or not a hash join can be used when compiling a data access plan. The default is NO.

DATA STORAGE MANAGEMENT AND DATABASE DESIGN

So far, we have discussed the various configuration parameters and registry variables. The next major consideration, in order to obtain acceptable response times, is the design of bufferpools, tablespaces, tables, and indexes. It is no secret that accessing memory is faster than disk I/O. DB2 uses database buffer pools to attempt to minimize disk I/O. When you design tablespaces and tables, you must choose an appropriate structure, data model, and data type for them. With respect to indexes, you need to be aware of the advantages and disadvantages of creating indexes, and create appropriate ones. DB2 UDB allows the database administrator to choose from a variety of options when creating these objects and it is essential for the database administrator to understand the various advantages and disadvantages of choosing a particular option. In this section, we will explain how each of these options affects database performance, and provide helpful information that a database administrator can use for his design.

Bufferpools

Properly defining buffer pools is one of the major keys to having a well-performing system. Use the following formula for calculating your approximate database global memory usage:

```
Bufferpools + dbheap + util_heap_sz + pkgcachesz + aslheapsz + locklist
+ sheapthres_shr (If INTRA_PARALLEL enabled) + approx 10% overhead
```

Determining how many buffer pools

You will need at least one buffer pool for each page size used by a table space in your database. Usually, the default IBMDEFAULTBP buffer pool is left for the system catalogs. New buffer pools are created to handle different page sizes and behaviours of the table spaces.

With the multiple buffer pool approach, a good starting point would be to use four buffer pools as follows:

- A medium-sized buffer pool for temporary table spaces.
- A large buffer pool for index table space.
- A large buffer pool for table spaces containing frequently-accessed tables.
- A small buffer pool for table spaces containing infrequently-accessed, randomly-accessed, or sequentially-accessed tables

DMS table spaces containing only LOB data can be assigned to any buffer pool because LOBs do not use buffer pool space.

Determining memory allocation for buffer pools

Never allocate more memory than you have available or you will incur costly OS memory paging. Generally speaking, it may be pretty hard to know how much memory initially to allocate to each buffer pool without monitoring.

For OLTP type workloads, 75% of available memory being allocated to the buffer pool(s) is a very good starting point.

For OLAP/DSS, the rule of thumb is to give 50% of your available memory to a single buffer pool (given that only one page size is being used) and the other 50% to the SORTHEAP.

Using block-based buffer pools

OLAP queries that rely heavily on prefetching may benefit from a block-based buffer pool. By default, all buffer pools are page-based, which means that prefetches will place contiguous pages on disk into non-contiguous memory. If a block-based buffer pool is used, DB2 will use block I/Os to read multiple pages into the buffer pool in a single I/O, which significantly improves the performance of sequential prefetching. A block-based buffer pool consists of both the standard page area and a block area. The NUMBLOCKPAGES parameter of the CREATE and ALTER BUFFERPOOL SQL statement is used to define the size of the block memory, while the BLOCKSIZE parameter specifies the size of the blocks, and hence the number of pages to read from a disk in a block I/O.

Table spaces

Since table spaces are assigned a buffer pool, buffer pools are very relevant to performance issues involving table spaces.

Determining the type of table space to create (DMS or SMS)

System Managed Storage (SMS) should be used for system temporary table spaces and the system catalog table space because it allows for the table space to grow and shrink dynamically.

Database Managed Storage (DMS) should be used for all other table spaces. DMS allows for a single table to span multiple table spaces (index, user data, and LOB), which decreases contention between index, user, and LOB data during prefetch and update operations, thereby improving data access times. Using DMS raw may even squeeze an additional 5–10% performance increase.

Determining page size

In order for you to create a table, there must be a table space with a page size large enough to contain a row with minimal waste. You have the option of using 4, 8, 16, or 32KB page sizes. For OLTP applications that perform random update operations, a smaller page size is usually preferable because it consumes less space in the buffer pool. For OLAP applications that access large numbers of consecutive rows at a time, a larger page size is usually better because it reduces the number of I/O requests that are required to read a specific number of rows. Larger page sizes also allow you to reduce the number of levels in the index by keeping more row pointers on a single page.

Determining how many table spaces

Like buffer pools, you should begin with one table space for each page size being used. For each page size, create:

- A system temporary table space.
- A regular table space for indexes.
- A regular table space for frequently accessed tables.
- A regular table space for infrequently accessed, randomly accessed, and sequentially accessed tables.
- A large table space for LOB data.

Container layout

A good starting point is about 6–10 disks per CPU dedicated to table spaces. Each table space should span all disks, that is, have one container (and no more) on each available disk. You should create the same number of logical volumes (Unix) on each disk as you have table spaces. This way, each table space will have its own logical volume on each disk for container placement. If you are not using DMS raw, you will need to create a filesystem within each logical volume.

Disk array and storage subsystems

For large disk systems, you should use a single container, the reason being that if you define multiple containers, the data will be striped, not only across the DB2 containers, but effectively also at the hardware level by the RAID device. This gives us striping on top of striping, which is not desirable. Additionally, you will need to set the DB2 Profile Registry variable DB2_PARALLEL_IO for that table space.

Extent size

The extent size specifies the number of PAGESIZE pages that will be written to a container before skipping to the next container and is defined at table space creation time (and cannot easily be modified after). Smaller tables are handled more efficiently with smaller extents.

The rule-of-thumb is based on the average size of a table in the table space:

- Less than 25MB, use an extent size of 8.
- Between 25 and 250MB, use an extent size of 16.
- Between 250MB and 2GB, use an extent size of 32.
- Greater than 2GB, use an extent size of 64.
- If the table space resides on a disk array, set the extent size to the stripe size (that is, data written to one disk of the array).

Prefetch size

The optimal setting seems to be:

```
Prefetch Size = (# Containers of the table space on different physical
disks) * Extent Size
```

If the table space resides on a disk array, set it as:

```
PREFETCH SIZE = EXTENT SIZE * (# of non-parity disks in array).
```

Tables

With tables:

- Avoid using the VARCHAR data type for columns of 30 bytes or less because it typically wastes space; instead use CHAR. Wasting space can even affect query times if the volume is significant.
- When using an IDENTITY or SEQUENCE, use at least the default cache size of 20 (unless there is serious concern about gaps in the numbering). This way you are not calling on the DBM to generate a number every time one is needed and you are avoiding the logging that occurs with number generation.
- VALUE COMPRESSION and COMPRESS SYSTEM DEFAULT can save disk space when a table uses many null and system default values. This can also help improve the query time when volume is significant. Compressed columns will incur a small overhead if a value is inserted or updated.
- Use APPEND ON for tables that have heavy inserts to avoid the search for free space during the insert process; instead, simply append the row to the end of the table. If you rely on the table being in a special order and cannot afford to perform REORGs, avoid using APPEND ON.
- Set LOCKSIZE to TABLE for read-only or exclusive access tables. This avoids the time to lock the rows and reduces the amount of LOCKLIST needed.
- MDC provides for flexible, continuous, and automatic clustering of data along multiple dimensions. It improves query performance and reduces the need for REORG and index maintenance during insert, update, and delete. The best candidates for an MDC are those queries having

range, equality, and join predicates that access many rows.

- Use PCTFREE to maintain free space to help with future INSERTs, LOADs, and REORGs. The default is 10; try 20–35 for tables with clustered indexes and a heavy insert volume. If using with APPEND ON, set PCTFREE to 0.

Indexes

The following are some index-related issues that you should be aware of:

- When queries are completing in a reasonable time, avoid adding indexes because they can slow down update operations and consume extra space.
- It is sometimes possible to have one larger index that will cover several queries.
- Columns with high cardinality are good candidates for indexing.
- Avoid using more than five columns in an index because of the management overhead.
- For multi-column indexes, place the column that is most referenced in queries first in the definition.
- Avoid adding an index that is similar to a pre-existing index. It creates more work for the optimizer and will slow down update operations.
- If your table is read-only and contains a large number of rows, try to define an index that contains all columns referenced in the query using the INCLUDE clause of CREATE INDEX (INCLUDED columns are not part of the index but are stored as part of the index page to avoid additional data FETCHES).
- For best performance, create the index over small data types (like integer and char(10)), unique columns, and columns most often used in range searches.

- Clustered indexes allow for a more linear access pattern to data pages and more effective prefetching, and help avoid sorts. This means longer inserts, but quicker selects.
- When using clustered indexes consider increasing the free space on data and index pages to about 15–35 (instead of the default 10 for PCTFREE) for high volumes of inserts.
- For tables that are heavily inserted into, consider using a single dimension MDC table.
- Use ALLOW REVERSE SCANS to allow for an index to be scanned bi-directionally, which means quicker retrieval of ascending and descending result sets. This has no negative performance impact since the index structure does not change internally to support this feature.
- TYPE-2 INDEXES drastically reduce next-key locking, allow for index columns greater than 255 bytes by default, allow for both online REORG and RUNSTATS, and support the new multidimensional clustering ability. All new indexes in V8 will be created as type-2 unless a (pre-migration) type-1 index was already defined on the table. Use REORG INDEXES to convert type-1 indexes to type-2.

SUMMARY

This article has described a number of AIX and DB2 performance fundamentals. It has discussed various AIX tools for monitoring CPU, memory, and I/O, DB2 tuning tips and techniques, tablespaces, tables, bufferpools, index design, and major DB2 configuration parameters that can affect performance. By following some of the simple steps described here, you can set up, monitor, and tune your DB2 database system. I hope that the guidance offered in this article will help you achieve the goal of optimizing performance for your DB2 applications.

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

Check the DNS on an AIX server for obsolete entries

Sometimes the DNS (Domain Name System) seems to experience some kind of problem. Instead of looking all the way through */var/adm/syslog*, it might be a good idea to just clean up a little.

The following script finds entries in the DNS configuration file that can no longer be **ping**ed.

The only disadvantage with this method is if your DNS contains non-permanent TCP/IP addresses such as local PCs that can be on or off at any given time. Do not delete them from the DNS even if the script marks them as void.

Here is a sample of output from the script:

```
[root@source]/usr/local/scripts > ./checkdns
Please delete the following hosts from /etc/named.data/named.hosts
algolen6 11.13.83.21
jupiteren1 112.16.8.71
saturnatØ 132.11.2.19
desy 142.15.2.15
desyatØ 129.16.3.11
desyen1 135.14.1.12
```

And here is the script itself:

```
[root@spcws]/usr/local/scripts > cat ./checkdns
#!/usr/bin/ksh

# Script scans /etc/named.data/named.hosts
# and lists all machines which can no longer be pinged

vNamedFile="/etc/named.data/named.hosts"

print "Please delete the following hosts from ${vNamedFile}"

cat ${vNamedFile}|awk -F" " '{print $1}'|sort -u|grep -Ev "^best|^[Ø-
9]|;|@|localhost|loopback"|while read vHost
        do
              # print "${vHost} \c"
              ping -i 1 -c 1 -w 1 "${vHost}" >/dev/null 2>&1
              vError=$?
              if [ ${vError} -eq 1 ]
                     then
                          print "${vHost} $(nslookup ${vHost}|tail -n
3|grep Address |awk -F" " '{print $2}')"
                     fi
        done
```

Robert Kaiser
System Analyst
Bayerischer Rundfunk (Germany)

November 2002 – October 2005 index

Items below are references to articles that have appeared in *AIX Update* since issue 85, November 2002. 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.

CSG Systems has announced that it has completed the port of the entire suite of its Kenan FX framework to AIX. CSG chose to port its Kenan FX solutions to AIX to give global telecommunications service providers more hardware and operating system choices for running CSG solutions in both outsourced and licensed formats.

Solutions now compatible with AIX include CSG's billing platform Kenan/BP, CSG Data Mediation, CSG Kenan Prepaid, CSG's customer self-care product, Total Care and additional modules designed to support ordering, thresholding, roaming, and rating, among others.

For further information contact:
URL: www.csgsystems.com/content.cfm/ID=366/MainNav=1/MN=Investor/L=US/PR=D/PRID=559.

* * *

Quest Software has introduced a new version of Vintela Authentication Services, which now features an enhanced set of capabilities including those previously offered in Vintela Group Policy, which will no longer be available as a separate product. As a single product, Vintela Authentication Services with Group Policy components allows Unix and Linux users and systems to become "full citizens" in Windows Active Directory, which provides single-sign on and unified administration benefits across the heterogeneous enterprise.

In addition to AIX, HP-UX, Solaris, Red Hat Linux, and SuSE Linux support, Vintela Authentication Services now supports Solaris 10 and Linux on the AMD 64 and Intel EM64T 64-bit processors.

For further information contact:
URL: www.quest.com/news/show.asp?ContentId=2086&ContentTypeId=2&site=.

* * *

Crosswalk has announced Version 2.5 of Crosswalk Storage Manager, which offers in-depth support for storage devices and operating systems. With Crosswalk Storage Manager 2.5, users can view their entire storage network from end to end.

Crosswalk Storage Manager 2.5 provides support for HP StorageWorks Enterprise Virtual Array, IBM DS4000 series (formerly FAStT), Engenio and IBM's Enterprise Storage Server (ESS) series storage arrays, and AIX hosts.

For further information contact:
URL: www.crosswalkinc.com/news/press7.html.

* * *

Following IBM's acquisition of Ascential Software, IBM has unveiled new information integration software support for AIX on iSeries, powered by products from Ascential.

Ascential provided data integration software that helps build enterprise data warehouses, power business intelligence systems, consolidate enterprise business applications, and manage repositories of business information.

For further information contact your local IBM representative:
URL: www.ibm.com.