August 2003

DB2 Information Management Software

Getting Started with DB2® for Linux on AMD64

By
Dan Behman and Rav Ahuja
IBM Software Group

CONTENTS

1. OVERVIEW

AMD64, a platform that extends the x86 architecture to 64-bit, provides compelling benefits. The Database Technology team at IBM recognized the benefits of this technology at an early stage of the platform's development and became the first to enable a full-featured 64-bit relational database on this platform. IBM® DB2® Universal Database™ (DB2 UDB) Version 8.1 for Linux is now generally available on AMD64 and is ready for production use.

The advantages of running DB2 for Linux on AMD64 include:

- *Improved database performance versus other 32-bit and 64-bit systems*
- *Ability to simultaneously run existing x86 32-bit and new 64-bit applications to access 64-bit databases*
- *Unique cost benefits by combining the low total cost of ownership of DB2 UDB and Linux with the "commodity"- priced AMD Opteron systems*

While DB2 UDB for Linux has been ported and optimized for AMD64, running on this platform is very similar to running DB2 on other Linux, UNIX®, and Windows® systems. The database architecture, commands, tools and application layers for DB2 on AMD64 retain their common code and interfaces from other platforms. This allows customers to easily utilize their existing DB2 skills and knowledge and seamlessly deploy DB2 for Linux on AMD64. There are however certain subtle differences and platform unique issues that you should be aware of. This paper provides an overview of the DB2 operating environment on AMD64, and addresses commonly asked questions to enable you to quickly and effectively get up and running with DB2 UDB for Linux on AMD64 based systems.

1.1 AMD64 Architecture Exploitation

DB2 UDB for Linux on AMD64 takes full advantage of the new features and high-end processing functionality that the AMD64 platform provides. The large address space (currently 512 TB per process) is perhaps the feature that will have the greatest impact on DB2 UDB performance and scalability. This allows the database administrator to make use of huge buffer pools, which is the quickest and most effective way of increasing database performance. Besides the large address space, DB2 UDB also takes advantage of the eight additional general-purpose registers, and the eight additional floating point registers, which speed up overall database performance.

2. SUPPORTED ENVIRONMENT

It is possible to run both 32-bit and 64-bit DB2 UDB database environments on AMD64-based hardware. 32-bit applications and operating systems can benefit from a boost in performance when running on 64-bit processors from AMD as compared to running the same applications on 32-bit processors. You may run 32-bit versions of DB2 UDB on validated 32-bit Linux distributions installed on AMD64-based processor systems as long as the underlying operating system vendor supports running on the chosen AMD64-based system.

32-bit applications are limited by the 4 GB address space. For databases this limitation restricts the amount of memory that can be efficiently utilized for buffer pools. Hence, to derive the maximum benefit when running on 64-bit processors from AMD, you are advised to deploy 64-bit DB2 UDB running on 64-bit Linux distributions. Applications such as database managers that benefit the most from the 64-bit architecture can be the first to be deployed on AMD64. Applications that access the database can be kept at 32-bit and recompiled to 64-bit only if necessary.

2.1 64-bit DB2 Offerings for AMD64

As on some other platforms, IBM offers hybrid 32-bit and 64-bit DB2 solutions for AMD64. Since only certain types of applications benefit most from coding to native 64-bit, IBM has engineered the core database engine and related components of DB2 Universal Database as 64-bit and kept other components, such as the graphical front-end tools, as 32-bit. Thus you may still use most other 32-bit DB2 products with AMD64. IBM will port additional DB2 products to native 64-bit code for AMD64 as it deems necessary. At the time of writing, the 64-bit DB2 offerings available for AMD64 on Linux include:

- *DB2 Universal Database Version 8.1 – Enterprise Server Edition*
- *DB2 Universal Database Version 8.1 – Personal Edition*
- *DB2 Universal Database Version 8.1 – Personal Developer's Edition*
- *DB2 Universal Database Version 8.1 – Universal Developer's Edition*
- *DB2 Connect™ Version 8.1 – Personal Edition*
- *DB2 Connect Version 8.1 – Enterprise Edition*
- *DB2 Connect Version 8.1 – Application Server Edition*
- *DB2 Connect Version 8.1 – Unlimited Edition*
- *DB2 Version 8.1 Application Development Client*
- *DB2 Version 8.1 Administration Client*
- *DB2 Version 8.1 Run-Time Client*

The initial GA release of DB2 UDB and DB2 Connect components for AMD64 listed above are at Version 8.1.3 code level, which is equivalent to DB2 UDB Version 8.1 base level with FixPak 3 applied.

2.2 Hardware Requirements

DB2 UDB for Linux on AMD64 is supported on systems with AMD Opteron processors. Disk and memory requirements for each of the supported DB2 products are the same as on other platforms. For example, the amount of disk space needed to install Enterprise Server Edition (ESE) can range from 450 MB to 1 GB, depending on the components you choose to install (this does not include the space required for creating and storing your data within DB2 UDB databases). Similarly, ESE requires a minimum of 256 MB of memory. Additional components require more memory, such as graphical tools, which take up an additional 192 MB of memory. The actual memory requirement depends on the applications you intend to run. Additional memory may be required to handle heavy workloads and large numbers of users. In fact, in order to speed up database performance, you are advised to have as much memory as possible to minimize expensive disk access.

2.3 Software Requirements

At the time of writing, only SuSE Linux Enterprise Server 8 for AMD64 is supported. As other major 64-bit distributions are released for this platform, they will be supported for running DB2 UDB for Linux on AMD64 only after IBM has successfully validated them. Please visit the DB2 UDB for Linux validation Web site to view the list of latest supported distributions:

ibm.com/db2/linux/validate

In order to use the DB2 UDB graphical tools (such as Control Center) or for creating and running Java™ applications that interact with the database, the Service Release 4 of the IBM Developer Kit for Linux, Java 2 Technology Edition, Version 1.3.1., 32-bit version is also required. This IBM developer's kit is shipped along with the DB2 UDB installation image for AMD64 and is automatically installed when you run db2setup (if it is not already installed). For installing the supported JDK by hand, please refer to the section "Installing the Recommended Java SDK".

2.4 Limitations and Special Notes

The 64-bit version of IBM Tivoli® Storage Manager (TSM) Client is currently not available for AMD64; hence, you cannot utilize the TSM backup and restore facilities for your 64-bit databases. However, you may still back up and restore your databases using the native DB2 UDB backup and restore utilities.

Since a 64-bit enterprise-class Java environment is not currently available for AMD64, the initial release of DB2 UDB on this platform supports only the 32-bit version of the IBM SDK for Linux.

3. INSTALLATION AND SETUP

3.1 Installing the Recommended Java SDK

To use DB2 UDB graphical tools and run Java applications that access the database, you should install the supported version of the IBM Java SDK. SuSE SLES 8 for AMD64 contains a version of the IBM 32-bit SDK for Linux. However, this version was packaged separately by SuSE and is not the version that is officially supported by IBM and DB2 UDB. It is highly recommended that any existing IBM Java SDKs be removed before starting the DB2 UDB installation. You can use the `rpm` command to view and remove the unsupported Java kits. This is also documented in the Release Notes under the section "Linux enhancements".

```
hammer01: # rpm –qa | grep Java
IBMJava2-JAAS-1.3.1-5
IBMJava2-JAVACOMM-1.3.1-5
IBMJava2-JRE-1.3.1-5
IBMJava2-SDK-1.3.1-5
hammer01: # rpm -e --nodeps IBMJava2-SDK-1.3.1-5
… (repeat for any other unsupported IBM Java packages)
```

Once all other previous IBM Java SDKs have been removed, the installation of DB2 UDB (using db2setup) will automatically install the supported Java SDK. Note that the location for this SDK is `/opt/IBMJava2-131` and is specified as the default location in the database manager configuration under JDK_PATH.

If DB2 UDB was installed without removing the SDK first, the developer's kit can be manually installed once the existing version is removed[1]. The supported Java SDK can be found on the DB2 UDB installation media:

`db2/linux/Java-1.3/IBMJava2-SDK-1.3.1-4.0.i386.rpm`

```
hammer01: # cd /media/cdrom/db2/linux/Java-1.3
hammer01: # rpm –ivh IBMJava2-SDK-1.3.1-
4.0.i386.rpm
hammer01: # rpm –qi IBMJava2-SDK-1.3.1-4.0
Name        : IBMJava2-SDK
Relocations: /opt
Version     : 1.3.1
Vendor: International Business Machines
Release     : 4.0
Build Date: Fri Mar 28 23:06:34 2003
…
```

[1] If you do not remove the Java SDK that comes installed by default with SuSE SLES 8, you will see an error such as this one when attempting to install the supported IBM Java kit: `package IBMJava2-SDK-1.3.1-5 (which is newer than IBMJava2-SDK-1.3.1-4.0) is already installed`

3.2 DB2 Installation Methods

As on other Linux platforms, the following installation methods are available for installing DB2 products on AMD64:

- *Graphical DB2 Setup Wizard (`db2setup`)*
- *Response File based silent install (`db2setup –r filename.rsp`)*
- *Command line based install (`db2_install`) - this does not create DB2 users and instances or install the IBM Java SDK.*
- *Installing appropriate RPMs by hand (not recommended)*

For details on these installation methods, please refer to the Redbook: *Up and Running with DB2 for Linux* (**ibm.com**/redbooks) or the "Quick Beginnings" documentation for the appropriate DB2 product.

3.3 Creating DB2 Instances

DB2 products on AMD64 come with hybrid 32-bit and 64-bit install images. That is, once the product is installed, you can create both 32-bit and 64-bit database instances. You can create multiple DB2 instances on the same system. Each DB2 instance is independent of others on the system, and has its own settings and security. An instance can have one or many databases.

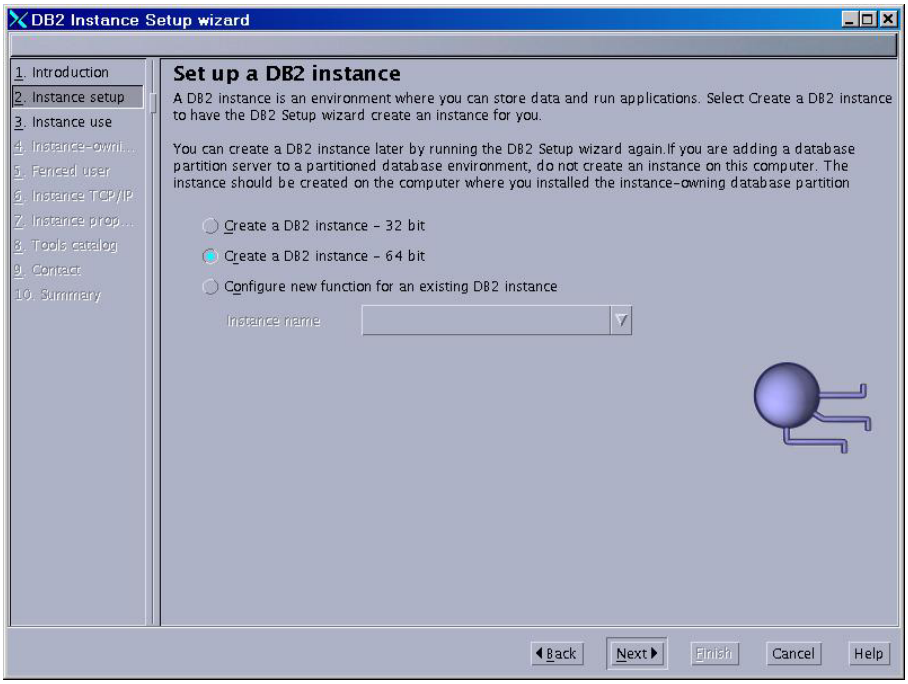
64-bit instances use native 64-bit code for managing the data. 32-bit instances use the same code as is available with DB2 for Linux on IA32 (x86) systems. The ability to create 32-bit instances is provided for running the Java-based graphical tools, serving as a 32-bit client to other instances and assisting with migration. For most other purposes you should create databases within 64-bit instances to utilize the benefits of DB2 optimizations for the AMD64 platform.

When you install the product using the DB2 Setup Wizard, and choose to create a DB2 instance, you are presented with the option of creating a 32-bit or 64-bit instance. For example, if you chose to create a 64-bit instance called `db2inst1` in the DB2 Setup wizard, you can verify that the instance is 64-bit by logging in as the user `db2inst1` and running the `db2level` command.

```
hammer01:~ # su - db2inst1
db2inst1@hammer01:~> db2level
DB21085I  Instance "db2inst1" uses "64" bits and
DB2 code release "SQL08013" with level identifier
"02040106". Informational tokens are "DB2
v8.1.1.24", "s030728", "", and FixPak "3".
Product is installed at "/opt/IBM/db2/V8.1".
```

If you would like to create additional instances after the product is installed, you may do so by either using the interactive DB2 Instance Setup wizard (`db2isetup`) or the `db2icrt` command line utility. The db2isetup utility allows you to choose the type of instance you want to create, 32-bit or 64-bit. It also creates the associated groups and userIDs, and configures communications for the instance. You need to be logged in as the root user to launch `db2isetup` from the directory `/opt/IBM/db2/V8.1/instance`.

```
db2inst1@hammer01:~> xhost +
db2inst1@hammer01:~> su - root
Password:
hammer01: # export DISPLAY=hammer01:0.0
hammer01: # cd /opt/IBM/db2/V8.1/instance/
hammer01: # ./db2isetup &
```



If you create the instance from the command line using the `db2icrt` utility, please ensure that you use the `–w 64` option to create a 64-bit instance. If you do not specify this flag, a 32-bit instance is created by default. Also note that if you want to create an instance using `db2icrt`, you need to create the associated instance owner userID and group first.

```
db2inst1@hammer01:~> su - root
Password:
hammer01: # groupadd -g 999 db2adm
hammer01: # useradd -u 1004 -g db2adm -m
-d /home/db2inst db2inst
hammer01: # cd /opt/IBM/db2/V8.1/instance/
hammer01: # ./db2icrt –w 64 -u db2inst db2inst
```

4. MIGRATING TO 64-BIT

AMD64 allows you to migrate to 64-bit at your own pace. DB2 UDB makes this migration to 64-bit for database environments seamless.

4.1 Application Scenarios

The DB2 common client architecture allows for both local and remote 32-bit and 64-bit applications to connect to the database. And since the AMD64 architecture allows for running both 32-bit and 64-bit applications on the same physical system, you may choose the application deployment strategy that works best for you:

- *Keep your application on 32-bit hardware and migrate your database to a remote 64-bit system (for client/server scenarios)*
- *Move both your application and database to the same 64-bit system, but keep your application unchanged as 32-bit and migrate the database to 64-bit (for same-tier scenarios where the application will derive little benefit by converting to 64-bit)*
- *Migrate both your application and database to 64-bit (for memory and compute intensive applications that will benefit from 64-bit)*

In most cases, the biggest advantage is gained by just moving your database to 64-bit. That way, you can get the most out of your existing application investments and minimize porting costs by continuing to run them unmodified and converting the applications to 64-bit only when needed.

4.2 Migrating databases from x86 Linux Systems to Linux on AMD64

If you want to move an existing database from a 32-bit Linux x86 computer running DB2 UDB, there are two ways of accomplishing this. The first and easiest is to back up the database on the x86 computer to removable media that can be read from the AMD64 computer, such as a tape or writable CD. Once this is complete, the database can be restored directly into the 64-bit instance on the AMD64 computer. Alternatively, the database can be restored into a 32-bit instance on the AMD64 computer followed later by updating the instance to 64-bit (see section 4.3 below). Please refer to the Administration Guide for more information on Backup and Restore. In order to migrate to DB2 UDB from other databases such as Oracle, Microsoft SQL Server, Sybase, MySQL, you can use free migration toolkits from IBM:

ibm.com/db2/migratenow

4.3 Updating a 32-bit Instance to 64-bit

If you have created a 32-bit DB2 UDB instance on an AMD64 system, you can easily migrate the instance to a 64-bit one to take full advantage of all the benefits that 64-bit computing provides.

To update an instance from 32-bit to 64-bit, run the following command as root:

```
/opt/IBM/db2/V8.1/instance/db2iupdt –w 64 <instance>
```

Please refer to the db2iupdt documentation in the *Command Reference* book. Also note that in DB2 UDB Version 8.x, databases are 32-bit/64-bit independent, which means that nothing special needs to be done with any databases that were created and manipulated with the 32-bit instance when migrating the instance to 64-bits.

5. GRAPHICAL TOOLS

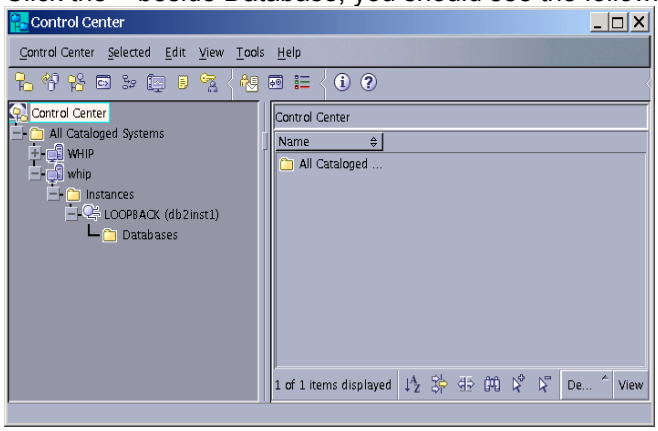
The GUI tools that come with DB2 UDB, such as the Control Center, Information Center, Configuration Assistant, and Development Center, are all Java-based graphical utilities. Since Linux for AMD64 does not currently have a 64-bit Java SDK or Runtime Environment, running these tools requires that you do so from a 32-bit DB2 UDB instance. However, you can still use these tools to manage databases in local and remote 64-bit instances. Here we provide an example using the Control Center.

5.1 Control Center

In order to administer 64-bit instances on a DB2 UDB for Linux on AMD64 computer with the Control Center, you must create a 32-bit instance first. Follow the directions in the section "Creating DB2 Instances" above to create a 32-bit instance. Once a 32-bit instance is created, you need to manually[2] catalog the local 64-bit instance that you want to administer.

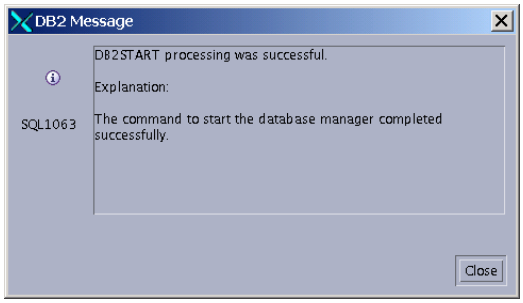
To ensure that this works correctly, please follow these directions, which assume a 64-bit instance called db2inst1 and a 32-bit instance called db2inst2 exist.

1. At the command prompt in the db2inst2 instance, enter this single command
   ```
   db2 catalog tcpip node LOOPBACK
   remote <hostname> server db2c_db2inst1
   remote_instance db2inst1
   ```
   where <hostname> is the name of your computer
2. Ensure that the DISPLAY environment variable is correctly set
3. Type `db2cc` to launch the Control Center
4. In the left pane of the Control Center window, click the **+** beside "All Cataloged Systems"; you should see two entries – one with the computer name in uppercase and another in lowercase
5. Click the **+** beside your computer name in lowercase
6. Click the **+** beside "Instances"
7. Click the **+** beside "LOOPBACK (db2inst1)"
8. Click the **+** beside Database; you should see the following:



[2] The initial GA release of DB2 for AMD64 does not support automatically discovering or cataloging local 64-bit DB2 instances from within the graphical tools. Also, do not attempt to catalog local 64-bit instances/databases using "local" protocol (use tcpip instead). These limitations will be removed in a subsequent fixpak/release.

9. Right click “LOOPBACK (db2inst1)” and select “Start”; after a few seconds you should see the following results:



You are now ready to administer your 64-bit instance!

6. APPLICATION DEVELOPMENT AND DEPLOYMENT

Because the DB2 UDB for Linux on AMD64 product is a hybrid of 32-bit DB2 UDB for Linux x86 and native 64-bit AMD64 binaries, it provides the user a great deal of flexibility and a very small up-and-running cost. If you currently use DB2 UDB for Linux on x86 and use applications or routines, it is likely that no changes are required to your code, and in fact, recompilation is likely not even required. 32-bit Java, and C/C++ applications and routines (such as stored procedures and UDFs) running against a 64-bit database are fully supported. Table 1 illustrates programming language support.

	64-bit Database Support?
32-bit Java application	yes
64-bit Java application	no[3]
32-bit Java routine	yes
64-bit Java routine	no[3]
32-bit C/C++ application	yes
64-bit C/C++ application	yes
32-bit C/C++ stored proc	yes
64-bit C/C++ stored proc	yes

Table 1

[3] Because of the lack of a 64-bit JDK for this platform

6.1 C/C++

SuSE SLES 8 includes two versions of the GNU C/C++ compiler. The default system compiler is version 3.2; version 3.3 is optionally installable. DB2 UDB for Linux on AMD64 supports routines and applications compiled with either version. Version 3.3 includes a significant number of AMD64 specific optimizations, which likely make it preferable to version 3.2. Be advised that it adheres more closely to the C++ Standard so minor code modifications may be required.

6.2 Java Applications

Java applications should be fully functional without any special configuration or setup, providing the officially supported Java SDK is properly installed.

6.3 JDBC Driver Types

All driver types except for Type 3 are supported in DB2 UDB for Linux on AMD64. Type 3 is not supported because there is no 64-bit Java SDK for Linux on AMD64.

6.4 Java Routines

The current lack of a 64-bit Java SDK or Runtime Environment means that Java routines must be configured to use the officially supported 32-bit JDK and JRE to compile and execute (see the section ”Installing the Recommended Java SDK” above). Running a 32-bit Java routine on a 64-bit database server does not incur any extra performance penalty with the exception of its very first invocation. The penalty is incurred because, by default, the 64-bit database server attempts to run the routine as if it were 64-bit. When this fails, DB2 UDB makes a record of this and then attempts to run the application or routine as 32-bit. For subsequent invocations, DB2 UDB remembers that the routine was 32-bit so it will invoke it as 32-bit immediately.

6.4.1 Getting a 32-bit Java Routine Working on a 64-bit Instance

Getting 32-bit Java routines working in a 64-bit instance can be tricky, but unfortunately, since there is no supported 64-bit JDK for Linux on AMD64, 32-bit Java routines must be used. Below is a series of steps to help you get up and running quickly. These steps use the Java routine samples that ship with DB2 UDB but do not get installed by default in a typical install; step 3 explains how to install the samples.

1. Ensure that the recommended Java SDK is installed properly (see the section “Installing the Recommended Java SDK”).
2. Ensure that the required Java SDK links exist in the `/usr/lib` directory. Please refer to the chapter on “Setting up the Linux Java Environment” in the *Application Development Guide: Building and Running Applications* for more information.
3. Ensure that the samples are installed by looking for the `~/sqllib/samples/java` directory. If this directory does not exist, perform the following steps as root:
    a. Run `db2setup` from the installation media
    b. On pane “1. Introduction”, deselect “Create a new DB2 instance or setup an existing DB2 Instance”
    c. On pane “2. Installation type”, choose “Custom” and select “Next”
    d. Select “Next” on pane 3.
    e. On pane “4. Features”, expand “Application Development tools” and select “ADT Sample Programs”; click “Next”
    f. Continue clicking “Next” until pane “7. Summary”; click “Finish”
    g. The `~/sqllib/samples/java` directory should now be present
4. As the instance ID, type cp –R ~/sqllib/samples/java ~/javatest
5. Type `chmod u+w ~/javatest/jdbc/*`
6. Change the registering of the stored procedures to be FENCED NOT THREADSAFE (see the section “Java Routine Registration” below for more information) in the `~/javatest/jdbc/SpCreate.db2` script
7. Run “`db2sampl`” to create the sample database
8. Type “`export LD_LIBRARY_PATH=~/sqllib/lib32:$LD_LIBRARY_PATH`” to ensure that the 32bit libraries can be loaded
9. In the `~/javatest/jdbc` directory, type `make SpClient` followed by `make SpServer` to make the application and stored procedure
10. Type `java SpClient`. The output should be similar to that of Figure 1 (note that only the results of the first few stored procedure calls are shown):

```
  Connect to 'sample' database using JDBC type 2 driver.

Call stored procedure named OUT_LANGUAGE
Stored procedures are implemented in language JAVA

Call stored procedure named OUT_PARAM
OUT_PARAM completed successfully
Median salary returned from OUT_PARAM = 17654.5

Sum of salaries for dept. E11 = 104990.0 before IN_PARAMS

Call stored procedure named IN_PARAMS
IN_PARAMS completed successfully
Sum of salaries for dept. E11 = 121600.0 after IN_PARAMS

Call stored procedure named INOUT_PARAM
INOUT_PARAM completed successfully
Median salary returned from INOUT_PARAM = 19260.25

Call stored procedure named INOUT_PARAM
with an input value that causes a NOT FOUND error
INOUT_PARAM failed with SQLCODE 100
INOUT_PARAM failed at GET NUMBER OF RECORDS

Call stored procedure named ONE_RESULT_SET
ONE_RESULT_SET completed successfully
============================================================
Row: 1: Edwards, Sales, 17844.00
Row: 2: Koonitz, Sales, 18001.75
Row: 3: O'Brien, Sales, 18006.00
Row: 4: Pernal, Sales, 18171.25
Row: 5: Plotz, Mgr  , 18352.80
Row: 6: Sanders, Mgr  , 18357.50
Row: 7: Lea, Mgr  , 18555.50
Row: 8: Wilson, Sales, 18674.50
Row: 9: Daniels, Mgr  , 19260.25
Row: 10: Williams, Sales, 19456.50
Row: 11: Quill, Mgr  , 19818.00
Row: 12: Lu, Mgr  , 20010.00
Row: 13: Hanes, Mgr  , 20659.80
Row: 14: Graham, Sales, 21000.00
Row: 15: Fraye, Mgr  , 21150.00
Row: 16: Jones, Mgr  , 21234.00
Row: 17: Molinare, Mgr  , 22959.20

<snip>
```

Figure 1

6.4.2 Java Routine Registration

It is also very important to understand that 32-bit Java routines must be registered as FENCED NOT THREADSAFE in order to work in a 64-bit instance. If the routine is not registered this way, an SQL1042C error will result. Note that when routines are registered this way, a performance penalty for multiple routines is incurred because a JVM must be invoked for each separate routine. Using the OUT_LANGUAGE procedure from the Java samples as an example, the original registration:

```
CREATE PROCEDURE OUT_LANGUAGE (OUT LANGUAGE
CHAR(8))
DYNAMIC RESULT SETS 0
LANGUAGE JAVA
PARAMETER STYLE JAVA
NO DBINFO
FENCED
MODIFIES SQL DATA
PROGRAM TYPE SUB
EXTERNAL NAME 'SpServer.outLanguage'@
```

should be changed to:

```
CREATE PROCEDURE OUT_LANGUAGE (OUT LANGUAGE
CHAR(8))
DYNAMIC RESULT SETS 0
LANGUAGE JAVA
PARAMETER STYLE JAVA
NO DBINFO
FENCED NOT THREADSAFE
MODIFIES SQL DATA
PROGRAM TYPE SUB
EXTERNAL NAME 'SpServer.outLanguage'@
```

6.5 Debugging

In any application development project, debugging is a very important aspect to ensure that the application is as error-free as possible. The debuggers included with SuSE SLES 8 are `gdb` for 64-bit applications, and `gdb32` for 32-bit applications.

6.4.1 "ptrace security fix" Bug

The first version of SuSE SLES 8 for AMD64 includes one major bug that has several symptoms that could negatively impact application development and troubleshooting. A patch to the kernel during development, called the "ptrace security fix" patch, causes one of the most troublesome bugs[4]. This kernel patch was applied to fix the "ptrace-related vulnerability"; see http://lwn.net/Vulnerabilities/25686/ for more information. The main symptoms of this bug are detailed below.

Symptom 1: Unable to debug setuid/setgid root applications

This symptom prevents any user, even root, from using `gdb` with executables that are setuid/setgid root. Attempting to do so will cause `gdb` to produce the output shown in Figure 2 below.

```
# gdb --pid=4690
GNU gdb 5.3
Copyright 2002 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public
license, and you are welcome to change it and/or distribute
copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty"
for details.
This GDB was configured as "x86_64-suse-linux".
Attaching to process 4690
Couldn't get registers: Operation not permitted.
A program is being debugged already.  Kill it? (y or n)
```

Figure 2

Note that the main DB2 UDB engine process, `db2sysc`, is setuid/setgid root; therefore, any attempt to debug these processes with a kernel that contains this bug will result in this problem.

Symptom 2: "ps" Listing not Showing Full Process Name

Another symptom of this bug that directly affects DB2 UDB because `db2sysc` is setuid/setgid root is that the output from the system command `ps` will not show the changed DB2 UDB process names. All DB2 UDB agent processes originate from the `db2sysc` executable. As they change to specialized processes, the process name itself gets

[4] The "ptrace security fix" bug was fixed in SuSE SLES 8 Service Pack 2 for Linux on AMD64

changed to reflect this. For example, the DB2 UDB prefetching agents start out as `db2sysc`, but when they change to their specialized function, the process name is changed to `db2pfchr`. Some other DB2 UDB processes that begin life as `db2sysc` but change their names are `db2ckpwd`, `db2syslog`, `db2agent`, and `db2pclnr`. It is easy to see that knowing the real process names is important in diagnosing DB2 UDB related problems. In a kernel with the "ptrace security fix" bug, typing `ps -fu dbehman` will produce output similar to that shown below when a connection to the SAMPLE database exists:

```
$ ps -fu dbehman | grep db2
dbehman   1103  1102  0 11:46 pts/8    00:00:00 [db2sysc]
dbehman   1108  1103  0 11:46 pts/8    00:00:00 [db2sysc]
dbehman   1109  1103  0 11:46 pts/8    00:00:00 [db2sysc]
dbehman   1110  1103  0 11:46 pts/8    00:00:00 [db2sysc]
dbehman   1111  1107  0 11:46 pts/8    00:00:00 [db2sysc]
dbehman   1115     1  0 11:46 pts/8    00:00:00
/home/dbehman/sqllib/bin/db2bp 31791C10053 5
dbehman   1116  1109  0 11:46 pts/8    00:00:00 [db2sysc]
dbehman   1117  1107  0 11:46 pts/8    00:00:00 [db2sysc]
dbehman   1118  1107  0 11:46 pts/8    00:00:00 [db2sysc]
dbehman   1119  1107  0 11:46 pts/8    00:00:00 [db2sysc]
dbehman   1120  1107  0 11:46 pts/8    00:00:00 [db2sysc]
dbehman   1121  1107  0 11:46 pts/8    00:00:00 [db2sysc]
dbehman   1122  1107  0 11:46 pts/8    00:00:00 [db2sysc]
dbehman   1123  1107  0 11:46 pts/8    00:00:00 [db2sysc]
dbehman   1124  1107  0 11:46 pts/8    00:00:00 [db2sysc]
dbehman   1132 31791  0 11:51 pts/8    00:00:00 grep db2
```

Doing the same thing on a computer with an updated kernel that contains a fix for this problem yields the following output:

```
$ ps -fu dbehman | grep db2
dbehman  20555 20554  0 11:55 pts/1    00:00:00 db2sysc
dbehman  20560 20555  0 11:55 pts/1    00:00:00 db2syslog
dbehman  20561 20555  0 11:55 pts/1    00:00:00 db2ipccm
dbehman  20562 20555  0 11:55 pts/1    00:00:00 db2resync
dbehman  20563 20559  0 11:55 pts/1    00:00:00 db2srvlst
dbehman  20566     1  0 11:55 pts/1    00:00:00
/home2/dbehman/sqllib/bin/db2bp
20472C10053 5
dbehman  20567 20561  0 11:55 pts/1    00:00:00 db2agent (SAMPLE)
dbehman  20568 20559  0 11:55 pts/1    00:00:00 db2loggr (SAMPLE)
dbehman  20569 20559  0 11:55 pts/1    00:00:00 db2loggw (SAMPLE)
dbehman  20570 20559  0 11:55 pts/1    00:00:00 db2dlock (SAMPLE)
dbehman  20571 20559  0 11:55 pts/1    00:00:00 db2pfchr
dbehman  20572 20559  0 11:55 pts/1    00:00:00 db2pfchr
dbehman  20573 20559  0 11:55 pts/1    00:00:00 db2pfchr
dbehman  20574 20559  0 11:55 pts/1    00:00:00 db2pclnr
dbehman  20575 20559  0 11:55 pts/1    00:00:00 db2event
(DB2DETAILDEADLOCK)
dbehman  20580 20472  0 11:56 pts/1    00:00:00 grep db2
```

6.4.2 The db2ps Utility

An alternative to updating the kernel is to the use the `db2ps` utility that is included with DB2 UDB for Linux on AMD64. In the same terminal in which the previous `ps –fu dbehman | grep db2` command was issued, running `db2ps` yields the following output:

```
$ db2ps
DB2 processes on node 0:
UID        PID  PPID  C STIME      TIME    CMD
root      1102     1  2 11:46  00:00:00 db2wdog
dbehman   1103  1102  2 11:46  00:00:00 db2sysc
root      1104  1103  1 11:46  00:00:00 db2ckpwd
root      1105  1103  1 11:46  00:00:00 db2ckpwd
root      1106  1103  1 11:46  00:00:00 db2ckpwd
root      1107  1103  3 11:46  00:00:00 db2gds
dbehman   1108  1103  1 11:46  00:00:00 db2syslog
dbehman   1109  1103  1 11:46  00:00:00 db2ipccm
dbehman   1110  1103  1 11:46  00:00:00 db2resync
dbehman   1111  1107  2 11:46  00:00:00 db2srvlst
dbehman   1116  1109  1 11:46  00:00:00 db2agent (SAMPLE)
dbehman   1117  1107  3 11:46  00:00:00 db2loggr (SAMPLE)
dbehman   1118  1107  0 11:46  00:00:00 db2loggw (SAMPLE)
dbehman   1119  1107  0 11:46  00:00:00 db2dlock (SAMPLE)
dbehman   1120  1107  3 11:46  00:00:00 db2pfchr
dbehman   1121  1107  3 11:46  00:00:00 db2pfchr
dbehman   1122  1107  3 11:46  00:00:00 db2pfchr
dbehman   1123  1107  3 11:46  00:00:00 db2pclnr
dbehman   1124  1107  3 11:46  00:00:00 db2event (DB2DETAILDEADLOCK)
```

Note that DB2 UDB processes that are related to this instance and are owned by root are also shown in this output, so db2ps can be even more useful than the standard `ps` system command.

5. SUMMARY

DB2 for Linux on AMD64 provides a powerful 64-bit environment for all kinds of data and information management at a lower cost than other 64-bit platforms. It also allows you to transparently migrate to 64-bit without abandoning your existing investments in 32-bit applications. DB2 exploits the AMD64 architecture to get the most out of this platform, yet retains its look and feel to help users familiar with DB2 products on other platforms to quickly take advantage of the benefits Linux on AMD64 has to offer.

To test drive DB2 UDB for Linux on AMD64 today, download the Version 8.1 Enterprise Server Edition from: **ibm.com**/db2/v8

For further information, visit: **ibm.com**/db2/linux

Or send e-mail to: DB2forLinux@ca.ibm.com

IBM Canada Ltd.
8200 Warden Avenue
Markham, ON
L6G 1C7
Canada

Printed in United States of America
08-03