IBM® DB2® for Linux®, UNIX®, and Windows®

Best practices

Troubleshooting DB2 servers

Nikolaj Richers
Information Architect
IBM

Amit Rai
Advisory Software Engineer
IBM

Serge Boivin
Senior Writer
IBM

Issued: January 2014

Table of Contents

Executive summary

Even in a perfectly engineered world, things can break. Hardware that is not redundant can fail, or software can encounter a condition that requires intervention. You can automate some of this intervention. For example, you can enable your DB2 server to automatically collect diagnostic data when it encounters a significant problem. Eventually, however, a human being must look at the data to diagnose and resolve the issue. When the need arises, you can use several DB2 troubleshooting tools that provide highly granular access to diagnostic data.

Introduction

The information and scenarios in this paper show how you can use the DB2 troubleshooting tools to diagnose problems on your server.

In large database environments, the collection of diagnostic data can introduce an unwanted impact to the system. This paper shows how you can minimize this impact by tailoring the values of a few basic troubleshooting configuration parameters such as `diagpath`, `DUMPDIR`, and `FODCPATH` and by collecting data more selectively.

The result? When things do break, you are well prepared to make troubleshooting as quick and painless as possible.

The following DB2 troubleshooting scenarios are covered in this paper:

- Troubleshooting high processor usage spikes
- Troubleshooting sort overflows
- Troubleshooting locking issues

For each scenario, this paper shows you how to identify the problem symptoms, how to collect the diagnostic data with minimal impact to your database environment, and how to diagnose the cause of the problem.

The target audience for this paper is database and system administrators who have some familiarity with operating system and DB2 commands.

This paper applies to DB2 V10.1 FP2 and later, but many of the features that are described here are available in earlier DB2 versions as well. For example, some of the serviceability functionality for large database environments was introduced in DB2 V9.7 FP4, and user-defined threshold detection for problem scenarios was introduced in DB2 V9.7 FP5. If you are not sure whether specific functionality is supported for your DB2 version, check the information center for that version.

Knowing when to contact IBM for help

An important part of troubleshooting is knowing when you cannot fix a problem yourself and you must ask for assistance. If you have a maintenance contract, you can engage IBM Support when you think that a problem goes beyond the scope of what you can or want to fix yourself. There might also be some problems that you most likely cannot fix yourself, such as problems that require diagnostic tools that are not generally available outside of IBM or problems that indicate a possible product defect.

For information about how to contact IBM and the available support options, see "Contacting IBM Software Support" in the DB2 information center (http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.admin.trb.doc/doc/t0053716.html).

Exchanging information with IBM through EcuRep

IBM has set up a standard method that you can use to exchange diagnostic information, called the Enhanced Customer Data Repository (EcuRep). EcuRep makes it easy for you to associate the diagnostic data that you upload with a problem management report (PMR), so that IBM Support personnel can find your data quickly. To avoid delays, there are naming conventions to follow when you prepare your diagnostic data for uploading.

For information about how to use EcuRep, see "Enhanced Customer Data Repository (EcuRep)" (http://www-05.ibm.com/de/support/ecurep/index.htm).

Be prepared: configure your data server ahead of time

The purpose of configuring parameter and registry variable settings before you encounter problems is to minimize the impact of diagnostic data collection and to ensure that diagnostic data is available when you need it. Generally, you want to control, not suppress, diagnostic data collection. Most importantly, you must control where diagnostic data is stored, and there must be enough free space to store the diagnostic data.

To see how your server is configured to behave during diagnostic data collection when a critical error occurs, issue the `db2pdcfg` command. The output shows how your data server responds to critical events such as trap conditions and what the current state is. Significant events, such as critical errors, trigger automatic data capture through first occurrence data capture (FODC, sometimes also referred to as `db2cos`), which is described elsewhere in this paper.

Sample output from the `db2pdcfg` command is as follows:

```
db2pdcfg

Current PD Control Block Settings:

All error catch flag settings cleared.

db2cos is enabled for engine traps.
   PD Bitmap:      0x1000
   Sleep Time:     3
   Timeout:        300
   Current Count:  0
   Max Count:      255

Current bitmap value:  0x0

Instance is not in a sleep state

Thread suspension is disabled for engine traps.

DB2 trap resilience is enabled.
   Current threshold setting : 0 ( disabled )
   Number of traps sustained : 0

Database Member 0

 FODC (First Occurrence Data Capture) options:
   Dump directory for large objects (DUMPDIR)=
/home/hbrites2/sqllib/db2dump/
   Dump Core files (DUMPCORE)= AUTO
   Current hard core file size limit = Unlimited
   Current soft core file size limit = 0 Bytes
```

For more information about the `db2pdcfg` command, see "db2pdcfg - Configure DB2 database for problem determination behavior command" (http://pic.dhe.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0023252.html).

You typically make configuration changes for troubleshooting in one of two places:

- The DB2 profile registry
- The database manager configuration

For DB2 profile registry variables, there is an important difference between the methods that you can use to make configuration changes:

- You can make changes permanently by using the `db2set` command, which requires an instance restart for changes to become effective.
- You can make changes temporarily by using the `db2pdcfg` command. Changes are effective until you restart the instance.

To retrieve information about **database manager configuration s**ettings, you issue the `GET DATABASE MANAGER CONFIGURATION` command or its abbreviated form, the `GET DBM CFG` command.

The command output includes all database manager configuration values. The following sample output has been abridged to show only values that are related to problem determination:

```
Diagnostic error capture level              (DIAGLEVEL) = 3
Notify Level                               (NOTIFYLEVEL) = 3
Diagnostic data directory path               (DIAGPATH) =
/home/db2inst2/db2diag1
Alternate diagnostic data directory path (ALT_DIAGPATH) =
/home/db2inst2/db2diag2
Size of rotating db2diag & notify logs (MB)  (DIAGSIZE) = 0
```

The diagnostic error capture error level (indicated by `DIAGLEVEL`) determines the level of detail that is recorded in the db2diag log file, and the notify level (indicated by `NOTIFYLEVL`) determines the level of detail that is recorded in the notification log file. The diagnostic data directory path (indicated by `DIAGPATH`) and the alternate diagnostic data directory path (indicated by `ALT_DIAGPATH`) determine where diagnostic data is stored.

Unless you are guided by IBM Support, do not change the default settings of parameters or registry variables that are specific to problem determination but are not described in this paper. For example, do not change the settings of the `diaglevel` configuration parameter or other `DB2FODC` registry variable parameters. If you set the level of detail for these parameters or registry variables too high, very large amounts of diagnostic data can be generated in a very short time, which in turn can negatively affect the performance of your data server. If you set the level of detail too low, insufficient data to troubleshoot a problem might be available, requiring further diagnostic data collection before you can diagnose and resolve a problem.

If you notice that the values for configuration parameters such as `diaglevel` and `notifylevel` are not set to the defaults and you are not troubleshooting a problem, you can use the `UPDATE DBM CFG` command to reset them to their defaults. In the following example, with abridged output, the `GET DBM CFG` command shows that the `diaglevel` parameter is set to the highest value possible:

```
db2 get dbm cfg

Diagnostic error capture level              (DIAGLEVEL) = 4
```

The default value is 3, which captures all errors, warnings, event messages, and administration notification messages. To reset the value for the `diaglevel` parameter to the default, issue the following command:

```
db2 update dbm cfg using DIAGLEVEL 3
```

For more information about the diaglevel configuration parameter, see "diaglevel - Diagnostic error capture level configuration parameter" (http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.admin.config.doc/doc/r0000298.html).

Redirect diagnostic data away from the DB2 installation path

The diagpath configuration parameter specifies the fully qualified primary path for DB2 diagnostic data. By default, the DB2 installation path is used. When you configure your data server, your first action is to point the diagpath parameter to a separate file system, away from the DB2 installation path.

The reason for redirecting diagnostic data away from the installation path is that the various types of diagnostic data can use significant amounts of space in the file system. By default, the db2diag and administration notification logs, core dump files, trap files, an error log, a notification file, an alert log file, and FODC packages are all written to the installation path. These files can negatively affect data server availability if the data fills up all the space in the file system.

Redirect diagnostic data away from the DB2 installation path by using the following command, replacing */var/log/db2diag* with a location on your system:

```
db2 update dbm cfg using diagpath "/var/log/db2diag"
```

The different types of diagnostic files are described in more detail in the section "db2diag and administration notification logs".

For greater diagnostic logging resilience, configure an alternate diagnostic path

The alt_diagpath configuration parameter specifies an alternate path for storing diagnostic information. Set this parameter and the diagpath parameter to different paths. The path that you specify for the alt_diagpath parameter is used only when the database manager fails to write to the path that you specified for the diagpath parameter and improves the likelihood that critical diagnostic information is not lost.

To see the value for the alt_diagpath configuration parameter, issue the following command:

```
db2 get dbm cfg

Alternate diagnostic data directory path (ALT_DIAGPATH) =
/home/db2inst2/db2diag2
```

To change the value of the alt_diagpath configuration parameter permanently, enter the following command, replacing */var/log/db2diag_alt* with a location on your system:

```
db2 update dbm cfg using ALT_DIAGPATH "/var/log/db2_diag_alt"
```

For more information about the `alt_diagpath` configuration parameter, see "alt_diagpath - Alternate diagnostic data directory path configuration parameter" (http://pic.dhe.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.admin.config.doc/doc/r0058822.html).

Redirect core file dumps and FODC data to a different directory path

Two types of diagnostic data are created only in response to specific events on your data server. One type is the FODC package, which stores diagnostic data as a problem is occurring, and the other is the core file, which preserves a memory image before a process is terminated. Both FODC packages and core files can require significant amounts of disk space. By default, both are sent to the directory path that you specified for the `diagpath` configuration parameter or, if you did not set a value for the `diagpath` parameter, the DB2 installation path.

To reduce the amount of diagnostic data that is sent to the directory path that you specify for the `diagpath` parameter, redirect FODC packages and the core file to a different directory path. You use the following `DB2FODC` registry variable settings to change the setting for the FODC packages and core files:

- `FODCPATH`: Specifies the absolute path name for the FODC package. The size of a FODC package depends on the type of collection, the operating system, and the sizes of the files that are collected. The size can reach several gigabytes.

- `DUMPDIR`: Specifies the absolute path name of the directory for core file creation. A core file can become as large as the amount of physical memory of the machine where the core file is generated. For example, a machine with 64 GB of physical memory requires at least 64 GB of space in the directory path where the core file will be stored. You can limit the size of the core file, but you should instead configure core file behavior to point to a file system with enough space to avoid lost or truncated diagnostic data.

You use the `db2set` command to make changes to these registry variable settings. For example, to redirect both FODC packages and core files to the `/tmp` path permanently, issue the following `db2set` command, which takes effect after you restart the instance:

```
db2set DB2FODC="DUMPDIR=/tmp"
```

You can also specify multiple registry variable settings for the `db2set` command, separating the settings with a space. For example, to set both the `FODCPATH` and the `DUMPDIR` registry variables at the same time, you can issue the following command, replacing the variable values with values that apply to your own system:

```
db2set DB2FODC="DUMPDIR=/home/testuser/mydumpdir
FODCPATH=/home/testuser/myfodcdir"
```

For more information about the registry variables that are supported by the `db2set` command, see "General registry variables" (http://pic.dhe.ibm.com/infocenter/db2luw/v10r5/index.jsp?topic=/com.ibm.db2.luw.admin.regvars.doc/doc/r0005657.html).

When you run the `db2support` command to collect environment data, it searches a number of paths for FODC packages, including the path that is indicated by the `FODCPATH` registry variable. You can specify an additional existing directory for the `db2support` command to search for FODC packages by using the `-fodcpath` command parameter. For more information about the parameters for the `db2support` command, see "db2support - Problem analysis and environment collection tool command" (http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0004503.html).

Configure for rotating diagnostic and administration notification logs

By default, a single DB2 diagnostic log file and a single notification log file are used. These log files grow in size indefinitely, which can become problematic if the files fill all the available space in the file system. A better approach is to use rotating diagnostic and administration notification logs, configured to work for your particular system.

When you specify that you want to use rotating diagnostic and notification logs, a series of rotating diagnostic log files and a series of rotating administration notification log files are used that fit into the size that you defined for the `diagsize` parameter. As log files fill up, the oldest files are deleted, and new files are created.

To see the current diagnostic logging setting, use the `GET DBM CFG` command:

```
db2 get dbm cfg

Size of rotating db2diag & notify logs (MB)  (DIAGSIZE) = 0
```

When the value of the `diagsize` parameter is the default of `0`, as shown in the output, there is only one diagnostic log file, called the db2diag.log file. There is also only one notification log file, which is named after the instance and has a .nfy file extension. If configured as in the above example, these files grow in size indefinitely.

To configure for rotating diagnostic and notification logs, set the `diagsize` configuration parameter to a nonzero value. The value that you specify depends on your system. Most importantly, you want to avoid losing information too quickly because of rapid file rotation (the deletion of the oldest log file) before you can archive the old files. Generally, set the `diagsize` parameter to at least 50 MB, and make sure that there is enough free space in the directory path that you specify for the

diagpath parameter. Provide the same amount of space in the directory path that you specify for the alt_diagpath parameter.

For example:

```
db2 update dbm cfg using diagsize 50
```

After you configure for rotating diagnostic logs, spend some time observing the rotation of these files. The DB2 diagnostic and notification log files should be rotated by the system every seven to 14 days. If they are rotated out too often, increase the value of the diagsize parameter. If they are rotated too infrequently, decrease the value of the parameter.

For more information about rotating diagnostic logs, see "DB2 diagnostic (db2diag) log files" (http://pic.dhe.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.admin.trb.doc/doc/c0054462.html).

Regularly archive and delete diagnostic data

Configuring your data server to use rotating diagnostic and notification logs solves the issue of log files that grow indefinitely. However, you must archive the contents of older log files before they are rotated out and deleted, so that you can access them if you need them for troubleshooting.

To archive the log files, use the db2diag -A command.[1] To avoid filling up the diagnostic directory path with the archived diagnostic data, archive the diagnostic log files to a different file system or to backup storage. After archiving a file, retain the diagnostic data for two to four weeks, for example, by backing it up to a storage solution. After this retention period has passed, you can automatically delete the diagnostic data. If you do not archive the log files to the intended location by specifying a directory path, make sure that you move the archived diagnostic data to a different location to free up the space in the diagnostic path.

The following example demonstrates how to archive. The directory listing shows which db2diag log file is in use. You can tell that this is a rotating diagnostic log file because a numerical identifier (0) is part of the file name.

```
-rw-rw-rw-    1 testuser   pdxdb2     15881799 Jun 14 14:09 db2diag.0.log
```

Now issue the db2diag -A command and include a destination path for the archived logs:

```
db2diag -A /home/testuser/archive/

db2diag: Moving "/home/testuser/sqllib/db2dump/db2diag.log"
         to     "/home/testuser/archive/db2diag.0.log_2012-06-14-15.12.47"
```

[1]On operating systems other than Windows operating systems, you can also archive all contents of the diagnostic path into an archive path by using the db2support -A command. If you use this command to archive everything, make sure that the target directory path is on a file system that has sufficient free space, equivalent to the amount of data that is in the diagnostic path.

The following directory listing shows the archived version of the log file:

```
ls -l /home/testuser/archive/ |grep -i diag

-rw-rw-rw-    1 testuser   pdxdb2     15881799 Jun 14 14:09
db2diag.0.log_2012-06-14-15.12.47
```

Having a good policy for regularly archiving and deleting the diagnostic and notification logs takes care of diagnostic data that is regularly generated, but it does not take care of all types of diagnostic data. You might need to remove other types of data after you no longer need it. For example, if you run the `db2support` command to prepare for uploading diagnostic data to the IBM Support site, you end up with a compressed archive that takes up space. Remember to remove this archive after your problem report is resolved. You must also manually remover any additional data dump or FODC packages that are generated.

Provide enough free space to store diagnostic data

Diagnostic data can use substantial amounts of space, and you must ensure that enough space is available to store this data. How much space is needed depends on the type of diagnostic data.

Diagnostic and notification logs: For both the primary diagnostic path that you specify for the `diagpath` parameter and the alternate diagnostic path that you specify for the `alt_diagpath` parameter, provide at least 20% more free space than the value of the `diagsize` parameter.

Minimum space for diagnostic and notification logs = value of the `diagsize` parameter x 1.2

Core file dumps and FODC data: For free space, provide at least twice the amount of physical memory of the machine, plus 20%. Providing this much space ensures that you can store at least two full core file dumps or several FODC packages without running the risk of truncated diagnostic data.

Minimum space for core files and FODC packages = 2 x physical memory x 1.2

For example, if a machine has 64 GB of physical memory, provide a minimum of 154 GB of space for core files and FODC packages in the file system (64 GB x 2 x 1.2 = 154 GB).

Diagnostic data that you are uploading to the IBM Support site: If you run the `db2support` command to prepare to upload diagnostic data to the IBM Support site, make sure that enough space is available. The size of the db2support.zip file depends on what parameters you specify for the `db2support` command, but the size of the db2support.zip file can range from several megabytes to more than tens of gigabytes. If you do not specify an output path, the resulting compressed archive is stored in the directory path that you specified for the `diagpath` parameter.

First steps for troubleshooting

This section explains the initial steps for identifying and diagnosing apparent errors and performance problems. The purpose of these troubleshooting steps is to determine the following information:

- What information to collect from DB2 tools and logs and from operating system tools and logs and what environmental information to collect
- How to use this information in problem investigation

The first step is to characterize the issue by asking the following questions:

- What are the symptoms?
- Where is the problem happening?
- When does the problem happen?
- Under which conditions does the problem happen?
- Is the problem reproducible?

You might also ask include whether there were any recent changes that might be implicated in the problem. Some problems, such as performance problems or problems that occur only intermittently or only after some time has elapsed, are much more open ended and require an iterative approach to troubleshooting.

After you have characterized the issue, you can use a number of tools and logs. In the sections that follow, the main tools and diagnostic logs are described. These tools include FODC, DB2 diagnostic and administration notification logs, DB2 tools, and operating system tools and logs.

The DB2 monitoring infrastructure can also provide a wealth of information about the health and performance of DB2 servers. Using table functions, you can access a broad range of real-time operational data (in-memory metrics) about the current workload and activities, along with average response times. Using event monitors, you can capture detailed activity information and aggregate activity statistics for historical analysis.

You can also perform some of the troubleshooting and monitoring tasks that are covered in this paper by using the IBM InfoSphere® Optim™ and IBM Data Studio tools. Information about how to use all of these tools is outside the scope of this paper, but you might consider the following Optim tools:

- IBM InfoSphere Optim Performance Manager (OPM): Provides easy-to-use performance monitoring. Alert mechanisms inform you of potential problems. Historical tracking and aggregation of metrics provide information about system performance trends. For more information, see “IBM InfoSphere Optim

Performance Manager" (http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.idm.tools.doc/doc/c0057035.html).

- OPM Extended Insight: Measures end-to-end response time to detect issues outside your DB2 data server. For more information, see "IBM InfoSphere Optim Performance Manager Extended Insight" (http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.idm.tools.doc/doc/c0057246.html).
- Optim Query Workload Tuner: Performs deep-dive analysis to identify and solve many types of query bottlenecks. For more information, see "IBM InfoSphere Optim Query Workload Tuner for DB2 for Linux, UNIX, and Windows" (http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.idm.tools.doc/doc/c0057033.html).

First occurrence data capture (FODC)

FODC is the built-in DB2 facility for detecting specific failure scenarios. FODC automatically captures diagnostic data when a specific error condition occurs, and you can also use it to manually capture data for a specific problem scenario that you are observing. FODC minimizes the need to reproduce problem scenarios, because diagnostic data is collected as the problem first occurs.

By default, FODC invokes a db2cos callout script to collect diagnostic data. The db2cos callout script is located in the bin directory in the DB2 installation path (in the sqllib/bin directory, for example). You can modify the db2cos callout script to customize diagnostic data collection.

In DB2 V9.7 FP5 and later (excluding V9.8), FODC supports defining your own threshold rules for detecting a specific problem scenario and collecting diagnostic data in response. You define threshold rules by specifying the `-detect` parameter for the `db2fodc` command. To detect a threshold condition and to trigger automatic diagnostic data collection when the threshold condition is exceeded multiple times, create an FODC threshold rule such as the following one:

```
db2fodc -memory basic -detect free"<=10" connections">=1000"
sleeptime="30" iteration="10" interval="10" triggercount="4"
duration="5" -member 3

"db2fodc": List of active databases: "SAMPLE"

"db2fodc": Starting detection ...
```

The effect of this threshold rule is as follows. On member 3, detection is performed to check whether the conditions that are specified by the threshold rules free<=10 and connections>=1000 are met. These threshold rules specify that the size of the free list must be 10 or less and the number of connections must be 1000 or more for the threshold to be detected. FODC memory collection is triggered on member 3 when the number of times that the threshold conditions are detected reaches the value that is

specified by the trigger count. In this example, for FODC collection to be triggered, the trigger conditions must exist for 40 seconds (`triggercount` value of 4 x `interval` value of 10 seconds = 40 seconds). The detection process sleeps for 30 seconds between each iteration, and the total time that detection is enabled is 5 hours. When FODC memory collection is triggered, a new directory whose name is prefixed with FODC_Memory_ is created in the current diagnostic path.

When the threshold conditions that are defined by the `-detect` parameter are met and FODC memory collection is triggered, a message similar to the following one is displayed in the command window:

```
"db2fodc": 4 consecutive threshold hits are detected.

"db2fodc": Triggering collection 1.
```

Messages are also written in the db2diag.log file, as shown in the following example. To get details about a triggered threshold, you can use tools and scripts to scan the db2diag.log file and look for the string **`pdFodcDetectAndRunCollection, probe:100`**.

```
2013-04-02-14.24.54.951556-240 I2341E790 LEVEL: Event
PID : 13279 TID : 47188095859472 PROC : db2fodc
INSTANCE: inst1 NODE : 003
FUNCTION: DB2 UDB, RAS/PD component,
pdFodcDetectAndRunCollection, probe:100
CHANGE :
Hostname: host11 Member(s): 3 Iteration: 1
Thresholds hit 0: free(8)<=10 connections(1010)>=1000
Thresholds hit 1: free(8)<=10 connections(1009)>=1000
Thresholds hit 2: free(9)<=10 connections(1001)>=1000
Thresholds hit 3: free(10)<=10 connections(1005)>=1000
```

You can also gather diagnostic performance data selectively without defining threshold rules by using the `-cpu`, `-connections`, or `-memory` parameter. These parameters are alternatives to collecting diagnostic data more extensively and expensively with the `-perf` and `-hang` parameters when you already have a preliminary indication of where a problem might be occurring.

As of DB2 V9.7 FP4, FODC collects diagnostic data at the member level to provide more granular access to diagnostic data. Member-level FODC settings provide greater control than the instance-level or host-level settings that were supported in previous releases and fix packs.

db2diag and administration notification logs

DB2 diagnostic and administration notification messages are both logged in the db2diag log files, making the db2diag log files one of the first places to check if you suspect a problem. Using the `db2diag` command, you can analyze the db2diag logs to extract problem-specific information. For example, you can extract error messages

that are related to health indicators on a specific date by using a command such as the following one:

```
db2diag -level Error -time 2012-06-13 -gi message:=health

2012-06-13-14.39.13.850508-240 E15877107A655      LEVEL: Error
PID     : 25821308             TID  : 772         PROC : db2acd
INSTANCE: test99               NODE : 000
EDUID   : 772                  EDUNAME: db2acd
FUNCTION: DB2 UDB, Health Monitor, HealthIndicator::update, probe:500
MESSAGE : ADM10500E  Health indicator "Log Filesystem Utilization"
          ("db.log_fs_util") breached the "upper" alarm threshold of "85
%"
          with value "89 %" on "database" "mtmelo.SAMPLE  ".  Calculation:
          "((os.fs_used/os.fs_total)*100);" = "((65226387456 / 73014444032
) *
          100)" = "89 %".  History (Timestamp, Value, Formula): "()"
```

You can also use the `db2diag` command to merge multiple log files.

You should monitor the administration notification log to determine whether any administrative or maintenance activities require manual intervention. For example, if the directory where transaction logs are kept is full, this blocks new transactions from being processed, resulting in an apparent application hang. In that situation, the DB2 process writes the error ADM1826E to the administration notification log, as shown in this example:

```
2013-04-19-13.03.30.699042   Instance:shenli   Node:000
PID:13045(db2sysc 0)    TID:2970467744   Appid:none
data protection services  sqlpgCallGIFL Probe:1540

ADM1826E  DB2 cannot continue because the disk used for
logging is full.
```

The error condition is also written to the db2diag log file, as shown here:

```
2013-04-19-13.03.30.618650-240 E2656E381            LEVEL:
Error
PID     : 13045               TID : 46912603274656  KTID :
13045
PROC    : db2sysc 0
INSTANCE: dbinst1              NODE : 000           DB   :
SAMPLE
HOSTNAME: host1
EDUID   : 49                  EDUNAME: db2loggr (SAMPLE) 0
FUNCTION: DB2 UDB, data protection services, sqlpgCallGIFL,
probe:1540
MESSAGE : ADM1826E  DB2 cannot continue because the disk used
for logging is full.
```

DB2 tools

Several DB2 commands are used regularly as troubleshooting tools. The commands that are described in this section are the ones that you might use most often. The scenarios in this paper provide examples of how to use these commands.

- **`db2pd` command:** The `db2pd` command is a stand-alone command that you can use to monitor and troubleshoot a DB2 instance from its database system memory. The `db2pd` command collects information without using any engine resources or acquiring any latches. Because the `db2pd` command does not acquire any latches, you can normally retrieve information that is changing while the `db2pd` command is collecting information. You can rerun the `db2pd` command if results don't appear to be accurate.

  On a slow or non-responsive database system, the `db2pd` command is one of the most important tools that you can use. The `db2pd` command works on a DB2 engine that might otherwise appear to be hung.

  For more information, see "db2pd - Monitor and troubleshoot DB2 database command" (http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0011729.html).

- **`db2diag` command:** This command filters, formats, and archives the diagnostic information in the db2diag log files. Filtering records in the db2diag log files can reduce the time that you require to locate the records that you need when troubleshooting problems. For example, you can use process information that you obtain from the `db2pd` command to filter related diagnostic information in the db2diag log files by using the `db2diag` command.

  You can archive rotating diagnostic log files to retain diagnostic data that would otherwise be eventually overwritten and move the files to a different location for storage.

  For more information, see "db2diag - db2diag logs analysis tool command" (http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0011728.html).

- **`db2top` command (**UNIX and Linux operating systems): This command uses the snapshot monitor to provide a single-system view for partitioned database environments. The `db2top` command can help you identify performance problems across the whole database system or in individual partitions. You can also use the `db2top` command on single-partition environments.

  On large systems, the `db2top` command can require large amounts of memory because the global snapshot buffer can grow large. Particularly for large partitioned environments, you should carefully choose the update interval so as not to generate excessive traffic and overtax the fast

communication manager (FCM) buffer shared memory. You specify the update interval by using the `-i` command parameter.

For more information, see "db2top - DB2 monitoring tool command" (http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0025222.html).

- **`db2support` command:** This command archives all the diagnostic data from the directory that you specify for the `diagpath` configuration parameter into a compressed file archive. You typically use the **`db2support`** command to prepare to upload the data to the IBM Support site or to analyze the diagnostic data locally. You can limit the amount of data that is collected to a specific time interval by using the `-history` or `-time` parameter, and you can decompress the compressed file archive.

  For more information, see "db2support - Problem analysis and environment collection tool command" (http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0004503.html).

- **`db2caem` command:** This command automates the process of creating and running an activity event monitor to collect detailed diagnostic and runtime information about one or more SQL statements. The `db2caem` command extracts and formats the information that is captured by the activity event monitor. The `db2support` command includes a number of options to collect the information that is generated by the `db2caem` command.

  For more information, see "db2caem - Capture activity event monitor data tool command" (http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0057282.html).

You typically run the following commands under the guidance of IBM Support personnel. The commands are useful to help you gather the required information to help support personnel assist you in diagnosing and correcting problems.

- **`db2trc` command:** This command collects traces through the DB2 trace facility. The process requires setting up the trace facility, reproducing the error, and collecting the data. In V9.7 FP4 and later, the `db2trcon` and `db2trcoff` scripts simplify using the `db2trc` command. The `db2trc` command can have a significant performance impact unless you limit what you trace to specific application IDs or top EDUs.

  For more information, see "db2trc - Trace command" (http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0002027.html).

- **`db2dart` command:** This command examines databases for architectural correctness and reports any errors. You typically use it for the following purposes:

  - To inspect an entire database, a table space, or a table for correctness
  - To repair a database
  - To change a database state
  - To dump formatted table data from a database, for example, if a database was corrupted because of a hardware failure and a current backup is not available,

  For more information, see "db2dart - Database analysis and reporting tool command" (http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0003477.html).

- **`INSPECT` command:** This command examines a database for architectural integrity, checking the pages of the database for page consistency. The `INSPECT` command checks that the structures of table objects and structures of table spaces are valid. Cross-object validation conducts an online consistency check between the index and the data. The `INSPECT` command can identify logical corruption that the `db2dart` command might not detect.

  For more information, see "INSPECT command" (http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0008633.html).

Operating system tools and log files

Diagnosing some problems requires you to look at both DB2 diagnostic data and operating system diagnostic data, such as problems related to memory, swap files, CPU, disk storage, and other operating system resources.

On UNIX and Linux operating systems, the following system tools are available:

- **`vmstat` command:** This is a good overall tool for showing whether processor or memory bottlenecks exist. You can run it continuously.
- **`iostat` command:** You can use this command to find out whether any disk I/O bottleneck exists and what the I/O throughput is.
- **`ps` command:** This command provides information about the processes that use the most processor time.
- **`svmon` command** (AIX operating systems only): This command provides an in-depth analysis of memory usage. It provides more detailed information

than what the `vmstat` and `ps` commands provide, although the performance impact is slightly higher.

On Windows operating systems, you can use the following system tools and commands:

- **`db2pd -vmstat` command:** Use the `db2pd` command with the `-vmstat` parameter to show whether processor or memory bottlenecks exist.
- **`db2pd -iostat` command:** Use the `db2pd` command with the `-iostat` parameter to show whether any disk I/O bottleneck exists.
- **Task Manager:** You can use this tool to show memory consumption and processor usage. Alternatively, you can use the `db2pd -edus` command to return similar information.
- **Process Explorer:** This tool is similar to the Task Manager but provides additional information and functionality for processes.
- **Windows Performance Monitor:** You can use this tool to monitor system and application performance in real time and historically. The tool supports data collection that you can customize, and you can define automatic thresholds for alerts and actions to take in response. The tool can also generate performance reports.

The following operating system error logs, which differ by operating system, can contain important diagnostic information:

- **AIX operating systems:** /usr/bin/errpt –a
- **HP-UX operating systems:** /var/adm/syslog/syslog.log and /usr/sbin/dmesg
- **Linux operating systems:** /var/log/messages and /usr/sbin/dmesg
- **Solaris operating systems:** /var/adm/messages and /usr/sbin/dmesg
- **Windows operating systems:** Event logs and the Dr. Watson log

Monitoring infrastructure

The lightweight, metrics-based monitoring infrastructure that was introduced in DB2 Version 9.7 and is available in DB2 Version 10.5 provides pervasive and continuous monitoring of both system and query performance. Compared to the older snapshot and system monitor, the monitoring infrastructure provides real-time in-memory aggregation and accumulation of metrics within the DB2 system at different levels, with a relatively low impact on the system.

You can use the DB2 monitoring infrastructure to gain an understanding of the typical workloads that your data server processes. Understanding your typical workloads is

an important step toward identifying atypical events that require further investigation or perhaps troubleshooting.

The three focus areas for monitoring information are as follows:

- System: Provides a complete perspective on the application work (database requests) being performed by the database system, collected through the WLM infrastructure.
- Activities: Provides a perspective on work being done by specific SQL statements, collected through the package cache infrastructure.
- Data objects: Provides a perspective on the impact of application work on data objects, collected through the data storage infrastructure.

The different kinds of information include the following ones:

- Time-spent metrics that identify how the time that is spent breaks down into time spent waiting (lock wait time, buffer pool I/O time, and direct I/O time) and time spent doing processing.
- In-memory metrics. SQL table functions provide highly granular access to these metrics.
- Section explains that show the access plan that was executed for a statement without the need for recompiling the statement.
- Section actuals, which can shorten the time to discover problem areas in an access plan when you compare them to the estimated access plan values. You use the `db2caem` command to get section actuals.

Configuring in-memory metrics for troubleshooting

The `mon_req_metrics`, `mon_act_metrics`, and `mon_obj_metrics` configuration parameters control the collection of metrics. By default, these metrics are set to `BASE` for new databases that you create in DB2 V9.7 and later. For most situations, the information that is returned with the `BASE` setting is sufficient. When troubleshooting, you might need to set the `mon_req_metrics` and `mon_act_metrics` configuration parameters to `EXTENDED` to collect more granular, time-based information.

Additionally, there are other configuration parameters, such as the `mon_uow_data` parameter, that are set to `NONE` by default. To collect some of the unit of work information, you might need to set the values for these parameters. To see the current values for monitoring-related configuration parameters, you can issue the following command:

```
db2 get db cfg | grep MON

Request metrics                         (MON_REQ_METRICS) = BASE
Activity metrics                        (MON_ACT_METRICS) = NONE
Object metrics                          (MON_OBJ_METRICS) =
EXTENDED
Unit of work events                        (MON_UOW_DATA) = NONE
Lock timeout events                     (MON_LOCKTIMEOUT) = NONE
```

```
Deadlock events                                 (MON_DEADLOCK) =
WITHOUT_HIST
Lock wait events                                (MON_LOCKWAIT) = NONE
Lock wait event threshold                      (MON_LW_THRESH) =
5000000
Number of package list entries                (MON_PKGLIST_SZ) = 32
```

For more information about the available configuration parameters and the metrics that are returned for each parameter setting, see "Configuration parameters" (http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/index.jsp?topic=/com.ibm.db2.luw.admin.config.doc/doc/c0004555.html).

Event monitor infrastructure

Event monitors in DB2 Version 9.7 and later use a highly scalable, lightweight infrastructure. Highlights of this architecture are as follows:

- It maximizes parallelism by using one thread per processor core.
- It uses multiple threads for high volume event monitors.
- It does not require a dedicated thread for low volume event monitors.

Fast-writer threads provide the infrastructure for supplementing event information and output formatting. Formatting of data and output of data are performed asynchronously from the processing agents. Multiple fast-writer threads make it much less likely for a backlog to occur in the event monitor queues.

To further reduce the system impact of high-volume event monitors and to reduce storage requirements, DB2 Version 9.7 introduced a new event monitor target type, the unformatted event table. You can format the unformatted event table data as follows:

- Into XML data output by using the `EVMON_FORMAT_UE_TO_XML` table function. An example follows:

```
SELECT evmon.* FROM TABLE (EVMON_FORMAT_UE_TO_XML (NULL,FOR EACH
ROW OF (select * from LOCK order by EVENT_TIMESTAMP where
EVENT_TYPE = 'LOCKWAIT' and EVENT_TIMESTAMP >= CURRENT_TIMESTAMP
. 5 hours )))
```

- Into relational table data output by using the `EVMON_FORMAT_UE_TO_TABLE` procedure. An example follows:

```
call EVMON_FORMAT_UE_TO_TABLES ('UOW', NULL, NULL, NULL, NULL,
'IBMDB2SAMPLEXML', 'RECREATE_FORCE', -1,'SELECT * FROM UOWTBL
ORDER BY event_timestamp')
```

Event monitors that use unformatted event tables to store their data include the following ones:

- Locking event monitor: This provides a consolidated mechanism for capturing locking data you can use for in-depth analysis. This event monitor replaces the

deadlock event monitor and lock timeout reports. Information about deadlocks, lock timeouts, and lock waits is returned. You can control the granularity of the information that is returned at the workload or at the database level. A statement history is also available. The following statement creates a locking event monitor:

```
CREATE EVENT MONITOR MY_LOCKEVMON FOR LOCKING WRITE TO
UNFORMATTED EVENT TABLE (IN USERSPACE1)
```

- Unit of work event monitor: This replaces the transaction event monitor. You can control the granularity of the information that is returned at the workload or at the database level. Data that is captured includes in-memory metrics. The following statement creates a unit of work event monitor:

```
CREATE EVENT MONITOR UOWEVMON FOR UNIT OF WORK WRITE TO
UNFORMATTED EVENT TABLE (IN USERSPACE1)
```

- Package cache event monitor: This captures both dynamic and static SQL entries when they are removed from the package cache. Entries begin to be captured as soon as the event monitor is activated. You can control the granularity of the information that is returned by using the `WHERE` clause when you define the event monitor. The `WHERE` clause for the event monitor can include one or more of the following predicates (ANDed): the number of executions, overall aggregate execution time, and evicted entries whose metrics were updated since last boundary time set using the `MON_GET_PKG_CACHE_STMT` function. You can also use the event monitor definition to control the level of information that is captured; options include `BASE` and `DETAILED`. The following statement shows an example of a package cache event monitor:

```
CREATE EVENT MONITOR MY_PKGCACHE_EVMON FOR PACKAGE CACHE WRITE
TO UNFORMATTED EVENT TABLE (IN USERSPACE1)
```

Text reports for monitoring data

Similar in purpose to the `GET SNAPSHOT` command but available through any SQL interface, monitoring reports are generated by the `MONREPORT` module. These reports are provided in the form of result sets that are returned from stored procedures. These reports can provide helpful monitoring information for pinpointing problem areas when troubleshooting.

Consider the following example. The users of a new Java application are complaining about slow performance. Expected results are not being returned within the expected time frame.

As a first step, you can run the `MONREPORT.DBSUMMARY` summary report for six minutes (as an example) to get a picture of the system and application performance metrics for the database while the application is running. To run the summary report for six minutes, issue the following command:

```
db2 "call monreport.dbsummary(360)"
```

After the report has finished running, you can look at the wait times that are included in the report to see whether there are any significant delays that might indicate a problem area. In this example, the summary report includes the following information:

```
------------------------------------------------------------------------------
        -- Detailed breakdown of TOTAL_WAIT_TIME --

                                %    Total
  -------------------------------------------
  TOTAL_WAIT_TIME               100  711546

  I/O wait time
    POOL_READ_TIME              0    323
    POOL_WRITE_TIME             0    0
    DIRECT_READ_TIME            0    0
    DIRECT_WRITE_TIME           0    0
    LOG_DISK_WAIT_TIME          0    134
  LOCK_WAIT_TIME                0    0
  AGENT_WAIT_TIME               0    0
  Network and FCM
    TCPIP_SEND_WAIT_TIME        96   684581
    TCPIP_RECV_WAIT_TIME        4    26510
    IPC_SEND_WAIT_TIME          0    0
    IPC_RECV_WAIT_TIME          0    0
    FCM_SEND_WAIT_TIME          0    0
    FCM_RECV_WAIT_TIME          0    0
  WLM_QUEUE_TIME_TOTAL          0    0
------------------------------------------------------------------------------
```

The output shows that most of the wait is happening while data is being sent back to the client (the value in the `TCPIP_SEND_WAIT_TIME` row is high). Further investigation with JDBC tracing reveals an improper override setting by the `Statement.setFetchSize` method. After you set the fetch size (`FetchSize` parameter) correctly, application performance not only improves but exceeds expectations.

Other monitoring reports that you might find useful in troubleshooting include the `MONREPORT.CONNECTION`, the `MONREPORT.CURRENTAPPS`, the `MONREPORT.CURRENTSQL`, the `MONREPORT.PKGCACHE`, and the `MONREPORT.LOCKWAIT` reports.

Full coverage of the monitoring infrastructure is beyond the scope of this paper, although the monitoring infrastructure is used in some of the scenarios. For more information about DB2 monitoring, see "Database monitoring" (http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.admin.mon.doc/doc/c0001138.html) in the DB2 Information Center.

Minimizing the impact of troubleshooting

A system already affected by performance issues might not be able to tolerate much, if any, additional load, even to collect basic diagnostic data. Configuration as recommended earlier does not reduce the performance impact of diagnostic data collection, but it does help you control where diagnostic data is stored. There are specific things you can do to reduce the performance impact, though.

Collect diagnostic data only where the problem is occurring

To avoid the overhead of unnecessary diagnostic data collection in large database environments, several troubleshooting commands support options to specify where to collect data. For example, you can collect data at the member level instead of the host level if you know that a problem affects only a member, not the host machine. These options speed up data collection by collecting only relevant information, which reduces the performance impact of data collection on the system and can shorten the time that is required to perform problem determination.

For example, to collect FODC data during a performance issue on members 10, 11, 12, 13, and 15, issue the following command:

```
db2fodc -perf -member 10-13,15
```

Collect only the diagnostic data you need

You can use FODC collections introduced in V9.7 FP5 to collect diagnostic data only for the specific type of problem that you encounter rather than collecting more comprehensive performance data by using the `db2fodc` command `-perf` and `-hang` parameters. The collections for problems that are related to processor usage, memory, and connections are lightweight and minimize the performance impact. Use the collections by specifying the `-cpu`, `-memory`, and `-connections` parameters with the `db2fodc` command . For more information, see "Collecting diagnostic data for specific performance problems" (http://pic.dhe.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.admin.trb.doc/doc/p0059590.html).

Avoid service delays due to transferring diagnostic data

Some problems might require you to contact IBM's technical support, and this means that a service analyst must look at the diagnostic data that you collected on your system. Typically, this means that you must use the `db2support` command and upload the diagnostic data. If the volume of diagnostic data that you generated on your system is large, uploading this volume of data to the IBM Support site for analysis can introduce additional delays before the problem can be diagnosed. In V9.7 FP4 and later, you can use the `db2support` command with the `-unzip` parameter to extract packages locally rather than uploading them to the IBM Support site . In addition, some tools that service analysts use routinely are now installed when you install the DB2 software.

The following command extracts packages from the file db2support.zip:

```
db2support -unzip db2support.zip
...
Extracting "DB2DUMP/db2diag.log" ...
Extracting "STMM/stmm.0.log" ...
Extracting "EVENTS/db2event.0.log" ...
Extracting "EVENTS/db2optstats.0.log" ...
Extracting "EVENTS/.db2optstats.rotate.lck" ...
Extracting "autopd.zip" ...
Extraction completes.
```

Scenarios

The following scenarios show how you can apply some of the troubleshooting best practices. The list of scenarios represents only a small sample of possible scenarios that you might encounter while troubleshooting a DB2 server. For links to additional scenarios and recommendations, see the "Additional information" section of this paper's web page in the DB2 best practices developerWorks community .

Scenario: Troubleshooting high processor usage spikes

Identifying the problem

Users of an application tell you that the response times for the application are occasionally very slow. Every so often, when the application is waiting on the dedicated database server, the application slows to a crawl for a while. You are asked to investigate the cause.

You log on to the database server to do a preliminary investigation. You run some operating system tools, such as the `top` command or the Windows Task Manager, to see what the current processor usage is. As these tools are running, you observe that the processor usage occasionally spikes to above 90% and remains high for several minutes before dropping down to what seem to be more typical usage levels:

```
top - 15:59:54 up  7:27,  6 users,  load average: 0.61, 0.19, 0.29
Tasks: 171 total,   1 running, 170 sleeping,   0 stopped,   0 zombie
Cpu(s): 98.0%us,  2.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,
0.0%st
Mem:   2063588k total,  1163576k used,   900012k free,    98580k buffers
Swap:        0k total,        0k used,        0k free,   741764k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 8806 db2inst1  20   0  600m 192m 151m S 96.6  9.5  21:05.45 db2sysc
 7303 db2inst1  20   0 84588  13m   9m S  1.3  0.7   0:11.05 gnome-
terminal
 6394 root      20   0 49584  25m 7632 S  1.0  1.3   0:28.20 X
 7161 db2inst1  20   0  116m  27m  20m S  0.7  1.4   0:22.44 nautilus
...
```

In this example, you can see that the user processor usage is 98%, with the bulk of the high processor usage, 96.6%, coming from the db2sys process. This process is for the DB2 system controller (on Windows operating systems, look for db2sys.exe). Without further information, you know only that the problem occurs intermittently and lasts several minutes. There doesn't seem to be a specific time at which it occurs.

Diagnosing the cause

Intermittent performance issues can be challenging to diagnose because of the difficulty in collecting the data that is necessary to troubleshoot the cause. This type of issue is unlikely to trigger automatic diagnostic data collection, and by the time that you can manually collect data, the issue might have passed, leaving you with little or no diagnostic data. For an intermittent performance problem, your first step is to set up a method to capture the diagnostic data that you need, as the problem is occurring.

To capture diagnostic data for the intermittent processor usage spikes that you observed, you define an FODC threshold rule. An FODC threshold rule is a tool that waits for the resource conditions that you define to occur. In this case, you have some preliminary information that points to high processor usage. If you don't know what system resources are constrained, you can adapt this scenario to collect data about additional system resources, such as connections and memory. After you set up the FODC threshold rule, it triggers an FODC collection whenever the threshold conditions that you specified for processor usage are exceeded, for as many occurrences of the problem as you specified.

You define a FODC threshold rule for processor usage by using the `db2fodc -detect` command. The `db2fodc -detect` command performs detection at regular intervals for as long as you tell it to, if you specify a duration. If you do not specify a duration, detection runs until the threshold conditions are triggered. The term *Threshold conditions* in this context refers to both a specific frequency of the problem and a duration that must be met before a collection is triggered.

The following threshold rule is a good start for detecting processor usage spikes:

```
$  db2fodc -cpu basic -detect us_sy">=90" sleeptime="30"
iteration="10" interval="10" triggercount="4"
duration="5"
```

In this case, the threshold rule is used to detect a combined user and system processor usage rate that is higher than 90%. For FODC collection to be triggered, the threshold conditions must exist for 40 seconds for each iteration (`triggercount` value of 4 x `interval` value of 10 seconds = 40 seconds). The detection process sleeps for 30 seconds between each iteration. The total time that detection is enabled is 5 hours or 10 iterations of successful detection in total, whichever comes first. If FODC collection is triggered, a new directory with a name that is prefixed with FODC_CPU_ is created in the current diagnostic path. Only the lighter-weight, basic collection of diagnostic data is performed.

Now, assume that detection has been running for a while and the threshold conditions for processor usage spikes are met, which means that FODC collection is triggered. During FODC collection, you might see output that is similar to this example:

```
"db2fodc": Starting detection ...
"db2fodc": "4" consecutive thresholds hits are detected.
"db2fodc": Triggering collection "1".
Script is running with following parameters
COLLECTION_MODE           : LIGHT
COLLECTION_TYPE           : CPU
COLLECTION_DURATION       : 5
COLLECTION_ITERATION      : 10
DATABASE/MEMBER           : -alldbs
FODC_PATH                 :
/var/log/db2diag/db2dump/FODC_Cpu_2013-07-14-
11.09.51.739430_0000
db2pd_options             : -agent -apinfo -active -tran
-locks -bufferpools -dbptnmem -memset -mempool -sort -fcm hwm
-dyn
SNAPSHOT                  : 2
STACKTRACE                : 2
TRACELIMIT                : 20
SNAPSHOT_TYPE             : ALL
```

This output specifies where to look for the diagnostic data for the particular FODC package (/var/log/db2diag/db2dump/FODC_Cpu_2013-07-14-11.09.51.739430_0000). The output also provides a bit of information about what types of data are collected. After the collection is finished, the `db2fodc -detect` command either stops running or executes the next iteration after sleeping for some time. The amount of time to sleep is determined by the value of the `sleeptime` option if you specify it or 1 second if you do not specify a value. Whether detection continues depends on often the threshold trigger conditions have been met at this point and how much time has passed (that is, the values of the `iteration` and `duration` options that you used. As defined in the previous example, detection and FODC collection continue until either all 10 iterations of detection are complete or the end of the threshold duration is reached.

Data collected

The diagnostic data that is collected is stored in an FODC package (a directory path). This path is created inside the path you that specified for the `FODCPATH` parameter when you configured your data server. If you did not configure the paths where diagnostic data is stored ahead of time, see the section “Be prepared: configure your data server ahead of time“ to learn about how to configure your system.

The contents of the FODC package directory path might look like the following example:

```
db2inst1@db2v10:~/sqllib/db2dump/FODC_Cpu_2013-07-14-15.15.40.604026_0000> ls -l
total 40
-rwxrwxrwx 1 db2inst1 db2grp1 1570 Jul 14 15:19 db2fodc.log
drwxrwxrwx 2 db2inst1 db2grp1 4096 Jul 14 17:19 DB2PD_2013-07-14.15.15.40.000000
drwxrwxrwx 5 db2inst1 db2grp1 4096 Jul 14 17:19 FODC_Perf_2013-07-14-
15.15.44.474725_0000
drwxrwxrwx 2 db2inst1 db2grp1 4096 Jul 14 17:19 iostat_2013-07-14.15.15.40.000000
drwxrwxrwx 2 db2inst1 db2grp1 4096 Jul 14 17:19 memory_2013-07-14.15.15.40.000000
drwxrwxrwx 2 db2inst1 db2grp1 4096 Jul 14 17:19 netstat_2013-07-14.15.15.40.000000
drwxrwxrwx 2 db2inst1 db2grp1 4096 Jul 14 17:19 ps_2013-07-14.15.15.40.000000
drwxrwxrwx 2 db2inst1 db2grp1 4096 Jul 14 17:19 vmstat_2013-07-14.15.15.40.000000
```

Data analysis

An analysis of the output of the `vmstat` command (FODC_Cpu_2013-07-14-15.15.40.604026_0000/vmstat_2013-07-14.15.15.40.000000/db2v10.vmstat.out) shows a combined user and system processor usage rate of 100% , which in turn triggered the FODC collection:

```
procs -----------memory---------- ---swap-- -----io---- -system-- -----cpu------
 r  b   swpd   free  inact active   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 826496 658380 507952    0    0     0     0  404  338 98  2  0  0  0
 1  0      0 826496 658380 507952    0    0     0     0  394  352 99  1  0  0  0
 1  0      0 826496 658380 507952    0    0     0     0  377  298 100 0  0  0  0
 1  0      0 826496 658380 507952    0    0     0     0  394  330 99  1  0  0  0
 1  0      0 826496 658380 507952    0    0     0     0  382  311 100 0  0  0  0
 2  0      0 851208 658380 483632    0    0     0     0  373  400 97  3  0  0  0
 3  0      0 851812 658420 483064    0    0     0     0  384  620 87 13  0  0  0
 2  0      0 852316 658460 482464    0    0     0     0  376  604 90 10  0  0  0
 2  0      0 852564 658504 481700    0    0     0     0  380  567 91  9  0  0  0
 2  0      0 843388 658556 491224    0    0     0    28  371  584 91  9  0  0  0
 2  0      0 844636 658616 490052    0    0     0   676  411  570 89 11  0  0  0
 1  0      0 852836 658632 481424    0    0     0     0  389  409 98  2  0  0  0
 2  0      0 852836 658636 481424    0    0     0     0  361  317 98  2  0  0  0
 1  0      0 852836 658632 481428    0    0     0     0  354  282 100 0  0  0  0
 1  0      0 852836 658632 481428    0    0     0     8  345  317 100 0  0  0  0
 1  0      0 852836 658636 481428    0    0     0     0  400  324 100 0  0  0  0
```

Now, investigate the cause of these high processor usage rates. To narrow down the cause, you must use both the stack trace log and the output of the `db2pd` command.

Two stack trace logs are created during the FODC collection. These indicate the top DB2 consumers of processor resources, in descending order over an interval of 30 seconds. The information that is given is for the top coordinator agents (db2agents), which perform all database requests on behalf of the application.

Here are the top coordinator agents from the stack trace log (FODC_Cpu_2013-07-14-15.15.40.604026_0000/FODC_Perf_2013-07-14-15.15.44.474725_0000/StackTrace.0075/StackTrace.log.0):

```
List of 20 top db2agent (db2ag*) EDUs:
54
57
53
73
77
```

```
66
72
70
67
69
75
100
103
98
111
106
105
101
99
104
```

Look for one or several coordinator agents that use significantly more processor resources than other agents use, which gives you a clue for the next step. In this output, db2agent EDU 54 looks promising, based on the amount of processor resources that it used:

```
...
54          2863655792      9135            db2agent (SAMPLE) 0             130.200000      1.170000
57          2643454832      9193            db2agent (idle) 0                 2.040000      1.070000
53          2864704368      9109            db2agent (idle) 0                 1.880000      0.550000
...
```

You can see that db2agent EDU 54 uses far more resources than the next two coordinator agents use. The other stack trace log (which is not shown but looks very similar to the previous sample output) also shows db2agent EDU 54 at the top of the list.

The db2agent number by itself is only an intermediate bit of information and is not useful by itself. You can use this information to gain additional insight into the application that the db2agent is working for, though. Look at the output folder of the `db2pd` command and see whether you can correlate the db2agent number with a specific application ID (alternatively, use the snapshot output for the same purpose). Several `db2pd` command output files are created during FODC, each showing similar output; you need the information from only one of them. Searching for the coordinator agent 54 in one of the `db2pd` command output files (FODC_Cpu_2013-07-14-15.15.40.604026_0000/DB2PD_2013-07-14.15.15.40.000000) yields the following result:

```
Address    AppHandl [nod-index] AgentEDUID Priority    Type     State       ClientPid Userid   ClientNm ...
0x13A37F80 195      [000-00195] 54          0           Coord    Inst-Active 8724      db2inst1 db2bp    ...
```

Note the process ID, 8724. This process ID is another important clue and gets you closer to determining the query statement that is the likely culprit behind the spikes in processor usage. All you have to do now is to search for additional occurrences of the

same process ID in the db2pd command output. The process ID leads you to the client application that originated the query and the query statement.

```
Application :
  Address :                0x13BB0060
  AppHandl [nod-index] :   195      [000-00195]
  TranHdl :                3
  Application PID :        8724
  Application Node Name :  db2v10
  IP Address:              n/a
  Connection Start Time :  (1373840099)Sun Jul 14 15:14:59 2013
  Client User ID :         db2inst1
  System Auth ID :         DB2INST1
  Coordinator EDU ID :     54
  Coordinator Member :  0
  Number of Agents :       1
  Locks timeout value :    NotSet
  Locks Escalation :       No
  Workload ID :            1
  Workload Occurrence ID : 1
  Trusted Context :        n/a
  Connection Trust Type :  non trusted
  Role Inherited :         n/a
  Application Status :     UOW-Executing
  Application Name :       db2bp
  Application ID :         *LOCAL.db2inst1.130714221459
  ClientUserID :           n/a
  ClientWrkstnName :       n/a
  ClientApplName :         CLP longquery.db2
  ClientAccntng :          n/a
  CollectActData:          N
  CollectActPartition:     C
  SectionActuals:          N

  List of active statements :
   *UOW-ID :          1
    Activity ID :     1
    Package Schema :  NULLID  #
    Package Name :    SQLC2J24
    Package Version : #
    Section Number :  201
    SQL Type :        Dynamic
    Isolation :       CS
    Statement Type :  DML, Select (blockable)
    Statement :       SELECT COUNT(*) FROM
    SYSCAT.TABLES, SYSCAT.TABLES, SYSCAT.TABLES, SYSCAT.TABLES,
SYSCAT.TABLES
```

The culprit is the application longquery.db2, which issues an expensive SELECT statement whenever it is run. In this case, you could also have used the coordinator agent number 54 to find the query statement directly, without looking up the process ID. This works here because the coordinator EDU ID is the same as the db2agent EDU ID or coordinator agent. There are likely cases where a direct look-up using only the

coordinator agent does not work, so it is useful to be able to correlate a coordinator agent with a process ID in the `db2pd` command output.

Resolving the problem

In this case, the users complaining of an intermittent performance slowdown are not affected by an issue with the application that they are using. Instead, they are affected by another query that is being run against the DB2 server from time to time. This other query turns out to be very expensive because it impacts the response times for everyone else.

How can you address the impact of this other query? You might be able to rewrite the query so that it becomes less expensive to run. Alternatively, you can use some standard DB2 workload management practices to run the query in a more controlled fashion, without using excessive system resources.

There might be cases where it is not easy to determine the cause of a problem. Even if you cannot resolve an issue yourself, you can set up an FODC threshold rule to collect the required diagnostic data for different system resources, which you can then provide to IBM Support for further analysis. IBM Support needs the diagnostic data to be able to help, especially with intermittent problems. If you have the diagnostic data ready, you can reduce the amount of time that it takes to diagnose the underlying issue.

Scenario: Troubleshooting sort overflows

Identifying the problem

Users report a significant increase in query run times, which you are asked to investigate.

A general performance slowdown that is perceived by users can have many different causes. In this case, the focus is on the performance impact that a large number of sort overflows, also known as sort spills, can cause. If you don't know whether sort overflows are a problem on your system, use this scenario to find out.

Queries often require a sort operation. A sort is performed when no index exists that would satisfy the sort order or when an index exists, but sorting is determined to be more efficient. Sort overflows occur when an index is so large that it cannot be sorted in the memory that is allocated for the sort heap. During the sort overflow, the data to be sorted is divided into several smaller sort runs and stored in a temporary table space. When sort overflows that are stored in the temporary table space also require writing to disk, they can negatively impact the performance of your data server.

Diagnosing the cause

You can use the `MON_GET_PKG_CACHE_STMT` table function to determine whether a sort overflow occurred. Try a query such as the following one, which returns not only information about sorts but also specifies the related SQL statements:

```
DB2 SELECT TOTAL_SORTS, SORT_OVERFLOWS, TOTAL_SECTION_SORT_TIME,
SUBSTR(STMT_TEXT,1,30) AS STMT_TEXT FROM TABLE (MON_GET_PKG_CACHE_STMT('D',
NULL, NULL, -1))

TOTAL_SORTS          SORT_OVERFLOWS       TOTAL_SECTION_SORT_TIME STMT_TEXT
-------------------- -------------------- ----------------------- ------------------------------
                   0                    0                       0 SELECT POLICY FROM SYSTOOLS.P
                   0                    0                       0 UPDATE SYSTOOLS.HMON_ATM_INFO
                   0                    0                       0 SELECT COLNAME, TYPENAME FROM
                   1                    0                       0 SELECT IBM.TID, IBM.FID FROM
                   0                    0                       0 UPDATE SYSTOOLS.HMON_ATM_INFO
                   0                    0                       0 SELECT STATS_TIME, INDEXTYPE
                   0                    0                       0 SELECT TABNAME FROM SYSCAT.TA
                   0                    0                       0 UPDATE SYSTOOLS.HMON_ATM_INFO
                   0                    0                       0 SELECT TRIGNAME FROM  SYSCAT.
                   0                    0                       0 LOCK TABLE SYSTOOLS.HMON_ATM_
                   0                    0                       0 SELECT CREATE_TIME FROM SYSTO
                   1                    1                 1975263 select c1 from t1 order by c1
                   0                    0                       0 SELECT COUNT( * ) FROM SYSTOO
                   0                    0                       0 CALL SYSPROC.SYSINSTALLOBJECT
                   0                    0                       0 UPDATE SYSTOOLS.HMON_ATM_INFO
                   2                    0                       1 DELETE FROM SYSTOOLS.HMON_ATM
                   0                    0                       0 UPDATE SYSTOOLS.HMON_ATM_INFO
                   0                    0                       0 CALL SYSINSTALLOBJECTS( 'DB2A
                   0                    0                       0 SELECT STATS_LOCK, REORG_LOCK
                   0                    0                       0 SELECT CREATOR, NAME, CTIME F
                   0                    0                       0 UPDATE SYSTOOLS.HMON_ATM_INFO
                   0                    0                       0 SET CURRENT LOCK TIMEOUT 5
                   0                    0                       0 SELECT TABNAME FROM SYSCAT.TA
                   0                    0                       0 SELECT total_sorts, sort_over
                   0                    0                       0 SELECT COUNT( * ) FROM SYSTOO
                   0                    0                       0 SELECT ATM.SCHEMA, ATM.NAME,

  26 record(s) selected.
```

Look at the SORT_OVERFLOWS column in the output; any nonzero value indicates that a query performed a sort operation that spilled over to disk. In this example, there is one SELECT statement on table T1 that resulted in a sort overflow. Also, the TOTAL_SECTION_SORT_TIME column for the same statement indicates that the section sort operation took a very long time. A section in this context is the compiled query plan that was generated by the SQL statement that was issued. The unit of measurement is milliseconds; when converted, the total sort time is around 32 minutes, which makes the query long running.

The `MON_GET_PKG_CACHE_STMT` table function can return other useful columns that you can include in your query. For example, the NUM_EXECUTIONS column can show how often a statement was executed. It might also be useful to return more of the statement text by modifying the sample query. For more information about metrics, see "MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache"

(http://pic.dhe.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.sql.rtn.doc/doc/r0055017.html).

The db2pd command is also useful in this context, because you can investigate sort performance while a perceived query performance problem is happening. You do not have to wait for the monitoring information to be updated before you can begin troubleshooting the problem. This feature is helpful if your queries are very long running, as in the example that is used here.

To monitor sort performance with the db2pd command, use the parameters in the following example:

```
db2pd -d sample -sort -app -dyn
```

The following sample output is abridged to highlight how you can determine whether sort overflows are happening and what applications and SQL statement are involved:

```
Database Member 0 -- Database SAMPLE -- Active -- Up 2 days 01:36:58 -- Date 04/22/2013 20:34:29
AppHandl [nod-index]
950      [000-00950]
   SortCB     MaxRowSize           EstNumRows EstAvgRowSize NumSMPSorts NumSpills
   0xE7750430 133                  109155368  136           1           187711
   KeySpec
   CHAR:128

      SMPSort# SortheapMem          NumBufferedRows NumSpilledRows
      0        16                   331             62132341

Applications:
Address    AppHandl [nod-index] ...  Status         C-AnchID ... Appid
0xF15F0060 950      [000-00950] ...  UOW-Executing  542      ... *LOCAL.DB2.130420233443

Dynamic SQL Statements:
Address    AnchID StmtUID    NumEnv  ...  NumRef     NumExe     Text
0xE711F560 542    2          1       ...  2          2          select c1 from t1 order by c1
```

In this case, the NumSpills column indicates that there are sort overflows. The NumSpilledRows column shows that these sort overflows resulted in writing a large number of rows to disk.

To determine the application and SQL statement, first use the ApplHandl value, 950 in this example, to locate the application information, and note the value in the C-AnchID column, here 542. Next, locate the same C-AnchID value in the output for SQL statements to find the statement text.

Resolving the problem

If the ratio of sort overflows to total sorts is quite high, you might need to change the value of the sortheap database configuration parameter to make more memory available for sort operations. You might also consider enabling self-tuning memory to allow the DB2 system to adjust the sortheap memory automatically. In DB2 Version

9.7 and later, the value of the `sortheap` parameter defaults to `AUTOMATIC`, which enables automatic tuning of the memory that is required for sort operations. If you are using an earlier DB2 version or if you determine by monitoring sort durations and sort overflows that the automatic tuning is not always sufficient, you can tune the value of the `sortheap` parameter manually. For example, sorting 446,000 records with a record length of 128 bytes requires 57,088,000 bytes of insert-buffer memory (446,000 x 128 bytes), equivalent to 14,000 four KB pages. Insert-buffer memory makes up approximately 50% of sort heap memory internally, so this figure of 14,000 four KB pages must be doubled to arrive at an approximate value for sort heap memory. Doubling the number of pages gives a suggested `sortheap` parameter setting of 28,000 four KB pages.

The `sortheap` parameter has a relationship with the `sheapthresh` and `sheapthres_shr` parameters. If you modify the `sortheap` parameter setting, also modify the value of the `sheapthresh` parameter to maintain sufficient sort parallelism. If you set both the `sortheap` and `sheapthresh_shr` parameters to automatic, the self-tuning memory manager (STMM) can keep these settings in tune with the current workload. Alternatively, you can generally enable self-tuning memory for the database by using the `self_tuning_mem` database configuration parameter, which tunes several parameters affecting memory usage, including for sort heap memory. In partitioned database environments, some additional considerations apply when you use self-tuning memory. For information, see "Self-tuning memory in partitioned database environments" (http://pic.dhe.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.admin.perf.doc/doc/c0023815.html).

There might be no way to avoid sort overflows by increasing the value of the `sortheap` parameter, because of the amount of memory that is required. However, you can still take some actions to minimize the impact of sort overflows. Ensure that the buffer pool for temporary table spaces is large enough to minimize the amount of disk I/O that sort overflows cause. Furthermore, to achieve I/O parallelism during the merging of sort runs, you can define temporary table spaces in multiple containers, each on a different disk. To assess how well temporary data is used in the buffer pool, use the `db2pd` command with the `-bufferpool` parameter. A section of the output shows the cache hit ratios of temporary table space data and indexes.

If more than one index is defined on a table, memory usage increases proportionally, because the sort operation keeps all index keys in memory. To keep memory usage to a minimum, create only the indexes that you need.

Scenario: Troubleshooting locking issues

This scenario illustrates how to use IBM InfoSphere Optim Performance Manager (OPM) to investigate the causes of lock wait problems in a DB2 system.

Before you can use OPM to its full potential to investigate and diagnose the cause of any performance issue, you must establish and save a performance baseline in OPM. OPM can then compare the baseline with the current metrics and highlight potential

issues when certain metrics vary from the established baseline. For more information about setting a baseline with OPM, see the OPM overview dashboard help.

Identifying the problem

You observe that your DB2 system is not processing the expected number of transactions, even though there doesn't appear to be a bottleneck with the CPU or disk. A typical symptom that you might encounter is a high average lock wait time that is accompanied by low CPU usage.

Figure 1 illustrates such an example. The highlighted section of the OPM Workload dashboard provides valuable information to help you determine whether the performance problem is due to lock wait issues.

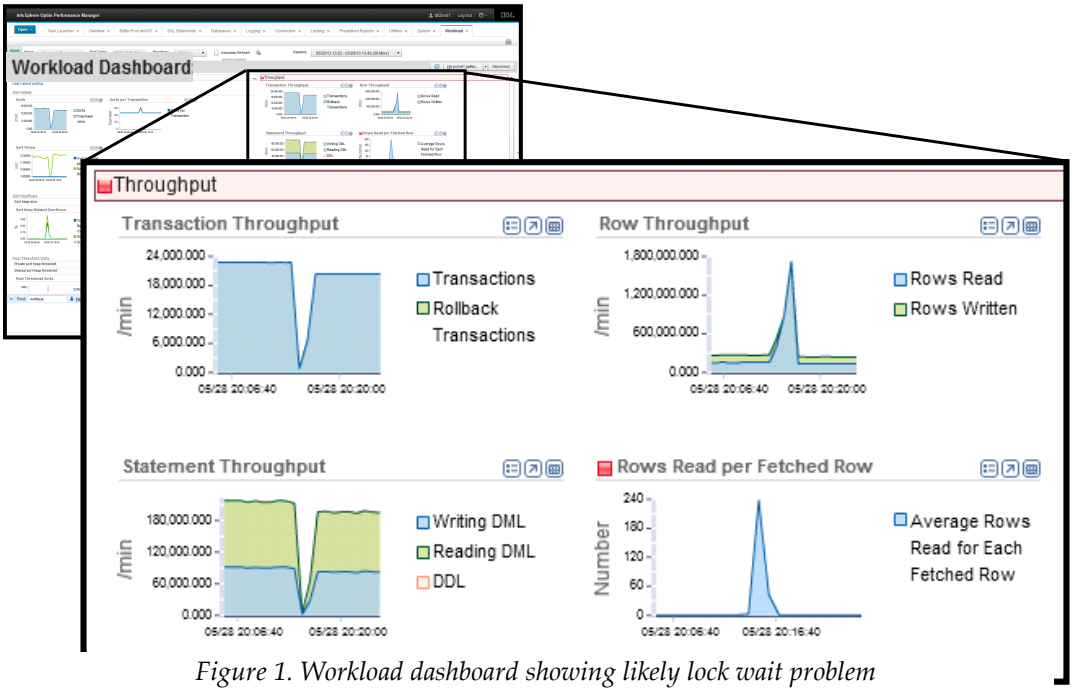

Figure 1. Workload dashboard showing likely lock wait problem

The Transaction Throughput and Statement Throughput graphs show that the system was originally working fine with good throughput. At a specific time, the system experienced a very significant drop in throughput, almost to zero. However, there was still some activity on the system during that period, as shown by the Row Throughput and Rows Read per Fetched Row graphs. They show that a high number of rows were read even though almost no transactions were completed. These symptoms suggest that one transaction might have held locks and blocked most other transactions from executing. These symptoms might also indicate that a very large query was reading a very large number of rows. To diagnose the cause, you must investigate further.

You can configure OPM to monitor locking events and notify you when particular events occur or exceed a threshold. You can use the Locking configuration dialog,

which is shown in Figure 2, to monitor certain conditions and control the level of detail that is collected for lock events.

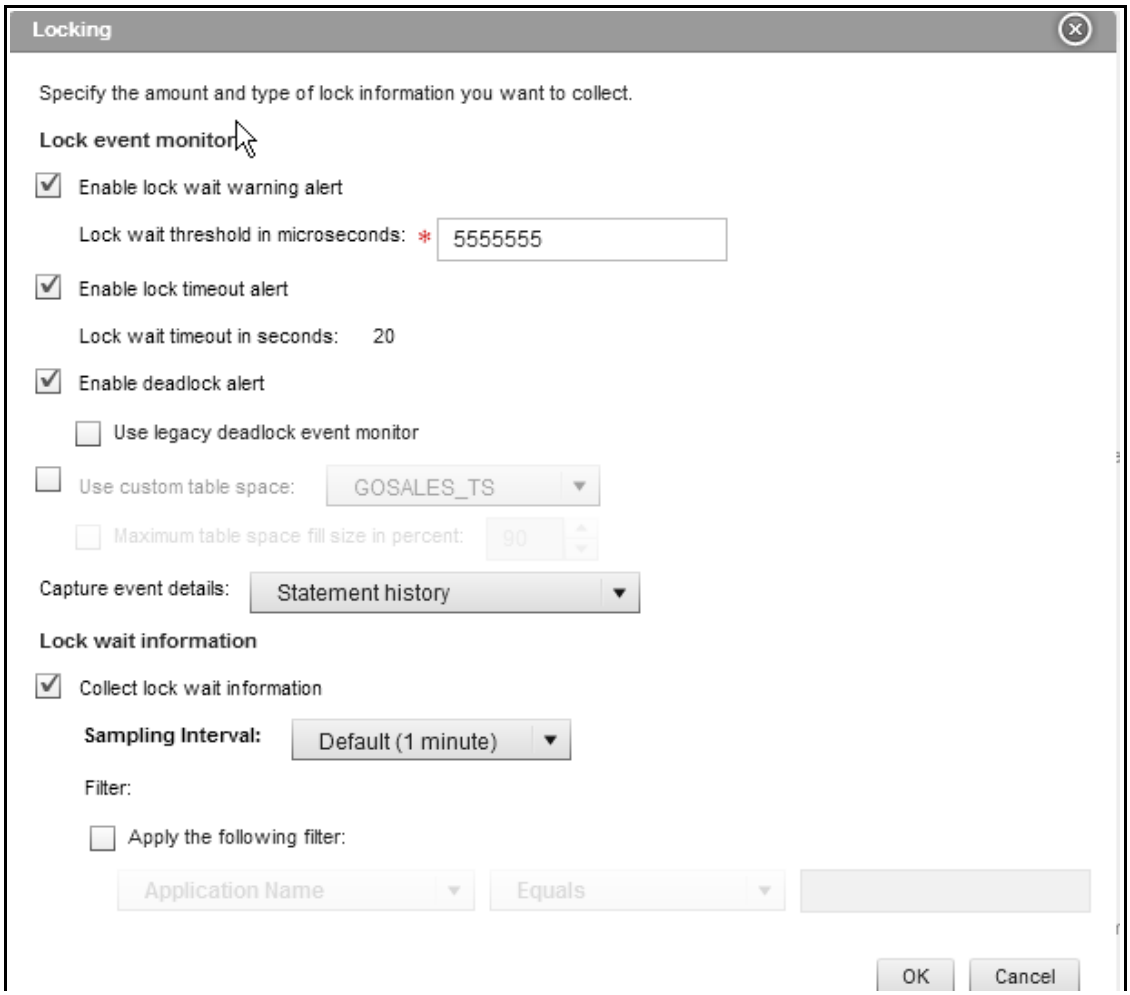

Figure 2. You can use the Locking configuration dialog to specify locking alerts and the amount of detail to collect for locking events

Diagnosing the cause

You can use the OPM Overview dashboard to help investigate the symptoms more closely. Figure 3 highlights important information that helps you understand the problem better.

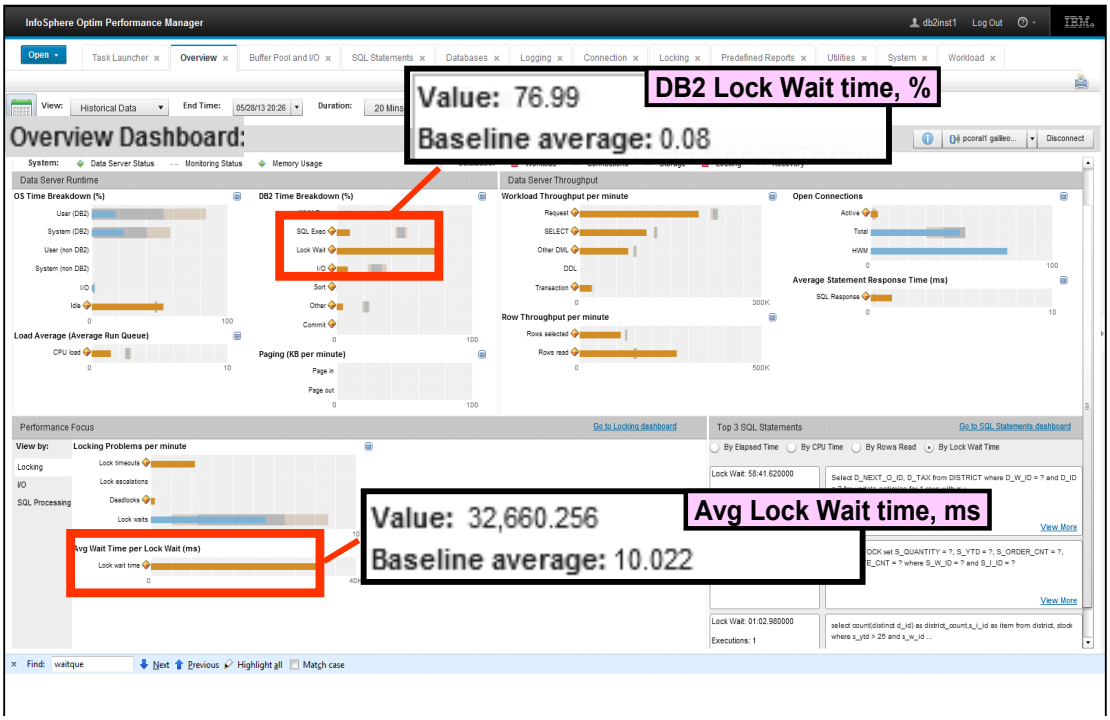

Figure 3. Overview dashboard highlighting large lock wait times

The Overview dashboard displays average values across the selected Time Slider interval, for a wide range of metrics. Two are of particular interest in this scenario. Both DB2 Lock Wait Time and Average Lock Wait Time metrics show significant increases from the baseline. These increases provide further evidence of a locking problem.

You can investigate the problem further with the Locking dashboard. You can access the Locking dashboard from the Overview dashboard. The Locking dashboard provides detailed information for all locking events on a separate tab. Figure 4 highlights a section of the Locking dashboard showing locking events.

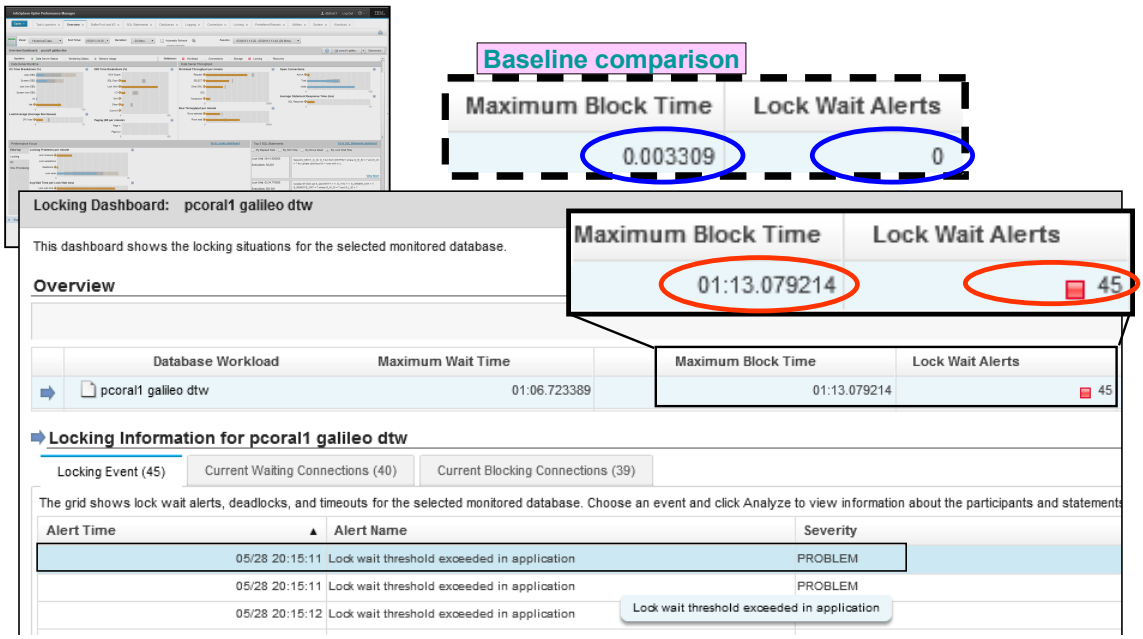

Figure 4. Locking dashboard highlighting large values of Lock Wait Alerts and Block Time

The current **Maximum Block Time** and **Lock Wait Alerts metrics** are of particular interest when you compare them with the baseline, which is shown in the dashed border box in the figure. OPM has recorded a much higher number of lock wait alerts than is typical. The baseline shows zero lock wait alerts and a very short maximum block time.

You can investigate the problem further by selecting an individual lock timeout event from the dashboard and displaying detailed information for it. Figure 5 highlights some of the key information that is displayed about the lock event after you double-click to select it.

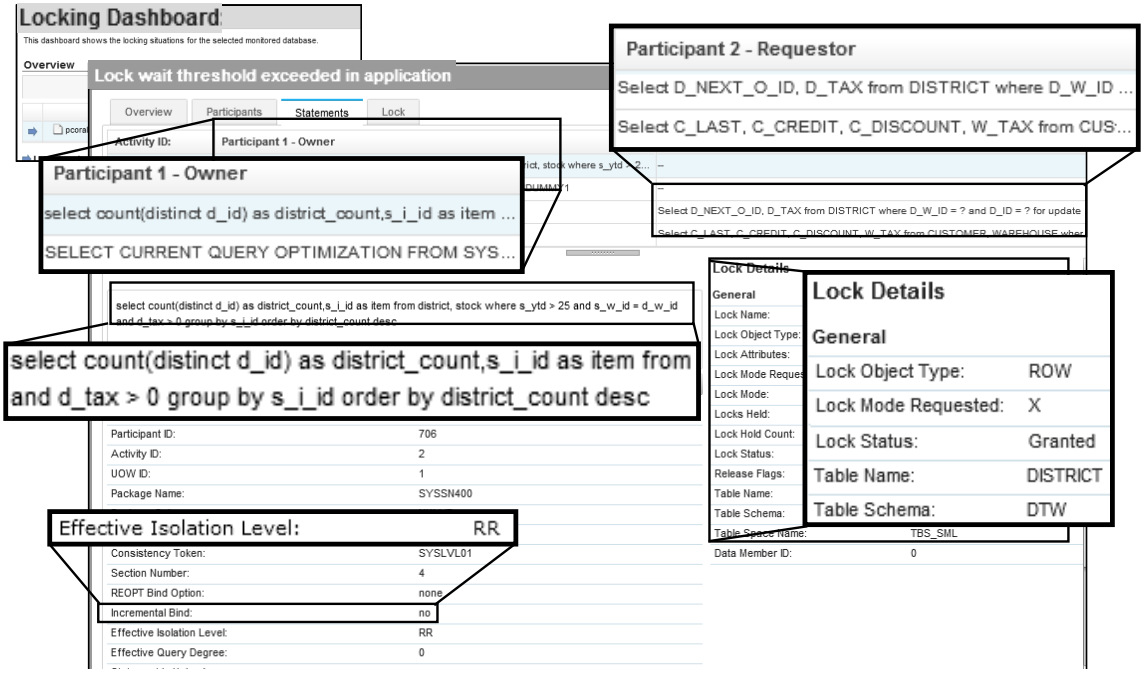

Figure 5. Analyzing the details of a lock timeout alert

The lock timeout event details show information for both participants in the lock event: the owner of the lock and its requestor. To see the lock owner's SQL statement, select the Statements details. The information includes the complete text of the SQL statement, the details of the lock that is being held, and the isolation level of the transaction. In this example, the isolation level is **r**epeatable read (RR). This is the likely cause of the multiple lock timeout events and slowdown in transaction throughput. A transaction using the RR isolation level can hold a large number of locks during a unit of work (UOW) and cause many other transactions to be blocked, waiting for locks to be released.

Resolving the problem

One possible resolution for the problem that is described in this scenario is to determine whether you can modify the application so that it does not use the RR isolation level. This change would reduce the number of locks that the application holds at one time, therefore reducing the likelihood of lock contention with other applications.

A DBA typically returns locking problems to the application team for resolution, but often, it is a challenge is to prove that the delays are caused by locking issues. Now the DBA can have proof of the locking delays and details for both the blocking application and the blocked application and their SQL statements.

An alternative way to diagnose a locking issue is based on alerts, if you configure OPM to monitor locking events and report alerts, as shown in Figure 2. The health summary indicates databases that have active alerts. For example, in Figure 6, you can

see that a locking alert was issued for the database named “Kepler” (highlighted with a red box).

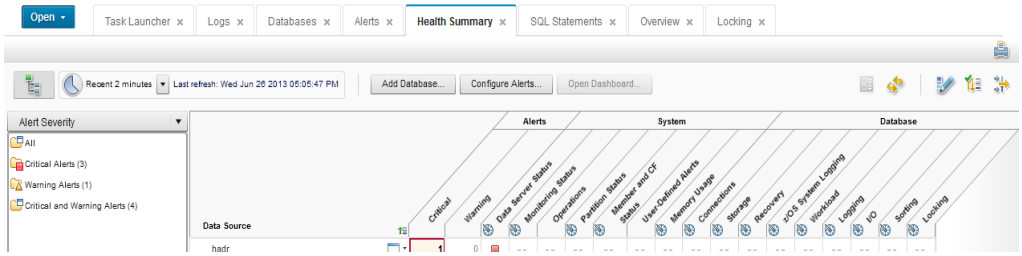

Figure 6. OPM health summary shows locking alerts

When you click the red icon, it opens the locking alert list for the database. If you select a specific alert, you can see the details of the alert, as shown in the following example.

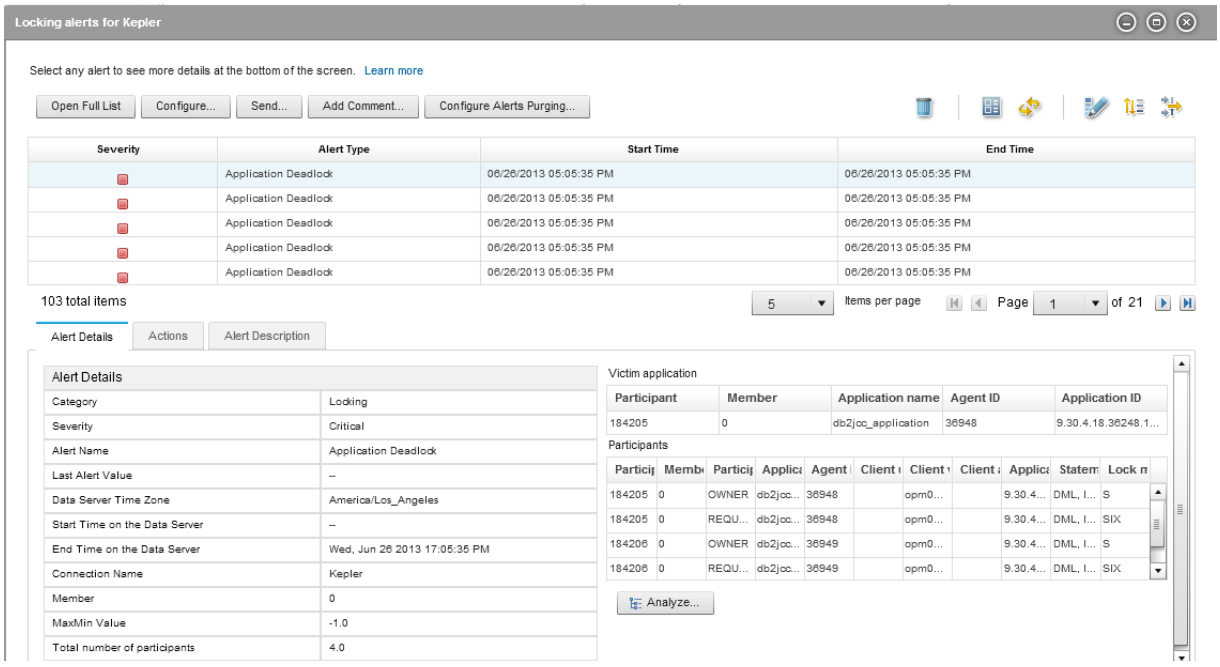

Figure 7. Locking alert list and details

To drill down into the full details for this event, click **Analyze** .

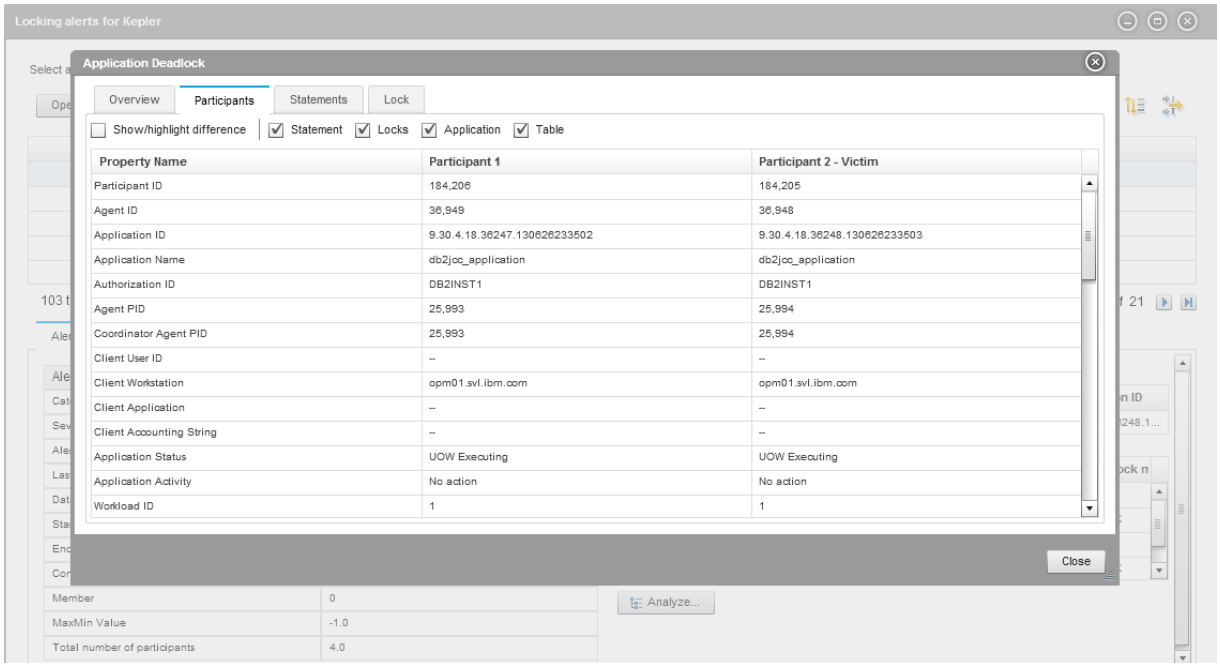

Figure 8. Locking event details

The event details window provides more information about each participant in the event. This information helps you pinpoint the cause of the problem and determine a course of action to correct the issue.

Best practices

- Be prepared by configuring your data server before problems might occur:
    - Redirect diagnostic data away from the DB2 installation path.
    - For greater resilience, configure an alternate diagnostic path.
    - Redirect core file dumps and FODC data to a different directory.
    - Configure for rotating diagnostic and administration notification logs.
    - Set up a process to archive and delete diagnostic data regularly.
    - Provide enough free space to store diagnostic data.
- Minimize the impact of diagnostic data collection:
    - Collect data as locally to the problem as possible.
    - Collect only the diagnostic data that you need.
- Use the monitoring infrastructure to gain an understanding of the typical workloads that your data server processes, so that you can tell when atypical events are happening.
- Use the scenarios in this paper as examples for how you can use the various troubleshooting and monitoring tools.
- Know when you are faced with a problem that you cannot resolve on your own and therefore must engage with IBM for technical support.

Conclusion

The trend is toward more granular diagnostic data collection, especially on large database systems. On these systems, end-to-end diagnostic data collection is often too expensive and carries the risk of affecting database availability. To lessen the impact of diagnostic data collection, DB2 tools such as FODC can collect data about ongoing problems locally and selectively.

To prepare for a possible problem, it is important that you configure your data server before problems might occur. Troubleshooting is much easier if the data is readily available and the impact to the performance of the system is well controlled.

Part of being prepared also means knowing the typical workloads that your data server processes. If you understand your typical workloads, you are much more likely to know quickly when an atypical event might be happening that requires further investigation.

Further reading

- IBM Information Management Best Practices website (www.ibm.com/developerworks/db2/bestpractices)
- *Tuning and Monitoring Database System Performance* (www.ibm.com/developerworks/data/bestpractices/systemperformance)
- IBM DB2 Version 10.5 Information Center (pic.dhe.ibm.com/infocenter/db2luw/v10r5/index.jsp)

Contributors

Dmitri Abrashkevich, DB2 Development, IBM

Albert Grankin, Senior Technical Staff Member, IBM

Bill Peck III, DB2 Advanced Support, IBM

Maira Teixeira De Melo, L2 Technical Support, IBM

Contacting IBM

To provide feedback about this paper, write to db2docs@ca.ibm.com.

To contact IBM in your country or region, see the IBM Directory of Worldwide Contacts at http://www.ibm.com/planetwide.

To learn more about IBM Information Management products, see to http://www.ibm.com/software/data/.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

Without limiting the above disclaimers, IBM provides no representations or warranties regarding the accuracy, reliability or serviceability of any information or recommendations provided in this publication, or with respect to any results that may be obtained by the use of the information or observance of any recommendations provided herein. The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any recommendations or techniques herein is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. People attempting to adapt these techniques to their own environment do so at their own risk.

This document and the information contained herein may be used solely in connection with the IBM products discussed in this document.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems.

Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.