IBM DB2® Universal Database™
DB2 Problem Determination Tutorial Series

Combining DB2 and OS Diagnostics

Table of Contents

Combining DB2 and OS Diagnostics 3
- About this tutorial 3
  - Tutorial objectives 3
  - Audience and assumptions 3
  - Pre-tutorial setup 4
  - Tutorial conventions used 4
  - About the author 5
- Understanding the environment 6
  - Introduction to system configuration and user environment information 6
  - Relevant DB2 configuration information 8
  - Obtaining configuration information on specific platforms 9
- Diagnostic and event logs 11
  - Correlating DB2 and system events or errors 11
  - System error and message logs on UNIX® 12
  - Windows® event logs and the Dr. Watson log 12
  - Identifying and researching operating system errors 13
- Operating system error examples on UNIX 15
  - Hitting a file size limit 15
  - Per-process IPC limits 16
  - System-wide IPC limits 17
- Defining a performance problem, initial data collection 19
  - Clarifying the problem 19
  - System diagnostics that can be performed quickly 20
  - DB2 diagnostics that can be performed quickly 21
  - DB2 function call stacks, trap files, and tracing 22
  - Formatting stack dump files on Windows 23
  - Using perfmon on Windows 24
  - Avoiding pitfalls and a first glance at system data 28
  - DB2 snapshot elements associated with system resource usage 29
- Performance problem examples 31
  - Too much system CPU 31
  - Too much I/O 31
  - Too much memory 33
- Summary 35
  - Where to go from here 35

Combining DB2 and OS Diagnostics

About this tutorial

Tutorial objectives

The objectives of this tutorial are to guide you through the initial steps in defining and diagnosing system errors and performance problems that involve DB2. Specifically, you will learn the following things:

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

Topics include understanding configuration, performance and resource monitoring, and basic diagnostic log analysis.

The following platforms are covered:

- AIX
- Solaris
- HP-UX
- Linux
- Windows

Audience and assumptions

This tutorial is for anybody who reports or analyzes DB2 problems where at least one of these conditions applies:

- The operating system or user environment may be a factor
- There is an issue with a resource managed by the operating system, such as CPU, memory, interprocess communication (IPC) resources, or disks

In order to take this tutorial, it is assumed that you have the following skills:

- Are familiar with the basic concepts of an operating system - including RAM, CPU, virtual memory, IPC resources, and I/O
- Are familiar with basic operating system commands
- Understand DB2 concepts including bufferpools, tablespaces, sorting, parallelism, the idle agent pool, and DB2 process names

Prior to taking this tutorial, it is suggested that you review the following chapters from the *DB2 v7.2 Administration Guide*:

Chapter 4 - Parallel Database Systems
Chapter 20 - Elements of Performance
Chapter 21 - Architecture and Processes Overview
Chapter 27 - Operational Performance
Chapter 32 - Configuring DB2
Appendix M - Using the Windows NT® Performance Monitor

The *Administration Guide* can be viewed online at the DB2 Technical Support Web site:
http://www-3.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/document.d2w/report?fn=db2v7d0frm3toc.htm

Pre-tutorial setup

DB2 Enterprise Edition, version 7.2 FixPak or greater should be installed on the platforms you have available. Some examples illustrate platform-specific behaviour or tools on AIX, Linux, Solaris, and Windows. It is possible to work through some of the UNIX examples on any of the supported UNIX platforms, though specific commands and their output will differ.

The SAMPLE database should be created with the **db2sampl** command.

You should have administrator privileges on the systems where you are working. If administrator privileges cannot be obtained, there are a few exercises that you might not be able to complete.

Note: Some examples in this tutorial will create stress on the system, so other users may be affected. The system may also become unstable.

Tutorial conventions used

When a tool or utility is first mentioned it will be shown in **bold** text.

All commands and their output will be shown in a `monospace` font.

Some examples will show specific command options that may change over time, but these changes will always be documented in DB2 UDB documentation.

About the author

Michael Cornish is a senior technical analyst with the DB2 UDB Support Team. He specializes in memory-related issues, system performance, and solving problems which cross the boundary between DB2 and operating systems.

You can reach Michael by locating his e-mail address in the *IBM Global Directory* at http://www.ibm.com/contact/employees/us .

Understanding the environment

Introduction to system configuration and user environment information

Diagnosing some problems related to memory, swap files, CPU, disk storage, and other resources requires a thorough understanding of how a given operating system manages these resources. At a minimum, defining resource-related problems requires knowing how much of that resource exists, and what resource limits may exist per user. (The relevant limits are typically for the user ID of the DB2 instance owner.)

Here is some of the important configuration information that you need to obtain:

- Operating system patch level, installed software, and upgrade history
- Number of CPUs
- Amount of RAM
- Swap and file cache settings
- User data and file resource limits and per user process limit
- IPC resource limits (message queues, shared memory segments, semaphores)
- Type of disk storage (for example EMC, Shark, Network Access Storage solution)
- What else is the machine used for? Does DB2 compete for resources?
- Where does authentication occur?

Most platforms have straightforward commands for retrieving resource information. As of FixPak 4 on DB2 version 7.2, the **db2support** utility collects this data and much more. In fact, The detailed system output of db2support (in the detailed_system_output.html file) contains the syntax for many of the commands which are used in this tutorial.

The first exercise in this tutorial teaches you how to run the db2support utility. Subsequent exercises cover trap files, which provide more DB2-generated data that can be useful in understanding the user environment and resource limits.

EXERCISE: Running the db2support command

1. On each platform that you have available, start DB2 with the **db2start** command.
2. Assuming you already have the SAMPLE database available, create a directory for storing the output from db2support.
3. Change to that directory and issue:
    - `db2support <directory> -d SAMPLE`
4. Review the console output, especially the types of information that are collected. You should see output like this:

```
Collecting "System files"
    "SQLDBCON"
    "db2rhist.asc"
    "SQLOGCTL.LFH"
    "SQLBP.1"
    "SQLBP.2"
    "SQLSPCS.1"
    "SQLSPCS.2"
```

```
            "db2systm"
            "db2cli.ini"
            "sqldbdir"
            "sqldbbak"
            "sqldbins"
            "sqlnodir"
            "sqlnobak"
            "archive.log"
        Collecting "Basic operating system and hardware information"
        Collecting "System resource info (disk, CPU, memory)"
        Collecting "Operating system and level"
        Collecting "JDK Level"
        Collecting "DB2 Release Info"
       Etc.
```

5. Now use a Web browser to view the detailed_system_output.html file.
   On each of your systems, identify the following information:
   - Number of CPUs
   - Amount of RAM
   - Operating system level
   - User environment
   - User resource limits (UNIX ulimit command)

EXERCISE: Locating environment information in a DB2 trap file

1. On a UNIX system, ensure DB2 is started, then issue
   ```
   db2_call_stack
   ```
   This command will generate files known as "trap" files of the form t<pid>.<node> in the DB2 diagnostic directory, where <pid> is the ID of the DB2 process to which the information pertains, and <node> is the number of the database partition. The node will be 000 for systems which are not running the Extended - Enterprise Edition product.&nbsp;
2. Locate the following in one of the trap files:
   - DB2 code level
   - `Data seg top` (this is the maximum private address space that has been required)
   - `Cur data size` (this is the maximum private address space limit)
   - `Cur core size` (this is the maximum core file limit)
   - `Signal Handlers`
   - `Environment variables`
   - `map output` (shows loaded libraries)

EXAMPLE: trap file (truncated):

```
2002-11-08-00.12.37.810012 : DB2 v7.1.0.60 s020313 SQL07024
S:AIX R:3 V:4 M:0009028F4C00 N:panipuri
mcornish.000 : db2agent (TEST) (0x1)
Signal #36
Data seg top [sbrk(0)] = 0x209C4020
Cur data size (bytes)  = 0xF424000
Cur stack size (bytes) = 0x10000000
Cur core size (bytes)  = 0x7FFFFFFF
```

```
...
Signal Handlers
         SIGABRT        : f1b09688
         SIGBUS         : default
         SIGCHLD        : f1b109b4
         SIGDANGER      : f1b09694
         SIGEMT         : default
         SIGGRANT       : f1b096ac
         SIGILL         : default
         SIGINT         : f1b096b8
...
Environment variables
DB2INSTPATH=/home/mcornish/mcornish
HOME=/home/mcornish
PWD=/home/mcornish/sqllib/db2dump
TZ=EST5EDT
DB2COMM=TCPIP
DB2INSTANCE=mcornish
```

Relevant DB2 configuration information

The most common items you need to collect for DB2 configuration are:

- Database manager configuration (`db2 get dbm cfg`)
- Database configuration (`db2 get db cfg for <database>`)
- DB2 registry (`db2set -all`)
- Node configuration (`db2nodes.cfg`)
- Bufferpool and node definitions (`SYSCAT.BUFFERPOOLS, SYSCAT.BUFFERPOOLNODES, SYSCAT.NODEGROUPDEF tables`)

These items contain some key DB2 configuration parameters that provide an indication of *potential* resource usage.

Shared memory for each active database:

- Approximately 1.1 * (bufferpools, dbheap, package cache, utility heap, locklist, and sheapthres if intra_parallel is enabled)
- Extended Storage (ESTORE) - memory for ESTORE segments + 100 bytes per ESTORE page (the number of ESTORE pages is the total amount of ESTORE memory divided by the maximum pagesize of the bufferpools which are enabled to use ESTORE)
- Address Windowing Extensions (AWE) for Windows if your bufferpools are configured to use AWE. If DB2_AWE is listed in the output of `db2set -all`, your bufferpools are using AWE.
- APP_CTL_HEAP_SZ (one per application for DB2 EEE and SMP databases)
- DB2_FORCE_FCM_BP (a registry setting which uses a shared memory segment for all database partitions if set)

Private memory for agents:

- AGENT_STACK_SZ on Windows
- DB2MEMDISCLAIM, DB2MEMMAXFREE (maximum retained free memory per agent) - DB2 registry variables
- Query heap size, sort heap, statement heap, application heap (note that these maximum sizes are not necessarily allocated concurrently, and there can be more than one sort heap allocated concurrently per agent).

Number of DB2 processes:

- MAXAGENTS, NUM_POOLAGENTS
- Prefetchers (IOSERVERS), pagecleaners (IOCLEANERS) for each database

Number of DB2 processes concurrently running:

- Number of instances, database partitions, databases
- MAXAGENTS, MAXCAGENTS, INTRA_PARALLEL, MAX_QUERYDEGREE, MAXAPPLS, DFT_DEGREE
- MAXAPPLS, DFT_DEGREE from database configuration

Obtaining configuration information on specific platforms

This table contains useful commands for obtaining configuration information.

	AIX	**HP-UX**	**Solaris**	**Linux**
OS level	/usr/sbin/oslevel	/usr/bin/uname -a	/usr/bin/uname -a	/bin/uname -a
Software	/usr/bin/lslpp -ah	/usr/sbin/swlist -v	/usr/bin/showrev -p	/bin/rpm -qa
CPUs	/usr/sbin/lsdev -C (grep proc)	/usr/sbin/ioscan (grep processor)	/usr/sbin/psrinfo -v	/bin/dmesg (grep CPU)
Memory	/usr/sbin/lsattr -El sys0 (grep realmem) /usr/samples/kernel/vmtune	/usr/sbin/dmesg (grep Physical)	/usr/sbin/prtconf (grep Memory)	/bin/dmesg (grep Memory) /usr/bin/free
IPC resources	N/A	/usr/sbin/kmtune SAM GUI tool	/usr/sbin/sysdef -i /etc/system (file)	/usr/bin/ipcs -l
User limits	ulimit -a /etc/security/limits (file)	ulimit -a	ulimit -a	ulimit -a
User environment	set	set	set	set

Windows NT has a nice interface for obtaining system information:

```
      winmsd /af
```

creates a file <MACHINE_NAME>.TXT in the current directory.

On Windows 2000® and XP®:

```
<OSdrv>:\Program Files\Common Files\Microsoft Shared\MsInfo\Msinfo32
/report
```

This command does not report on installed software. One way to obtain this information is to export the registry (at least the HKEY_LOCAL_MACHINE\SOFTWARE branch) to a file by using the **regedit** tool.

Diagnostic and event logs

Correlating DB2 and system events or errors

System messages and error logs are too often ignored. You can save hours, days, and even weeks on the time it takes to solve a problem if you take the time to perform one simple task at the initial stage of problem definition and investigation. That task is to compare entries in different logs and take note of any that appear to be related both in time and in terms of what resource the entries are referring to.

While not always relevant to problem diagnosis, in many cases the best clue is readily available in the system logs. If we can correlate a reported system problem with DB2 errors we will have often identified what is directly causing the DB2 symptom. Obvious examples are disk errors, network errors, and hardware errors. Not-so obvious are problems reported on different machines, for example domain controllers or NIS servers, which may affect connection time or authentication.

System logs can be investigated in order to assess stability, especially when problems are reported on brand new systems. Intermittent traps occurring in common applications may be a sign that there is an underlying hardware problem.

Here is some other information provided by system logs.

- Significant events such as when the system was rebooted
- Chronology of DB2 traps on the system (and errors/traps/exceptions from other software that is failing)
- Kernel panics, out-of-filesystem-space, and out-of-swap-space errors (which may prevent the system from creating/forking a new process)

System logs can help to rule out crash entries in the db2diag.log as causes for concern. One consistent DB2 crash recovery investigation was resolved by discovering that the system was rebooting - it turned out that the cleaning staff was unplugging the computer every morning at the same time!

If we see crash recovery in the db2diag.log with no preceding errors: ...

```
2002-11-11-02.52.09.031000   Instance:DB2   Node:000
PID:139(db2syscs.exe)   TID:230   Appid:*LOCAL.DB2.021111075207
base_sys_utilities  sqledint   Probe:0   Database:SAMPLE

Crash Recovery is needed.
```

... and just before we see the following Windows system log entry:

```
11/11/02    2:49:26 AM  EventLog    Information None    6005
N/A SERVER1 The Event log service was started.
11/11/02    2:49:26 AM  EventLog    Information None    6009
N/A SERVER1 Microsoft (R) Windows NT (R) 4.0 1381 Service Pack 4
```

```
      Multiprocessor Free.
```

... the DB2 crash recovery was likely a result of a Windows shutdown.

This principle of correlating information extends to logs from any source and identifiable user symptoms. For example, it can be very useful to identify and document correlating entries from another application's log even if you can't fully interpret them.

System error and message logs on UNIX®

The following table shows where the logs are located on different UNIX platforms.

	AIX	HP-UX	Linux	Solaris
Error / Message Logs	/usr/bin/errpt -a	/var/adm/syslog/syslog.log /usr/sbin/dmesg	/var/log/messages*	/var/adm/messages*

EXERCISE: Examine the logs on your system.

1. Use the table provided above to locate your system logs.
2. Search for keywords like db2 and sql.
3. Can you determine when the system was last rebooted?
4. What is the chronological ordering of the logs?
5. Search for timestamps. This is sometimes tricky due to special characters.
6. Were any DB2 traps reported? If there are none, perform the following procedures:
    a. `db2stop` (force off applications if required)
    b. `db2start`
    c. `db2 connect to sample`
    d. Identify the process ID of the db2agent servicing your connection:
    ```
    /usr/bin/ps -elf | grep $DB2INSTANCE | grep db2a
    ```
    e. Send the db2agent a signal
    ```
    kill -11 <db2agent_pid>
    db2 list tables
    kill -11 <db2agent_pid>
    ```
    f. Now examine the system logs. Is there a call stack for this trap?

Windows® event logs and the Dr. Watson log

Windows event logs:

Windows event logs can also provide useful information. While the system event log tends to be the most useful in the case of DB2 crashes or other mysterious errors related to system resources, it is worthwhile obtaining all three types of event logs:

- System
- Application
- Security

To access event logs on Windows 2000:
1. Select **Administrative Tools** from the Control Panel
2. Select the **Event Viewer**.

To access event logs on Windows NT:
- Select **Start Menu -> Programs -> Administrative Tools -> Event Viewer**

Exporting event logs:
From the event viewer, you can export event logs in two formats - in .evt format, which can be loaded back into an event viewer (for example on another machine) or in text format.
- On NT, choose **Log** from the menu and **Save As**.
- On Windows 2000, choose **Action** from the menu and **Save File As**.

Event viewer format is easy to work with since you can use the GUI to switch the chronology order, filter for certain events, and advance forwards or backwards.

Text files provide one significant advantage - you can trust the timestamps! When you export event logs in .evt format, the timestamps are in Coordinated Universal Time and get converted to the local time of the machine in the viewer. If you are not careful, you can miss key events because of time zone differences.

Text files are also easier to search, but once you load an event log from another machine into the event viewer, it is easy enough to export it again in text format.

Dr. Watson log
The Dr. Watson log, drwtsn32.log, is a chronology of all the exceptions that have occurred on the system. The DB2 trap files are more useful than the Dr. Watson log, though it can be helpful in assessing overall system stability and as a document of the history of DB2 traps. The default path is <install_drive>:\Documents and Settings\All Users\Documents\DrWatson

Identifying and researching operating system errors

A problem with a system resource may be indicated by a clear error in the db2diag.log or in some cases a signal or exception. Often low values dumped to the db2diag.log are operating system errors.

On most UNIX systems, system errors can be found in /usr/include/sys/errno.h.. Here are some of the most common:

```
#define EPERM   1       /* Not super-user              */
#define ENOENT  2       /* No such file or directory   */
#define EIO     5       /* I/O error                   */
#define ENOMEM  12      /* Not enough core             */
#define EACCES  13      /* Permission denied           */
#define ETXTBSY 26      /* Text file busy              */
#define EFBIG   27      /* File too large              */
#define ENOSPC  28      /* No space left on device     */
```

On Windows, system error header files are installed on the system with a compiler or the Windows SDK. Here are a few constants:

```
#define ERROR_FILE_NOT_FOUND             2L
#define ERROR_ACCESS_DENIED              5L
#define ERROR_INVALID_ACCESS             12L
```

You can also invoke the **net helpmsg** command on Windows, which may provide information about the error.
For example,

```
net helpmsg 1450

Insufficient system resources exist to complete
the requested service.
```

For both UNIX and Windows platforms, besides searching any problem databases you may have access to, you can use resources on the Web to research errors. Searching the following locations for the error constant (for example ENOSPC, not 28) and the operating system API being called (if known) will often turn up a few possible causes of the error:

- AIX: http://publib.boulder.ibm.com/cgi-bin/ds_form?lang=en_US
- HP-UX: http://docs.hp.com/
- Solaris: http://docs.sun.com/
- Linux: http://www.fokus.gmd.de/linux/
- Windows: http://support.microsoft.com/
- And, of course, don't forget the DB2 online support site:
  http://www.ibm.com/software/data/db2/udb/winos2unix/support

Operating system error examples on UNIX

Hitting a file size limit

It is common to hit either the user file size limit or a 2GB file limit on a file system when performing backups. Here's an example of hitting the limit on pre-allocated DB2 DMS files containers, something which shouldn't normally happen. While this is a "forced" example, user limits are sometimes altered, and occasionally DB2 is started up by an ID other than the instance ID (such as root). It is important to verify the user ID that starts DB2 along with the limits associated with that user.

1. Create a DMS tablespace of size 1000 4K pages on the SAMPLE database:
   ```
   db2 create tablespace DMS managed by database using (FILE ‘dms’
   1000)
   ```
2. Populate the tablespace with the employee table:
   ```
   db2 create table emp like employee in DMS
   db2 insert into emp select \* from employee
   db2 insert into emp select \* from emp (Repeat 9 more times)
   ```
3. Kill the instance:
   ```
   db2_kill  ( avoid this on critical databases ! )
   ```
4. Lower the user file size limit below the size of the DMS container:
   `ulimit -f 1000` (this is in 1K units)
5. Restart the instance and try to connect to the database:
   ```
   db2start
   db2 connect to sample
   ```

You should receive this following error message:

```
SQL1042C  An unexpected system error occurred.  SQLSTATE=58004
```

The db2diag.log entries should include:

```
2002-11-11-00.32.24.930242   Instance:mcornish   Node:000
PID:8226(db2pclnr)   Appid:none
oper_system_services  sqloAioReturn  ( sqlommapwrite on AIX ) Probe:36
errno: 0000 001b                                    ....
...
2002-11-11-00.32.26.249910   Instance:mcornish   Node:000
PID:8220(db2agent (SAMPLE))   Appid:*LOCAL.mcornish.021111053213
buffer_pool_services  sqlbWritePageToContainer   Probe:20
Database:SAMPLE
SMS Tablespace 3(DMS) is FULL or file is  too large (at OS or user
limit).
Detected on Container 0. ContPage= 290 Obj=4 Type=0
```

The main clue here (besides the message in the db2diag.log itself) is the error code `0x1b = 27`. Look this up in /usr/sys/include/errno.h:

```
#define EFBIG   27      /* File too large      */
```

Here's one of the top hits on a search at AIX's document site - the man pages for write():

```
EFBIG       (AIX versions 4.2 and later) An attempt was
made to write a file that exceeds the process' file size
limit or the maximum file size.
```

Per-process IPC limits

Here is one example of hitting an IPC limit on 32-bit DB2/AIX. The same scenario can be observed on other UNIX operating systems, though the number of database connections/aliases may have to be higher. On other UNIX operating systems, the kernel parameter shmsys:shmseg controls the number of connections possible from a single process. Use the **sysdef** command to check the value.

1. Catalog the sample database as 11 different aliases:
   ```
   db2 catalog db sample as sample1
   db2 catalog db sample as sample2
   Etc.
   ```
2. Set the CLP (Command-Line Process) to use type 2 connect. This allows an application to connect to more than one database within the same application.
   ```
   db2 terminate
   db2 set client connect 2
   ```
3. Try to connect to each of the 11 database aliases:
   ```
   db2 connect to sample1
   db2 connect to sample2
   Etc.
   ```

On the attempt to connect to the 11th database, you should get:

```
SQL1224N  A database agent could not be started to service a request, or
was terminated as a result of a database system shutdown or a force
command.
SQLSTATE=55032
```

The db2diag.log contains an operating system error connecting to shared memory:

```
2002-11-13-08.43.57.382554   Instance:mcornish   Node:000
PID:189336(db2bp)   Appid:*LOCAL.mcornish.021113134353
oper_system_services  sqlocshr2   Probe:200
0000 0018
```

Converting 0x18 to decimal, we get 24. In /usr/include/sys/errno.h, this is:

```
#define EMFILE  24      /* Too many open files                  */
```

Searching in problem databases or the online support site will likely reveal what causes this problem and the solution. If you know that connecting to shared memory on UNIX is done via the shmat() function, you can look up the man pages for shmat() and find the following:

```
      EMFILE   The number of shared memory segments attached to
      the calling process exceeds the system-imposed limit.
```

There are various methods of resolving this - such as implementing EXTSHM on AIX, increasing shmsys:shmseg on Solaris and HP-UX, or cataloging the database with TCP/IP loopback to avoid using shared memory segments for local connections.

System-wide IPC limits

Hangs, database manager or database startup problems, and errors when peak loads are increased are often caused by IPC resource configurations that are inadequate.

Checking IPC resource usage on systems is a quick task using combinations of the **ipcs**, **awk**, and **wc** commands. While the parameters referred to below do not apply to AIX (IPC resources are not configurable on AIX), monitoring can still be useful in determining whether excessive IPC resources are being allocated on the system.

For example, after receiving an unexpected error on a connect on a Linux system, I checked the various IPC limits with the `ipcs -l` command:

```
<-db2inst1->~==> ipcs -l

------ Shared Memory Limits --------
max number of segments = 4096
max seg size (kbytes) = 32768
max total shared memory (kbytes) = 8388608
min seg size (bytes) = 1

------ Semaphore Limits --------
max number of arrays = 128
max semaphores per array = 250
max semaphores system wide = 32000
max ops per semop call = 32
semaphore max value = 32767

------ Messages: Limits --------
max queues system wide = 1024
max size of message (bytes) = 8192
default max size of queue (bytes) = 16384
```

Then I checked the IPC resource usage on the system (`ipcs -u` will also do this on Linux):

- Shared Memory Identifiers (shmmni)
```
ipcs -m | wc -l
Output: 99 (subtract 4 for output head and tail = 95)
```

- Message Queue Identifiers (msgmni)
```
ipcs -q | wc -l
Output: 96 (subtract 4 for output head and tail = 92)
```
- Semaphore Sets/Arrays (semmni)
```
ipcs -s | wc -l
Output:132 (subtract 4 for output head and tail = 128)
```
- Semaphores on System (semmns)
```
ipcs -s|awk '{sems=sems+$5} END {print sems}'
Output: 166
```

We can see that we have hit the maximum number of semaphore arrays at 128. In this situation you would refer to the operating system documentation on how to increase semmni.

In order to get an idea of DB2's IPC resource usage, execute the ipcs command grepping for the instance owner ID after each of the following steps. Note that NSEMS may be in a different column on your UNIX platform:

```
Session 1: db2start
Session 1: db2 connect to sample
Session 2: db2 create db test
Session 2: db2 connect to test
Session 3: db2 connect to test
```

Defining a performance problem, initial data collection

Clarifying the problem

Performance problems cover a wide range of scenarios:

- Identifiable query performing slower than expected
- Workload or batch job not completing as soon as expected, reduction in transaction rate or throughput
- Overall system slowdown
- Suspected bottleneck in some type of system resource such as CPU, I/O, memory
- Query or workload consuming more resource than expected or available
- Comparison is being made between one system and another
- Query, application, DB2, or system hangs

There are some subtleties in the scenarios depicted above. For problem diagnosis purposes, it is important to clarify whether something is not meeting expectations or is exceeding resource capacity. Sometimes it is both.

For problem determination purposes, hangs can be lumped together with performance problems because many investigative strategies apply to both. In addition, it may be not be possible at first to define the problem as a hang versus a performance problem. To a user waiting for a response, a long-running job can look like a hang even if in fact much activity can be taking place on behalf of the application on the database server. There can also be a significant buildup of activity during a severe system slowdown such that all or most commands appear to hang on a system.

In addition to characterizing the problem correctly in terms of where the symptom is observed (query/application/system resource) and what is wrong with it (slowness or too much resource used), you require many other pieces of information to put the problem in context:

- When the problem started occur ring and what recent changes were identified, if any (hardware or software upgrades, new application rollout, tuning, additional users, etc.)
- How often it occurs
- Exactly what is observed and by whom
- What steps have been taken so far in terms of monitoring, tuning, and other changes
- What are the requirements - as detailed as possible
- What benchmarks have been taken
- What other problems, if any, have occurred on the system

System diagnostics that can be performed quickly

Here are some common monitoring tools that are useful on UNIX:

	AIX	**HP-UX**	**Linux**	**Solaris**
CPU	vmstat	vmstat	vmstat	vmstat mpstat
I/O	iostat	iostat -xt	iostat -t	iostat -xcn
Memory	ps aug ps -elf vmstat ipcs -a lsps -a /usr/samples/kernel/vmtune	ps -elf vmstat ipcs -a swapinfo -t	ps -aelf vmstat ipcs -a free	/usr/bin/ps -elf /usr/ucb/ps avwx vmstat ipcs -a

There are various other tools available, but these native tools are sufficient for investigating most performance problems. Note that the **iostat**, **vmstat**, and **mpstat** tools are usualy proceded by two arguments - the interval and number of iterations. For example,

```
vmstat 1 10
```

displays 10 snapshots one second apart, starting with a record that represents statistics since system startup and which can usually be ignored.

Some rules for gathering monitoring information:

1. Always capture sufficient non-filtered output. It is important to see what the system looks like as a whole. Often other software is running that may be impacting the behaviour of the system and resources available to DB2.
2. Always capture at least two iterations of data. Without two iterations of the data, some of the cumulative data is meaningless - you won't know if there is any increase in the data element at the time the problem is occurring. A measurement over some time period while the problem is occurring or leading up to the problem occurrence is essential.
3. Don't have obtrusive diagnostics like DB2 trace running while gathering monitor data.
4. Ensure the data is being gathered when the problem occurs! It is surprising how much time is spent looking at data that is not representative of the problem!
5. Gather the data with scripts that embed a timestamp in the output file. I use the uptime command for this. This makes it easier to correlate data and understand what period of time is being represented. Example:
6. `uptime >> vmstat.out ;  vmstat 1 10 >> vmstat.out`
7. Consider using the **sar** command on UNIX and the logging option of the Windows perfmon tool if ongoing monitoring is required.

On Windows, the quickest way to get information about the system is to get screen snapshots of Task Manager. For example, here is a screen snapshot of the Processes tab of Task Manager with columns providing CPU, Memory, and I/O statistics per process. (Click **View->Select Columns** to customize Task Manager.) This output is ordered by memory usage, which usually moves db2syscs.exe, the database instance process, towards the top of the output.

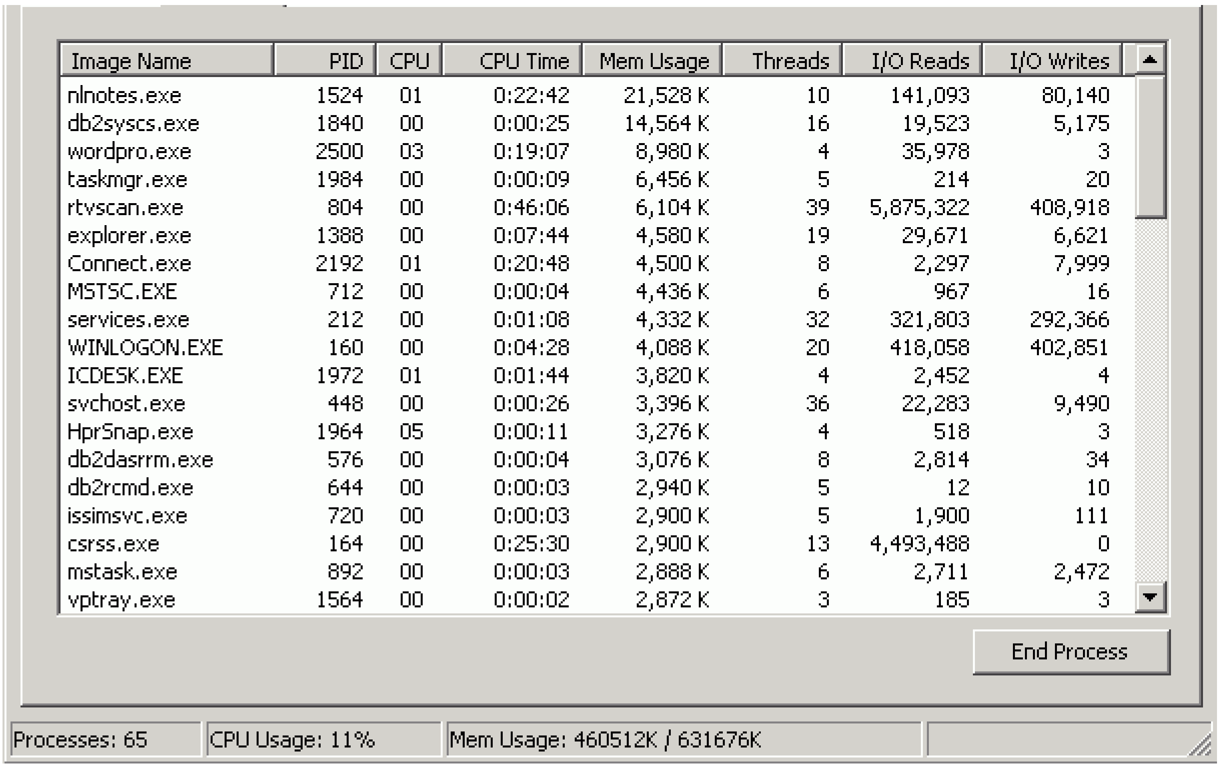


Image Name	PID	CPU	CPU Time	Mem Usage	Threads	I/O Reads	I/O Writes
nlnotes.exe	1524	01	0:22:42	21,528 K	10	141,093	80,140
db2syscs.exe	1840	00	0:00:25	14,564 K	16	19,523	5,175
wordpro.exe	2500	03	0:19:07	8,980 K	4	35,978	3
taskmgr.exe	1984	00	0:00:09	6,456 K	5	214	20
rtvscan.exe	804	00	0:46:06	6,104 K	39	5,875,322	408,918
explorer.exe	1388	00	0:07:44	4,580 K	19	29,671	6,621
Connect.exe	2192	01	0:20:48	4,500 K	8	2,297	7,999
MSTSC.EXE	712	00	0:00:04	4,436 K	6	967	16
services.exe	212	00	0:01:08	4,332 K	32	321,803	292,366
WINLOGON.EXE	160	00	0:04:28	4,088 K	20	418,058	402,851
ICDESK.EXE	1972	01	0:01:44	3,820 K	4	2,452	4
svchost.exe	448	00	0:00:26	3,396 K	36	22,283	9,490
HprSnap.exe	1964	05	0:00:11	3,276 K	4	518	3
db2dasrrm.exe	576	00	0:00:04	3,076 K	8	2,814	34
db2rcmd.exe	644	00	0:00:03	2,940 K	5	12	10
issimsvc.exe	720	00	0:00:03	2,900 K	5	1,900	111
csrss.exe	164	00	0:25:30	2,900 K	13	4,493,488	0
mstask.exe	892	00	0:00:03	2,888 K	6	2,711	2,472
vptray.exe	1564	00	0:00:02	2,872 K	3	185	3

Processes: 65 | CPU Usage: 11% | Mem Usage: 460512K / 631676K

DB2 diagnostics that can be performed quickly

The DB2 snapshot monitor is a tool that assists in quickly narrowing down problems. If the monitor switches are not already enabled by default, turn them all on for your current session:

```
db2 update monitor switches using BUFFERPOOL on LOCK on
      SORT on STATEMENT on TABLE on UOW on SORT on
```

Note: In some large systems with a high temporary table creation rate, enabling the table switch can cause performance degradation. If this occurs, simply turn the table switch back off.

Also, when dealing with DB2 EEE environments, ensure you are monitoring a data partition where the problem is occurring by performing the following:

```
db2 terminate
export DB2NODE=<partition number>
db2 update monitor switches
```

```
db2 get snapshot ...
```

Unless the problem is already narrowed down to a specific application or query, it is best to start with iterations (at least 2, preferably 3) of the more general snapshots: database manager, database, bufferpool, and tablespace. The application, table, and dynamic SQL snapshots have much larger outputs. When working towards defining a problem, it is best to start with the database and database manager outputs to get a feel for the type of activity taking place: how many agents are running in the server, how many connections exist, whether lock issues are occuring, overall bufferpool hit ratios and I/O time, sort activity, etc.

It is often required to script some ongoing database monitoring. Here is an example of such a script, where the iostat and vmstat timings control the frequency of the loop:

```
while [ 1 ]
  do
    datestamp=`date +"%m%d"`.`date +"%H%M"`
    db2 get snapshot for database manager > dbmsnap.$datestamp
    db2 get snapshot for database on  SAMPLE > dbsnap.$datestamp
     ... etc. for each desired type of snapshot.
    ps -elf | sort +5 -rn > pself.$datestamp
    ps aux | sort +5 -rn > psaux.$datestamp
    ipcs -a > ipcs.$datestamp
    uptime >> vmstat.out
    vmstat 1 11 >> vmstat.out
    echo "" >> vmstat.out
    uptime > iostat.$datestamp
   iostat 30 16 >> iostat.$datestamp
done
```

DB2 function call stacks, trap files, and tracing

Other information that can be useful includes DB2 traces and stack tracebacks, both of which show the functions that are executing. Often the function names alone provide some indication of what the process is doing.

For a problem where most DB2 commands are hanging but there is still a lot of activity taking place on the server, take a DB2 trace (with -t timestamp option and keeping the first trace records with the `-i` switch) of a DB2 command, then dump the trace from another session after 30 seconds.

```
Session 1:      db2trc on -i 8000000 -t
                db2 connect to sample
       < wait 30 seconds>
Session 2:      db2trc dmp trace.out
                db2trc off
                db2trc flw trace.out trace.flw
                db2trc fmt trace.out trace.fmt
```

Stack tracebacks can be taken with the **db2_call_stack** command on UNIX, which creates readable trap files t<pid>.<node> in the DIAGPATH, and on Windows with the command

```
db2bddbg -d db2ntDumpTid <any_path> -1 <any_file>
```

The resulting binary file can be formatted with the **db2xprt** utility. This tool is not shipped with DB2, but can be downloaded along with the symbol files required for proper formatting from the appropriate directory of the FixPak download site. See the example on the next panel.

In the case where there is clearly one application or agent that is not responding, you can take trace and call stack diagnostics specific to that process or group of processes. For db2 trace, use the -p option to trace a specific process. Follow the table below to generate DB2 diagnostic trap files on an individual process and to initialize tracing in a particular process or thread. (If a process is in a tight loop, it may not pass through the function that tells it whether or not to start writing trace records).

	Generating a call stack	**Initializing trace**
AIX	kill -PRE <process id>	kill -GRANT <process id>
Other UNIX	kill -URG <process id>	kill -PROF <process id>
Windows	db2bddbg -d db2ntDumpTid <path> <thread> <file>	Not Applicable

Note that the thread of interest on Windows can be obtained from an application snapshot or from Windows performance monitoring where, for example, you can see a particular thread using up excessive cpu.

Formatting stack dump files on Windows

Start DB2 and create a "raw" stack dump file on Windows with this command:

```
db2bddbg -d db2ntDumpTid C:\TEMP -l stack.dmp
```

Now download the appropriate debug.zip file and unzip it into some directory, for example C:\db2debug. If you are running DB2 Version 7.2, FixPak 7, you would download:
ftp://ftp.software.ibm.com/ps/products/db2/fixes/english-us/db2ntv7/FP7_WR21311/Debug.zip

Now format the stack.dump (issue db2xprt with no arguments for syntax help):

```
C:\db2debug\db2xprt /p C:\db2debug stack.dmp stack.fmt
```

Binary stack dump files from any level of DB2 are formatted in this way. You can even format a stack dump file from another machine that is running a different level; you just need to place the debug files in a separate directory and point to that directory with the /p parameter.

You may see "date mismatch" warnings in the output. If these are on the same day but exact hours apart (plus or minus a few seconds), these can be ignored. If they are days or months apart, there is a mismatch in build level between the stack dump file and the symbol files being used for formatting. The date reported from the stack dump file will be correct. Here is an example of a one hour difference that can be ignored:

```
WARNING .date mismatch for DB2ABIND [dbg = 03/14/2002 07:34:11]
[trapfile = 03/14/2002 06:34:12]
WARNING .date mismatch for DB2APP [dbg = 03/14/2002 07:34:10]
[trapfile = 03/14/2002 06:34:12]
WARNING .date mismatch for DB2CLI [dbg = 03/14/2002 07:34:07]
[trapfile = 03/14/2002 06:34:08]
```

Using perfmon on Windows

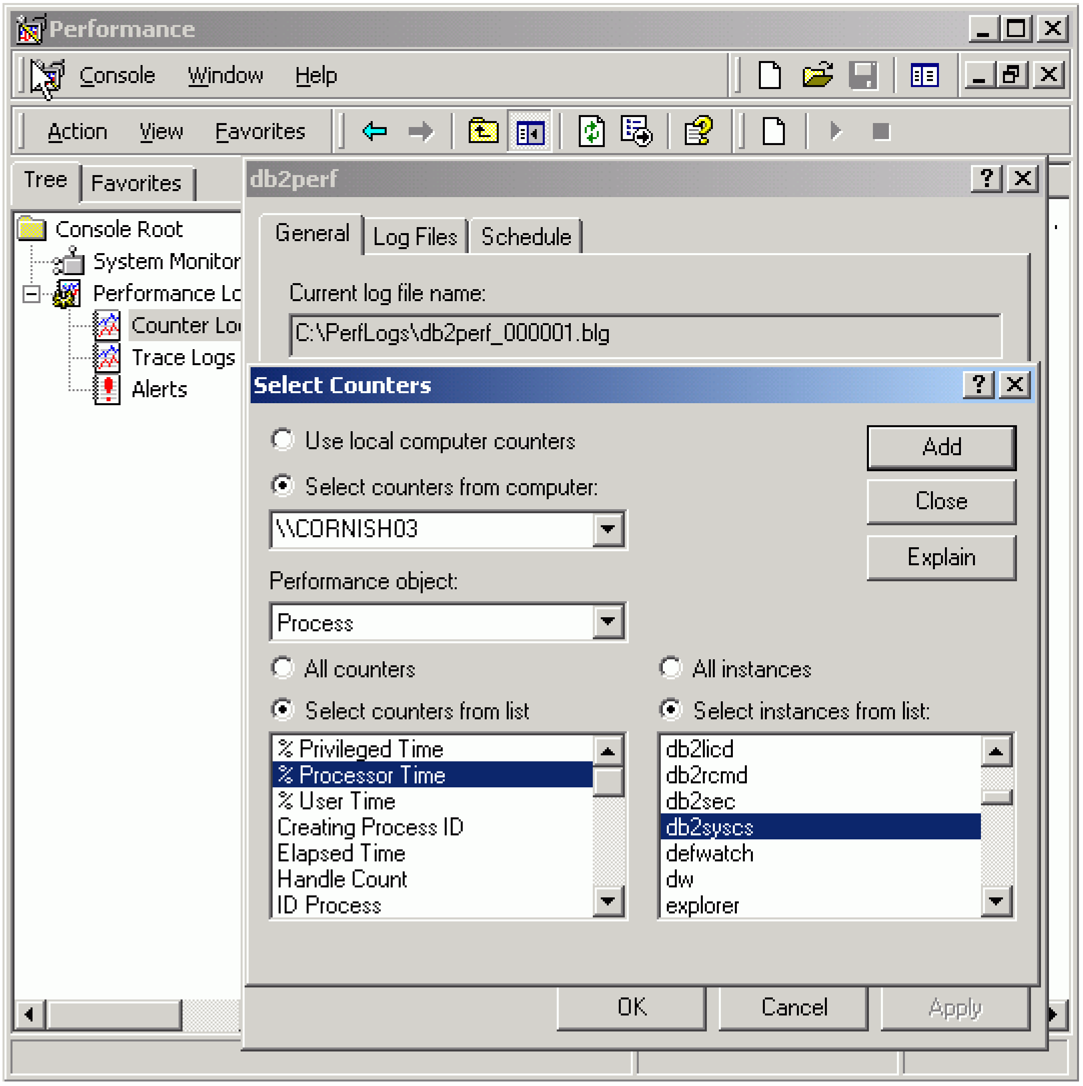
The **perfmon** tool, shipped with Windows, is used to capture performance data and statistics on resource usage. Understanding how to set up and capture a perfmon log is crucial to many types of problem investigation.

The first catch is that, for monitoring I/O, disk counters need to be enabled by running **diskperf -y** (`-ye` for stripe sets), followed by a reboot.

To set up a log on Windows 2000:

1. Run `perfmon` from a command prompt.
2. Select **Performance Logs and Alerts** from the left frame.
3. Right-click on **Counter logs** and select **New Log Settings**.
4. Give the settings a name and click on **Add** to add some performance counters.

Your screen should now look something like this:



Current log file name: C:\PerfLogs\db2perf_000001.blg

Use local computer counters
[✓ Select counters from computer:] \\CORNISH03

Performance object: Process

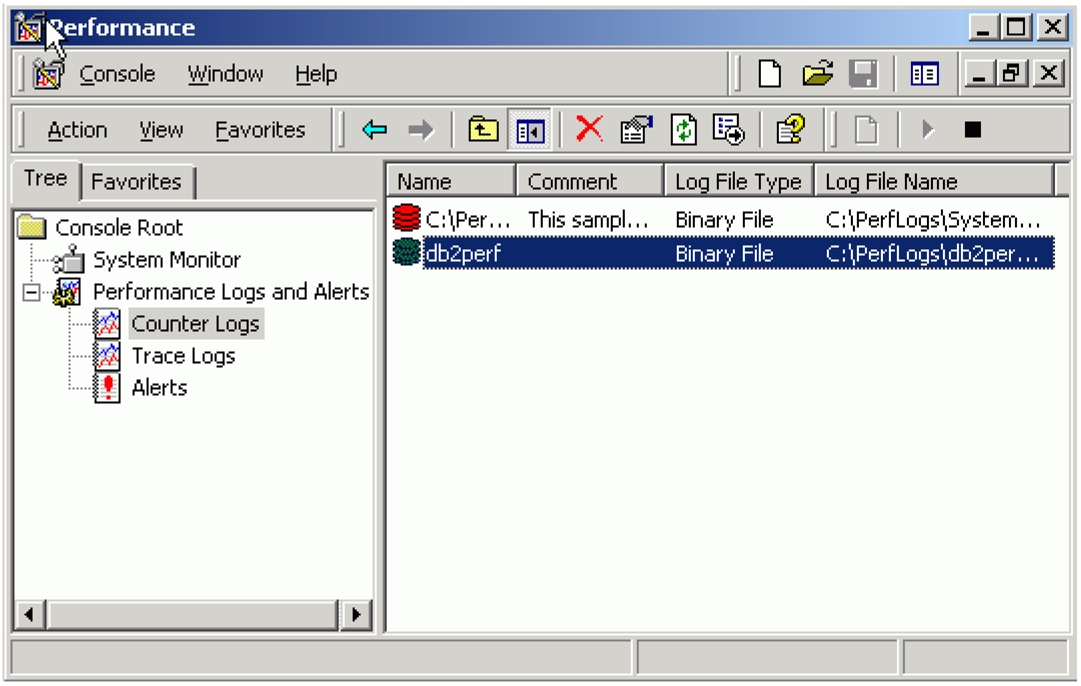
All counters
[✓ Select counters from list]

% Privileged Time
% Processor Time
% User Time
Creating Process ID
Elapsed Time
Handle Count
ID Process

All instances
[✓ Select instances from list:]

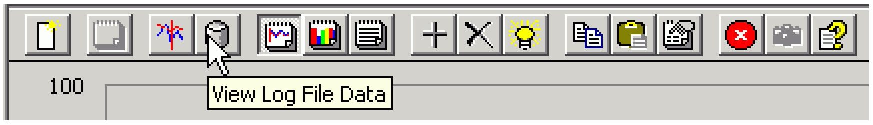
db2licd
db2rcmd
db2sec
db2syscs
defwatch
dw
explorer

Now close the windows by clicking on the appropriate **Add**, **Close**, and **OK** buttons. Once you are back to the initial screen, double-click on **Counter Logs**. You may see that the log is already running, as indicated by the green icon to the left of the log name. If it is not started, right-click on the log and select **Start**. For example, if your log is called db2perf, your screen should now look like this:



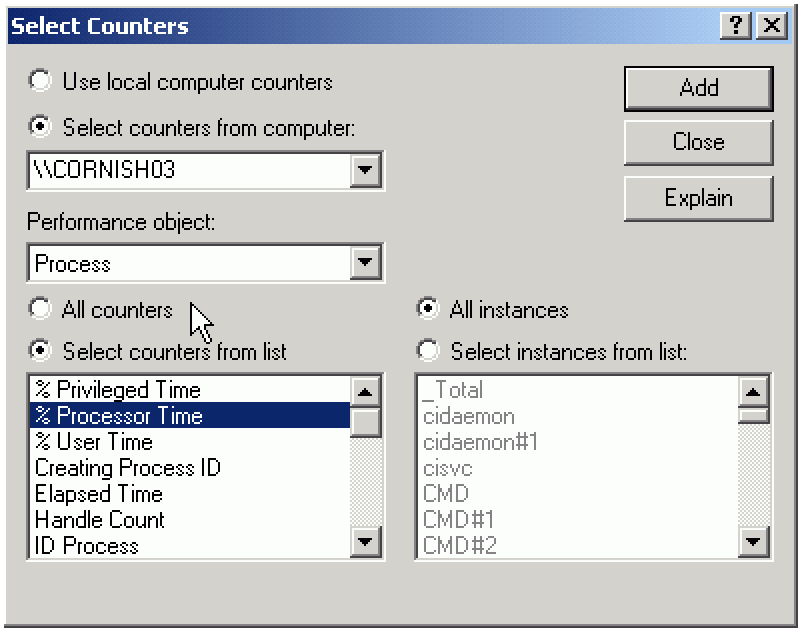
Name	Comment	Log File Type	Log File Name
C:\Per...	This sampl...	Binary File	C:\PerfLogs\System...
db2perf		Binary File	C:\PerfLogs\db2per...

To view the log file, select **System Monitor** from the tree, then select the **View Log File Data** icon:



Once you have selected a log file, you still need to add counters to the chart. Right-click in the chart area, and add counters as desired. Here is a sample setting and output where **All instances** (processes) of the **% Processor Time** counter have been selected.

Setting:



( ) Use local computer counters
(●) Select counters from computer:
\\CORNISH03

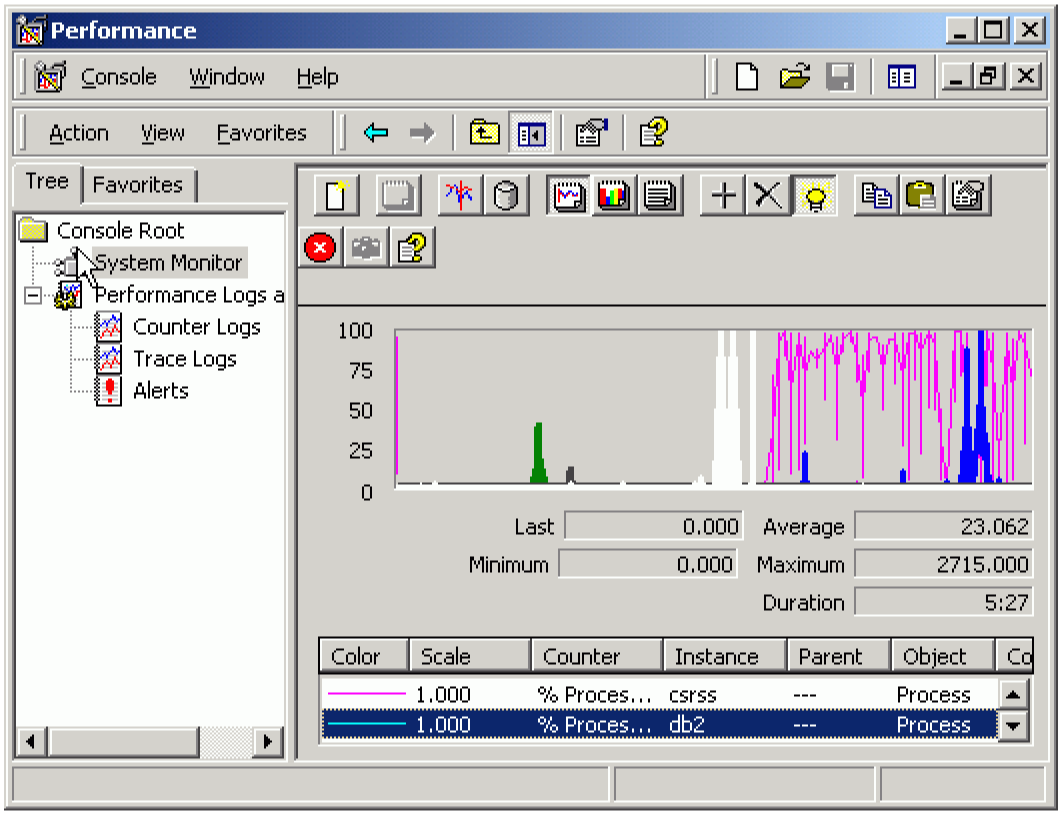
Performance object:
Process

( ) All counters
(●) Select counters from list
(●) All instances
( ) Select instances from list:

% Privileged Time
% Processor Time
% User Time
Creating Process ID
Elapsed Time
Handle Count
ID Process

_Total
cidaemon
cidaemon#1
cisvc
CMD
CMD#1
CMD#2

To make the output easier to digest, select an instance (record) and highlight it in white by pressing Ctrl+H, then cycle through the list and delete the instances that aren't interesting (those that show low CPU utilization):



Last 0.000 Average 23.062
Minimum 0.000 Maximum 2715.000
Duration 5:27

Color	Scale	Counter	Instance	Parent	Object	Co
	1.000	% Proces...	csrss	---	Process	
	1.000	% Proces...	db2	---	Process	

One of the attractive features of perfmon is the ability to combine DB2 and system performance counters together in one log. As you will see in a later example, this makes correlating data a piece of cake!

Avoiding pitfalls and a first glance at system data

Once initial data has been collected, you may be able to find some information that signals a resource issue or some abnormal state with regards to the system, a process, or something DB2-specific. If you are not intimately familiar with the system and the activity taking place, you do need to be on your guard for some of the following pitfalls:

- The data might not have been captured during the time the problem occurred.
- Some of the data might point to an issue that is not of primary interest - in fact, there might be several different problems on the system that contribute to one or more symptoms of varying severities.
- The data might point to a secondary symptom. For example, a new condition or state is triggered from the primary problem that then causes a second symptom, and that symptom

may overtake the first. A good example of this is when there is some resource contention on the database server such that requests get "tied up", and the application servers simply let more requests go to the server on previously idle connections. This might result in a secondary symptom as more and more memory is taken up on the server by active connections.

In all cases it is important to try to link up what the data implies with the problem as initially defined. Often this cannot be achieved immediately, but keeping this goal in sight can prevent you from wasting time trying to solve the wrong problem, or worse, one that doesn't exist.

On UNIX, problem investigation often starts with the following:

- Check vmstat or iostat output for the amount of system and user CPU being consumed versus wait and idle time. Wait time usually, but not always, means waiting on I/O. If system CPU is equal to or greater than user CPU, this almost always indicates a problem.
- Check the top CPU-using processes by ordering ps output by descending CPU penalty (C) and memory usage (RSS and SZ columns). For example:
  ```
      ps -elf | sort +5 -rn
  ```
- Are the same processes near the top? Is the CPU penalty staying in the 60-120 range? This would mean these processes are constantly consuming CPU cycles and need to be investigated.

On Windows, check for the top CPU and memory users by ordering the output on those columns. I/O is more difficult to judge without getting perfmon output.

DB2 snapshot elements associated with system resource usage

Here are the DB2 snapshot elements that map directly to system resource usage.

Database Manager:

- Sort heap allocated - total private memory in use for sorting
- Remote/local connections executing in the database manager - these connections tend to consume the most CPU and memory
- Agents registered - these map to operating system threads under the instance db2syscs process on Windows, DB2 processes on UNIX
- Agents created from empty pool - a high agent creation rate means high system overhead from forking processes on UNIX
- Agents stolen from another application - a high steal rate can result in higher CPU consumption
- Committed private memory - maps loosely to instance private memory consumption on Windows

Database:

- Applications connected currently - indicator of overall load on the system
- Applications executing in the database manager currently - these applications generate the highest need for resources
- Agents associated with applications - indicator of overall load, operating system threads for db2syscs process on Windows, processes on UNIX
- Total sort heap allocated - private memory used for sorts across all database agents
- Bufferpool counters - physical reads and writes, read and write time, time waited for prefetch are all indicators of I/O activity

Application:

- Application status of UOW Executing indicates there are processes or threads doing something on the server, that is running or waiting to run on a CPU, performing or waiting on I/O, etc.
- Bufferpool counters - as previously stated these are good indicators of I/O activity
- Total user or system CPU time used by agents - shows CPU resource usage and can be correlated directly to ps output on UNIX, thread processor time on Windows. Process listings at the bottom of the snapshot helps to map process resource usage from system data to the activity generated by the application

Tablespace:

- Bufferpool counters in a tablespace snapshot help correlate I/O activity for the database from specific tablespaces to I/O statistics at the disk or filesystem level.

Performance problem examples

Too much system CPU

Here is one way to blow CPU through the roof on UNIX:

1. Enable intra_parallelism:
```
db2 update dbm cfg using intra_parallel yes
```
2. Disable agent pooling, increase maximum agents:
```
db2 update dbm cfg using num_poolagents 0 maxagents 500
db2stop force
db2start
```
3. Tell db2 to use 16 subagents per connection:
```
db2 update db cfg for sample using dft_degree 16
```
4. Run the following script:
```
db2 connect to sample
while [ 1 ]
  do
    db2 select \* from project > /dev/null
  done
```

This configuration and this activity results in a very high rate of agent creation by DB2 and process creation on the system. Taking the agent creation rate from a database manager snapshot combined with `vmstat -f` on AIX in a loop we can correlate the data as follows. Note the two numbers are increasing at approximately the same rate:

```
Agents created from empty pool = 11994
   18072379 forks (from vmstat -f)
Agents created from empty pool = 12140
   18072551 forks
Agents created from empty pool = 12375
   18072814 forks
```

`vmstat 1` tells a similar story, with high system calls and high system CPU time:

```
kthr     memory             page              faults          cpu
----- -----------   ------------------------ ------------    -----------
r  b  avm    fre    re  pi  po  fr   sr  cy   in   sy    cs   us sy id wa
5  3  824875 249427 0   0   0   0    0   0   1177 16500 3077 12 70 14  3
6  3  825602 250783 0   0   0   0    0   0   1038 11962 2975 25 72  3  0
6  2  827668 248681 0   0   0   0    0   0   1046 14406 2941  6 91  1  2
```

Too much I/O

Let's use an example on Windows with an underconfigured bufferpool caching a heavily hit table.

1. Disable file caching:

```
db2set DB2NTNOCACHE=YES
db2stop force
db2start
```

2. Increase the size of the employee table until it occupies 1000 pages. Use the following command to accomplish this, and reorgchk to check the table size.

```
insert into <table> select * from <table>
```

3. `db2 alter bufferpool ibmdefaultbp size 50` (Recycle the database so this will take effect).

4. Enable perfmon. Select the System Monitor/chart view, and select the following performance monitor counters by right-clicking in the chart and selecting **Add Counters**:

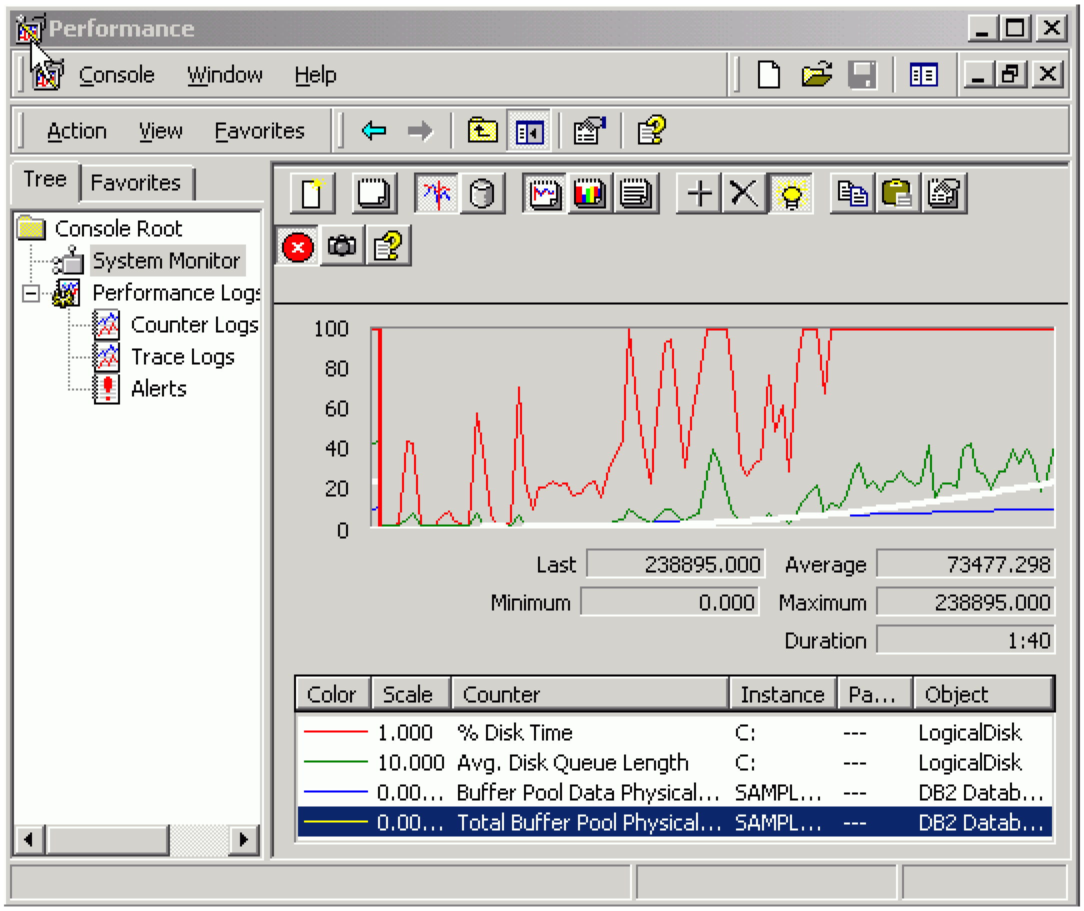
Object	Counter
Logical Disk	% Disk Time (choose the disks containing employee table)
Logical Disk	Avg Disk Queue Length
DB2 Databases	Bufferpool Data Physical Reads
DB2 Databases	Total Bufferpool Physical Read Time

5. Now select all records from the employee table in a loop using the following batch file:

```
db2 connect to sample
:begin
db2 select * from employee
goto begin
```

6. Issue this script from new CLP sessions until you reach 100% busy time on the disk where the employee table resides.

After adjusting the scaling on some of the variables, your chart output should look something like this:



Last 238895.000 Average 73477.298
Minimum 0.000 Maximum 238895.000
Duration 1:40

Color	Scale	Counter	Instance	Pa...	Object
🟥	1.000	% Disk Time	C:	---	LogicalDisk
🟩	10.000	Avg. Disk Queue Length	C:	---	LogicalDisk
🟦	0.00...	Buffer Pool Data Physical...	SAMPL...	---	DB2 Datab...
🟨	0.00...	Total Buffer Pool Physical...	SAMPL...	---	DB2 Datab...

Too much memory

This example uses large sort heaps to demonstrate increased db2agent private memory usage. The monitor data is taken from Solaris, but the symptoms can be observed on any platform.

1. Increase the employee table to about 200 pages using the following statement. Check the table size with the `reorgchk` command.

```
insert into employee select from employee&nbsp;
```

2. Increase sheapthres (database manager configuration) to 500000 and enable the sort monitor switch. Increase sortheap to 20000. Recycle the instance.&nbsp;

3.
4. Place the following statement in a script:

```
select e1.empno, e2.firstnme, e3.midinit, e4.lastname,
e5.workdept, e6.phoneno,e7.hiredate, e8.job from employee e1,
employee e2, employee e3, employee e4, employee e5,
employee e6, employee e7, employee e8 where e1.empno=e2.empno
AND e2.firstnme=e3.firstnme AND e3.midinit=e4.midinit AND
e4.lastname=e5.lastname AND e5.workdept=e6.workdept AND
e6.phoneno=e7.phoneno AND e7.hiredate=e8.hiredate AND
e8.job=e1.job order by 7,5,3,1,2,4,6,8
```

5. Issue this statement from multiple sessions, checking the following commands after each:
   - `vmstat 1`
   - `/usr/bin/ps -elf | sort +9 -rn | head`
   - `swap -s`
   - `db2 get snapshot for database manager | grep "Sort heap allocated"`

You should notice several things:

- `get snapshot for database manager` shows the sort heap allocated increasing by ~20000 pages for each new session
- The SIZE (in 4K memory pages) of the large agents in the ps output is made up mostly of the private memory they occupied for the sorts. (ps output on Solaris includes the shared memory attached to by the agents, but the shared memory in this configuration is small)
- `vmstat 1` may show some paging and a continually decreasing free list. While paging is not good, a low free list is quite normal and does not indicate a problem istelf. For example, 3% of RAM is a common low threshold, but hitting around 0 consistently is a problem.

```
procs     memory            page            disk          faults      cpu
r b w  swap  free  re  mf    pi  po fr de sr s0 s1 s2 s6  in   sy   cs us sy id
0 0 0 923320 26688  0  160   0   0 0  0  0  0  0  0  0  610   23  200 28  2 70
3 1 0 922904 22864 228 632  2720 0 0  0  0  6  0  0  0 3421 1312 1516 26 27 48
1 0 0 922576 16856 308 1284 3360 0 0  0  0  9  0  0  0 4442 2310 2137 67 33  0
```

Finally, cancel each session that is running the sorts. Take a look at `usr/bin/ps -elf | sort +9 -rn | head` again. Has the SIZE column decreased since the sessions have been terminated? It is quite normal that the agents are still the same size. Processes retain their high water mark in memory usage on most UNIX platforms.

Summary

Where to go from here

Of course this is only an introduction to get you started.

You can solve some problems or at least get on the right track just by taking an initial look at the right data. Other problems are more complicated or subtle and require a strategic approach that uses several data collection and more complex analysis techniques.

Here are a few additional resources available:

- DB2 online documentation: http://www-3.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/document.d2w/report?fn=db2v7d0frm3toc.htm
- UNIX man pages contain a wealth of information about tools
- AIX documentation: http://publibn.boulder.ibm.com/cgi-bin/ds_form?lang=en_US&viewset=AIX
- Solaris documentation: http://docs.sun.com/
- HP-UX documentation: http://docs.hp.com/
- Linux documentation: http://docs.linux.com/
- Windows Support: http://www.microsoft.com/technet/support