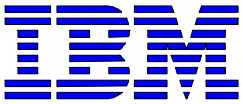
Performance Tuning Notes for DB2 Client/Server



Liwen Yeow
IBM Toronto Lab
DB2 Client/Server Technical Sales Service
1150 Eglinton Ave. East
Toronto, Ontario
M3C 1H7
Internal Address: 1B/688/1150/TOR
E-mail: yeow@vnet.ibm.com

Nov 18th, 1996

Although the author acknowledges the contribution of the Toronto DB2 performance and service teams, the contents of this document are the personal notes of the author and are **not** those of IBM. The author does not guarantee accuracy of the information herein and may make changes at any time.

After reviewing this document, if you would like to suggest changes, additions or corrections, please send a note to the author at the address on the cover page. Your comments on this document, whether positive or negative, are greatly appreciated.

DISCLAIMER

THE INFORMATION CONTAINED IN THIS DOCUMENT HAS NOT BEEN SUBMITTED TO ANY FORMAL IBM TEST AND IS DISTRIBUTED ON AN "AS IS" BASIS WITHOUT ANY WARRANTY EITHER EXPRESSED OR IMPLIED. The use of this information or the implementation of any of these techniques 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. Customers attempting to adapt these techniques to their own environments do so at their own risk.

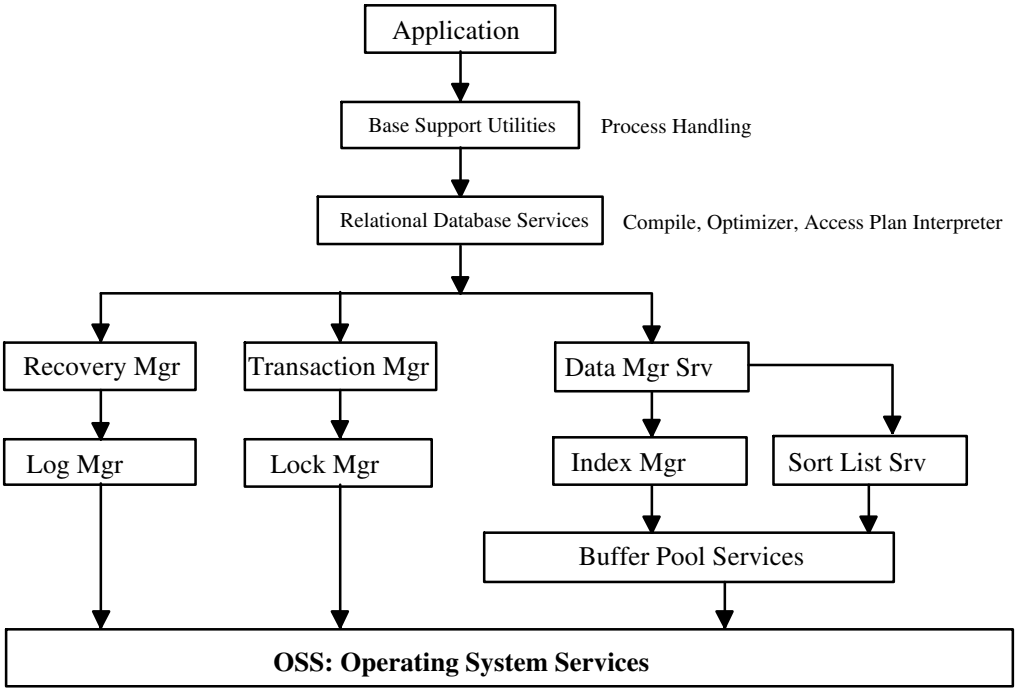
Permission to use, copy, and distribute the document for any purpose without fee is hereby granted, but is subject to the following conditions: 1) The document may only be distributed in countries recognizing the copyright rights of U.S. nationals. 2) All text, including this notice and all copyright notices, must be copied without modification and all pages must be included.

General Notes on DB2 Performance ..... Page 1
Component Overview ..... Page 1
Database Design ..... Page 1
Application Design ..... Page 2
Optimizer ..... Page 3
System Considerations ..... Page 3
Client/Server Database ..... Page 4
System Tuning ..... Page 4
Notes on DB2 Version 1 ..... Page 5
System Tuning ..... Page 5
Client/Server Tuning ..... Page 7
System considerations ..... Page 8
DDCS Considerations ..... Page 9
DDCS Gateway ..... Page 9
VTAM/NCP Definitions ..... Page 9
DDCS Authentication ..... Page 11
Scenario for Local Client• ..... Page 11
Scenario for Remote Client ..... Page 11
AIX and SNA/6000 Considerations ..... Page 11
MVS Considerations ..... Page 11
Version 2 Notes ..... Page 12
Files ..... Page 12
Tables ..... Page 12
LOBS ..... Page 14
Triggers ..... Page 14
UDFs ..... Page 14
UDTs ..... Page 14
Load ..... Page 15
Recovery ..... Page 15
Event Monitor ..... Page 15
Miscellaneous Performance Considerations ..... Page 16
Distributed Unit of Work (DUOW) and 2-Phase Commit ..... Page 18
CAE/Win Performance Parameters ..... Page 18
DB2 for NT ..... Page 20
Memory Requirements ..... Page 20
Performance Considerations ..... Page 21
General Notes ..... Page 21
DB2 for Solaris and HP-UX ..... Page 23

Kernel Configurations ..... Page 23
Nodelock file ..... Page 23
Messaging ..... Page 23
Semaphores ..... Page 23
Shared Memory ..... Page 24
General Notes on Solaris ..... Page 24
Corrections to the Version 2.1.0 DB2 for Solaris README File ..... Page 24
Appendix: UNIX Commands ..... Page 25
Appendix: Notes on RAID ..... Page 26
RAID Levels ..... Page 26
RAID 0 ..... Page 26
RAID 1 ..... Page 26
RAID 3 ..... Page 26
RAID 5 ..... Page 26
RAID Block Size ..... Page 27
Other important things about RAIDs ..... Page 27
Appendix: Differences in Predicate Evaluation After Migration ..... Page 28
Version 2.1.x ..... Page 28
Version 1.2 ..... Page 29
Appendix: Sort Heap memory changes in v2.1.2 ..... Page 29
Appendix: AIX Example of a UDF ..... Page 31

General Notes on DB2 Performance

Component Overview



Database Design

- Normalized database design
  - avoids data redundancy for simple maintenance
  - more tables contribute to more input/output
  - more joins contribute to complex SQL
  - preferred for transaction-oriented applications
- Denormalized database design
  - produces some data redundancy and is harder to update
  - fewer tables contribute to less input/output
  - fewer joins contribute to simpler SQL
  - preferred for query applications
- Additional summary data tables
  - more tables require more disk space
  - used to offload from production, especially for decision support
  - duplicated data makes data less consistent
- Tables
  - Small tables may have contention problems. Large tables rarely have such problems.
  - Large tables may impact utilities such as IMPORT. Online applications may perform better on large tables because page contention is less likely.
  - Splitting tables reduces scan cost
  - Horizontally (by key ranges or by current/archived)
  - Vertically (by sets of columns). Note that you must duplicate matching key columns, and must consider the overhead for joining tables.
- Row length and positioning of columns

- row length vs. page size: longest row is 4KB, maximum of 127 rows per page
- rows less then 24 bytes waste space
- rows that are not divisors of 4 KB waste space. For example, use rows that are 2 KB so that 2 of them use the entire 4 KB space.
- minimize the number of pages
- minimize variable length columns:
  - They require 2 extra bytes.
  - Columns following variable length columns do not have fixed positions in rows. Therefore, CPU cycles are required to calculate positions.
  - Do not use LONG VARCHARS unless you intend to use maximum row length, since they increase CPU consumption and logging.
- for read-only tables, place variable length columns at the end of the row. For updated tables, place frequently updated columns, whether fixed or variable, at the end of the row.
- do not define columns that are less than 18 bytes as VARCHAR.
- DECIMAL columns consume more CPU resources.

- NULL values
  - are useful if numeric data with missing values is computed
  - require more complex programming
  - require additional disk storage. Consider NOT NULL WITH DEFAULT, since NULL requires an extra byte
  - NULL values are treated as unequal; NULL is presented only once when selecting DISTINCT values; if counting DISTINCT, NULL values are not counted; NULL values are ordered last.
- Index selection
  - is required to enforce unique key values and support primary keys
  - is desirable for:
    - WHERE predicate evaluation
    - ORDER BY and GROUP BY
    - index-only access
    - foreign keys
    - support JOINS
  - balance retrieval benefits vs. the update cost of indexes
  - indexes with low cardinality hurt index maintenance performance
  - the DB2 optimizer uses index statistics gathered by RUNSTATS
  - additional indexes increase the possibility of deadlocks because they provide alternative update paths to the same data page

Application Design

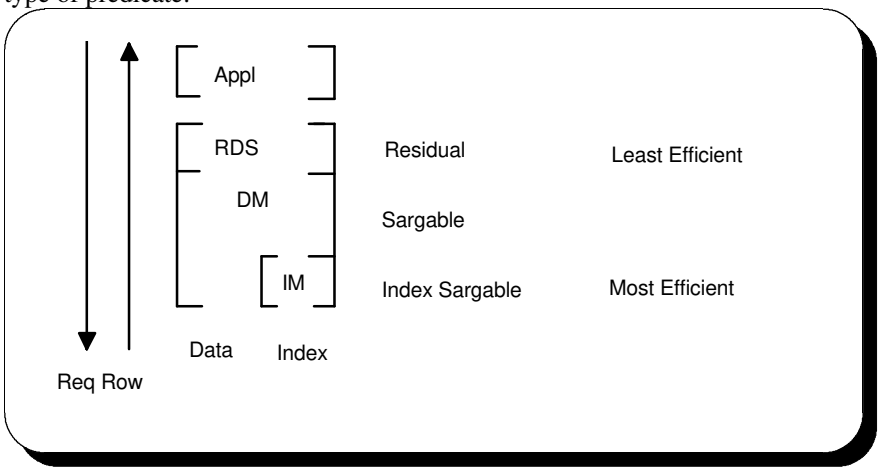
- Lock waits and deadlocks problems
  - lock escalation hurts concurrency level because whoever gets escalated prevents others from accessing or updating data
  - avoid inappropriate isolation levels
  - update in a standardized order (make sure programs don't update in a different order)
  - avoid a large number of table indexes
  - avoid inappropriate use of TABLE LOCK
  - examine the lock types chosen by the optimizer via EXPLAIN
  - review the lock list limit
  - perform insert, update, and delete at the end of a unit of work (UOW)
  - consider forcing update locks on data
    - for cursor updates, use FOR UPDATE OF on cursors
    - for cursor deletes, use FOR UPDATE OF on dummy non-index columns
  - include retry logic in applications
  - use FETCH to hold row locks
  - use SELECT to read data and free rows
- Transaction management
  - ensure that UOW is clearly defined

- avoid excessive or insufficient COMMITS: frequent commits enhance concurrency and shorten recovery time, but require more CPU resource
- holding a lock while waiting for a user to respond can result in excessive lock time
- unneccessary CONNECT statements are expensive.

- Data handling
  - every column and row touched increases CPU and elapsed time
  - use selective predicates to reduce the number of rows searched
  - reduce the number of columns retrieved and updated. Use SELECT * only if required
  - use REORG often to improve clustering

Optimizer

- the DB2 optimizer considers the following in choosing a join method (nested loop, merge), order (inner vs. outer tables), and the access path:
  - buffer pool size (rebind if this size changes)
  - indexes - primary and secondary key column statistics
  - restrictiveness of predicates
  - table size
  - data ordering (GROUP BY, ORDER BY, DISTINCT)
  - type of predicate:



- use EXPLAIN to view optimizer decisions
  - watch for join method and access path
  - use JOINS instead of sub-SELECTS where possible
- when you write SQL, keep in mind that the optimizer works best if you avoid NOT and use:
  - EXIST instead of IN
  - BETWEEN instead of LIKE
  - JOINs instead of subqueries

System Considerations

- DASD
  - executable code should be placed in a separate partition
  - files that are subject to reinitialization (for example, the swapper and database logs) should be placed in partitions of their own
  - runtime logs should be placed in a separate partition
  - frequently updated and modified data should be in a separate partition
  - use adjacent partitions to reduce head movement
- File Systems
  - place log files on FAT partitions for fast serial access

- place database DAT and INX files on HPFS partitions, which are good for large disk partitions, large files, and a large number of subdirectories.
- keep the HPFS cache size to a minimum (DB2 for OS/2 does its own caching.)
- put the SQLLIB directories at the beginning of the PATH and LIBPATH statements

Client/Server Database

- Cursor record blocking
  - if possible, avoid the WHERE CURRENT OF <cursor> clause in UPDATE and DELETE statements, because it prevents the target cursor from being blocked. Use a cursor and separate update and delete statements instead.
  - avoid ambiguous cursors. For example, an EXECUTE IMMEDIATE statement that issues an UPDATE against a static cursor. (The cursor is ambiguous because the bind process cannot predict the type of statement in EXECUTE IMMEDIATE.)
  - use the FOR FETCH ONLY or FOR UPDATE clauses on cursors
  - use blocking all or the /K=ALL bind parameter
  - use DARI because it can reduce network messages. It benefits applications that:
    - process large amounts of data between a remote procedure and a database, but return only a small answer set.
    - contain logical sets of SQL that are always executed together.

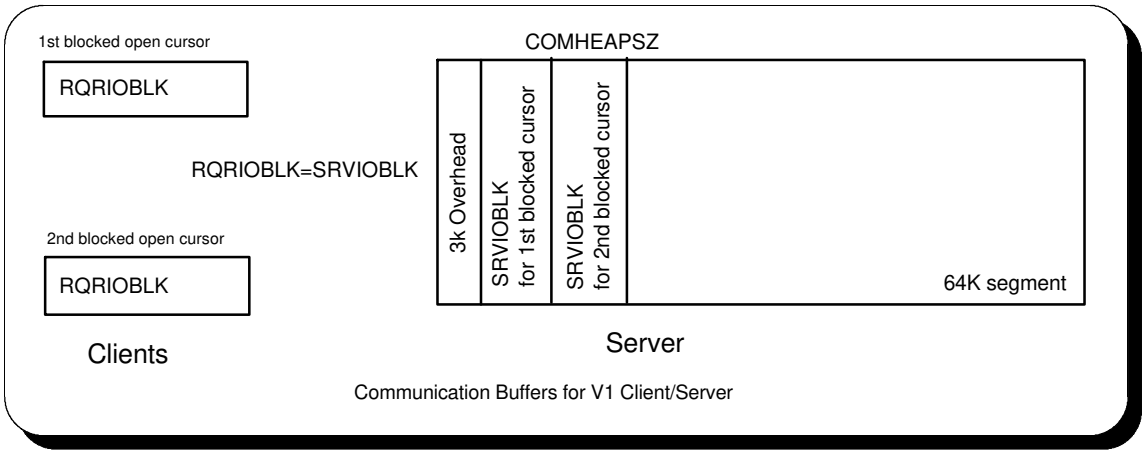
System Tuning

- DB2 uses the following tuning parameters:
  - Database manager configuration parameters determine the amount of system resources that affect all databases and applications using DB2
  - Database configuration parameters apply to a specific database, and specify the amount of resource allocated to that database.

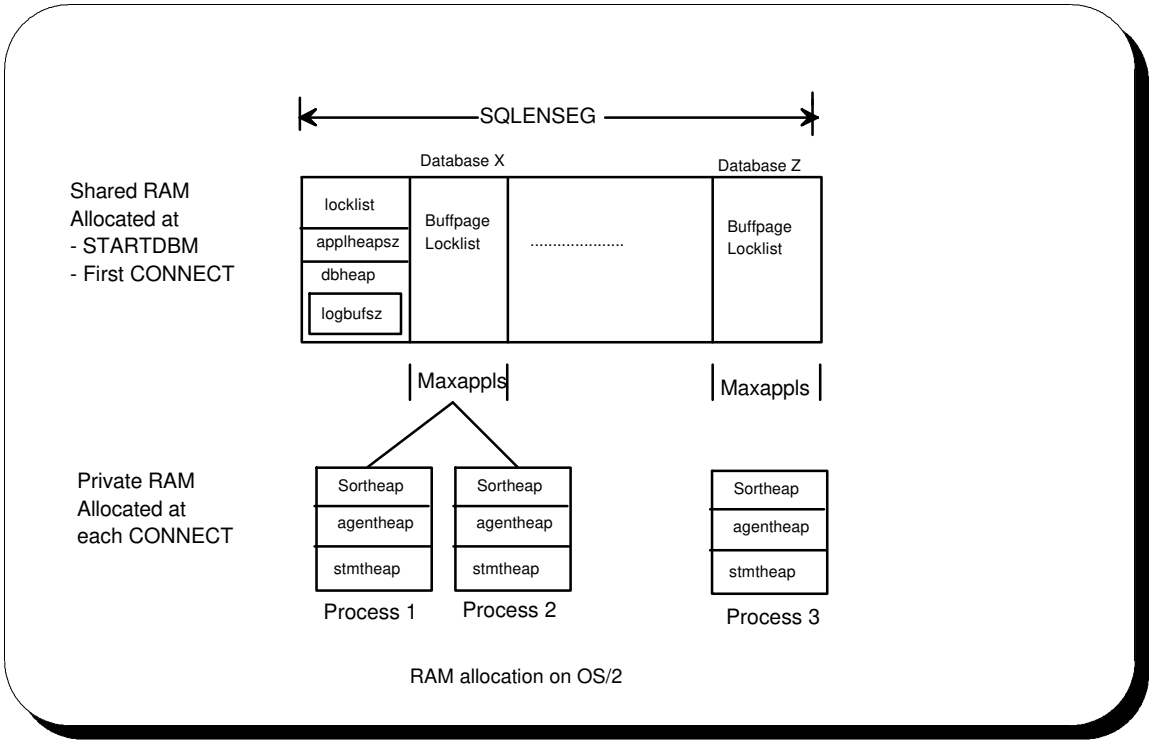
Notes on DB2 Version 1

System Tuning

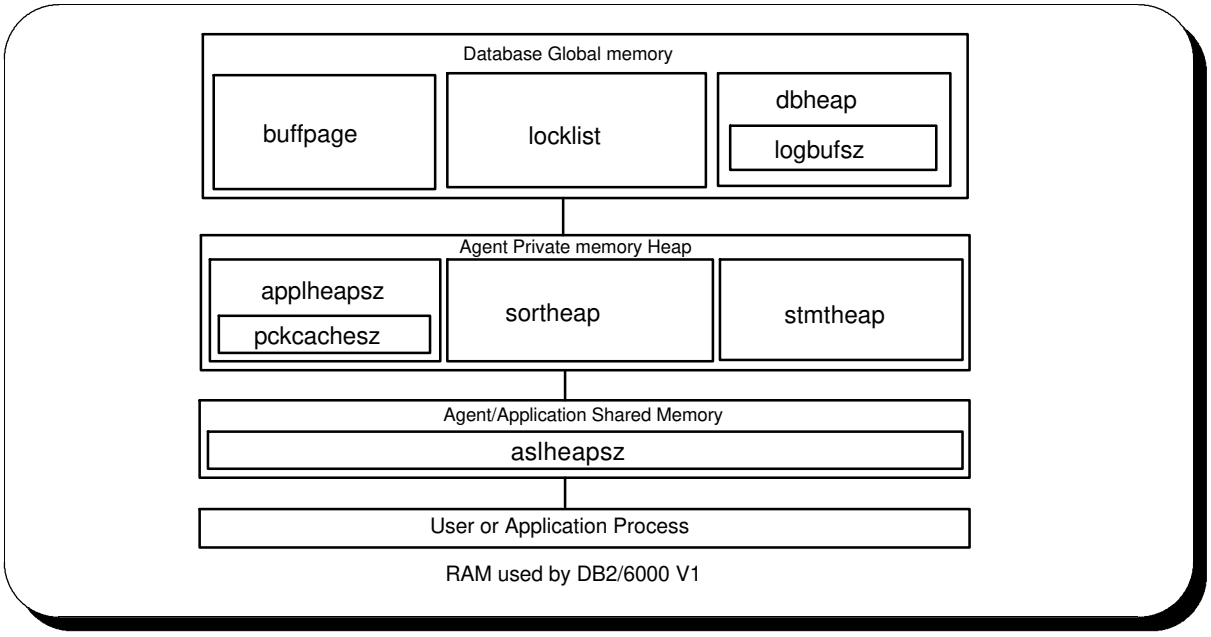
- DOS/Windows client parameters are stored in DBDRQLIB.CFG and affect the database manager environment. Examples include the NETBIOS name and the size of the communication area.
- Be sure to determine if RAM is shared (is available for all processes connected to database), or private (is available for each process)



Communication Buffers for V1 Client/Server



RAM allocation on OS/2

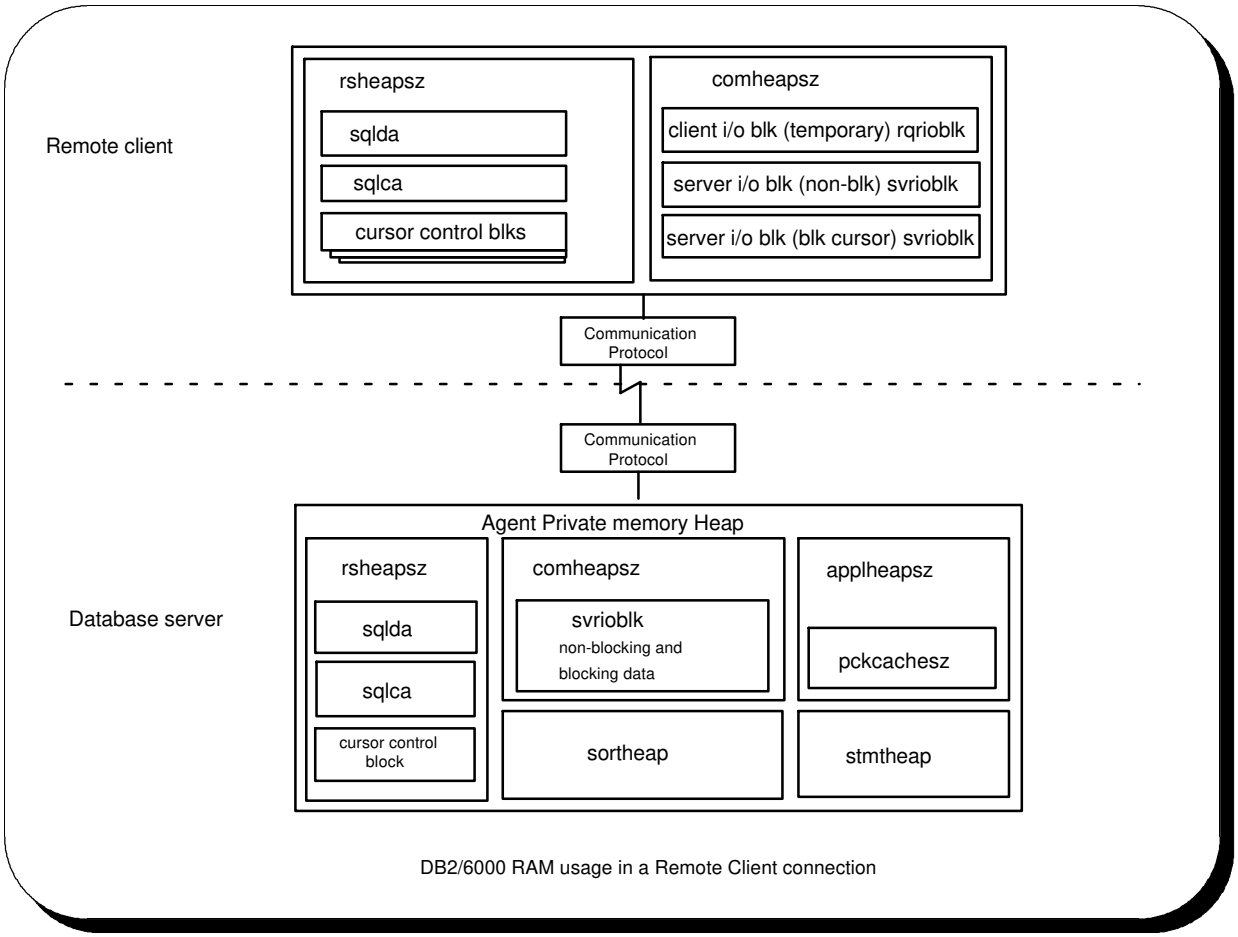


- SQLNETB (in the CONFIG.SYS file) defines the number of sessions that DB2 for OS/2 will request from NetBIOS at STARTDBM time. It represents the number of concurrent sessions, plus DB2 for OS/2 overhead:
  SQLNETB = 6 + inbound sessions + outbound sessions
- BUFFPAGE is the buffer pool size for a database (in number of 4 KB pages)
  - data read from or written to a disk is transferred to/from this buffer
  - data stays in the buffer for the next request, requiring less input/output
  - affects the access plan
  - if the buffer pool is too big, some data may be paged out to the swapper
  - when setting this value consider how much space (in 4 KB pages) is occupied by those tables and indexes that are most frequently accessed
  - buffer pool pages are written back to disk when one of the following occurs:
    - all processes disconnect from the database
    - the database quiesces (that is, all connected processes have been committed)
    - room is needed for another incoming page
    - Warning: If you notice that commits take a very long time to complete and disk activity is high, the database manager may be flushing the buffer pool. The time required to do this is dependent on the size of the BUFFPAGE parameter. That is, the larger BUFFPAGE is, the longer the time required by your system to flush the buffer pool. If your system appears to "hang", it may be a combination of the log file being full (frequency of flushing), a large buffer pool size, and frequent commits.
- SORTHEAP is the sort heap size (in 4 KB pages) for each process connected to a database
  - it is allocated as needed and freed when sorting completes
  - it is used during index creation
  - its value should not exceed the size specified in the SHEAPTHRES database manager configuration parameter
  - recommendation: `2.5 x (max table to be sorted) / 64`
  - if SHEAPTHRES is exceeded, any subsequent sorts will be performed in temporary files, thereby reducing the maximum sort heap size.
- DBHEAP is the heap size (in 4 KB pages) for a database
  - the database heap size mostly contains cursor control block information. Its size is dependent on the number of simultaneous open cursors. Increase its value if an application receives the SQL0956C message.
- APPLHEAPSZ is the application heap size (in 4 KB pages) for each process using a database. It is the working area for a process.
  - Increase its value if an application receives the SQL0954C message

- MAXAPPLS is the maximum number of concurrent processes that can be connected (locally and remotely) to a database
- LOCKLIST is the maximum storage for lock lists for a database
  - it sets the number of 4 KB pages for keeping locks
  - each lock requires 25 bytes of RAM
  - reaching this limit causes lock escalation and deadlock. (Lock escalation is the process of replacing row locks with table locks, thereby reducing the number of elements in the list.)
  - control size with frequent COMMITs and cursor stability (rather than repeatable read)
- MAXLOCKS is the maximum percentage of the lock list to be filled before escalation
  - it should be greater than or equal to `(200 / Maxappls) + 1`
- DLCHKTIME is the time interval in milliseconds to check for deadlock conditions. When it is reached, database services decides which process to kill and subsequently roll back
- SQLENSEG is the memory shared segment for the database manager
  - it is allocated as needed
  - set it at the maximum of 8192

Client/Server Tuning

- NUMRC limits the number of concurrent connections
  - for a DDCS for OS/2 gateway, `NUMRC = A + 2B + C`
    where A is the maximum number of local sessions to DRDA AS, B is the maximum number of client sessions to DRDA AS, and C is the maximum number of client sessions to the database located on the DDCS for OS/2 machine
- COMHEAPSZ is allocated on servers and clients for each client process that connects to a server database
  - add one to the value of COMHEAPSZ on a DDCS for OS/2 gateway for each client request through it.
  - COMHEAPSZ is greater or equal to
    `(SRVIOBLK x m + 65535) / 64KB`
    where m is the number of concurrently open cursors
  - if the communication heap is full, no record blocking is done. This leads to row processing and subsequently poor performance at STARTDBM. A region of contiguous memory the size `COMMHEAPSZ x NUMRC` is allocated (but not committed).
- SRVIOBLK/RQRIOBLK are the number of I/O blocks (in bytes) for communication between the server and requester. One block is allocated for each open blocked cursor.
  - the size allocated between the client and server is the size of the SRVIOBLK at the server. It is recommended that you make the RQIOBLK size on the client equal to the SRVIOBLK size on the server.
  - for DDCS, COMHEAPSZ must be increased by one for DDCS to HOST input/output connections
  - increasing the size of input/output blocks yields better performance if the number or size of rows being transferred is large.
  - in DOS or Windows, the SQLSIZE parameter specifies the size in bytes of the communications block and work area.



DB2/6000 RAM usage in a Remote Client connection

System considerations

- Memory usage for Version 1 of DB2 for OS/2:

Database Services (standalone)	1.4 MB
Database Requester	0.5
Database Server	2.6
For each additional concurrent connection	0.2
Query Manager	1.6

  Note: Database Server includes a 1-MB buffer pool

- CONFIG.SYS parameters:
  - The DISKCACHE and IFS defaults are DISKCACHE=64 and IFS=C:\OS2\HPFS.IFS /C:64
  - DEVICE provides support for most hardware
  - DEVICE=VDISK (virtual disk is not as important in OS/2 as in the DOS environment)
  - THREADS is the maximum number of concurrent execution threads. Memory overhead is required for each thread. The default of 512 can be reduced
  - PROTECTONLY controls DOS compatibility mode support. Turning it off (NO) can save 0.5 MB
  - MEMMAN is the storage allocation option for memory manager
  - put the SQLLIB directories at the beginning of the PATH and LIBPATH statements

DDCS Considerations

DDCS Gateway

- Ensure that the Receive Window Count (Data Link Control profile in the SNA definition) is set to equal to or less than the MAXOUT parameter on VTAM (see the VTAM definitions below for details). This parameter sets the number of path information units (PIUs) that workstations will receive before sending a response to the host. If the Receive Window Count (RCW) is greater than MAXOUT (for example, RWC=10 and MAXOUT=7), the host will send 7 PIUs and wait for a response from the receiving PIU before sending more. Since RWC is set to 10, the workstation will not respond until it receives 10 PIUs or a timeout occurs. This response is at a lower level then the PACING set at the LU level. LU pacing is determined at the mode entry level (for example, IBMRDB), and is known as session-level pacing.
- The PACING value on the workstation should be set at one less than that on the VTAM PU/LU and APPL PACING/VPACING value. Suggested values are PACING=8 on the workstation and PACING/VPACING on VTAM. Ideally the PACING value should be set at: RQRIOBLK divided by the RU size.
- The maximum amount of data sent in a block is limited to the value set in the database manager configuration parameter RQRIOBLK. That is, if RQRIOBLK is set to 32 KB and RU size is set to 4 KB, the LU to LU pacing will not be higher than 8 even if you have a higher value defined in the mode entry (for example, IBMRDB).
- SNA SERVER/6000 is limited to an RU size of 32 KB.
- Early Token Release should be enabled. This permits more than one token to be active on the ring. Caution: for this option to be enabled, all machines on the same LAN segment MUST be configured with this option enabled.
- Best general throughput to and from the host has been to set the RU size at 4 KB and RQRIOBLK at 32 KB.

VTAM/NCP Definitions

- The IOBUF parameter in the VTAMLST member ATCSTRAn determines the buffers used by VTAM. The parameters are

  IOBUF=(*baseno,bufsize,slowpt,F,xpanno,xpanpt,*)

  where:
  - BASENO should be the number of buffers in a steady state. Use the VTAM command D NET,BFRUSE
  - BUFSIZE is the size in bytes of each buffer. Buffers are allocated in 4 KB pages. Choose a buffer size that does not waste space. The formula is:

    `4096/(BUFSIZE+71)`

    Good values are 953 for 4 buffers per page and 441 for 8 buffers per page
  - XPANPT should be greater then SLOWPT
- Token Ring PUs are defined in Switched Major Nodes. The MAXOUT parameter here should correspond with the Receive Window Count in the workstation DLC definition
- In the NCP definition, the Token Ring Interconnect is defined in a LINE statement. Set MAXTSL=2044. This parameter defines the maximum PIU that NTRI can transmit. 2044 is the maximum value. MAXTSL should be greater or equal to RECVBUF.
- Code DELAY=0 in the BUILD, HOST parameter in NCP and PCCU and GROUP LNCTL=CA
- Include the following in the mode entry code: PSNDPAC=X'00',SSNDPAC=X'02',SRCVPAC=X'00'
- In the DB2 APPL *majnode*, make sure AUTOSES=1
- The following is the calculation for the maximum PIU size that can flow to
  - **VTAM**
    (MAXBFRU * IOBUFsize) - BFRPAD
  - **NCP**
    the **lesser** of MAXDATA or (TRANSFR * BFRS) -18
- MAXBFRU and BFRPAD can be found in the NCP-HOST definition. IOBUFsize can be found in the IOBUF parameter. MAXDATA can be found in NCP-PCCU. TRANSFR and BFRS can be found in NCP-BUILD.
- The UNITSZ parameter found in the NCP-HOST statement should be equal to IOBUFsize

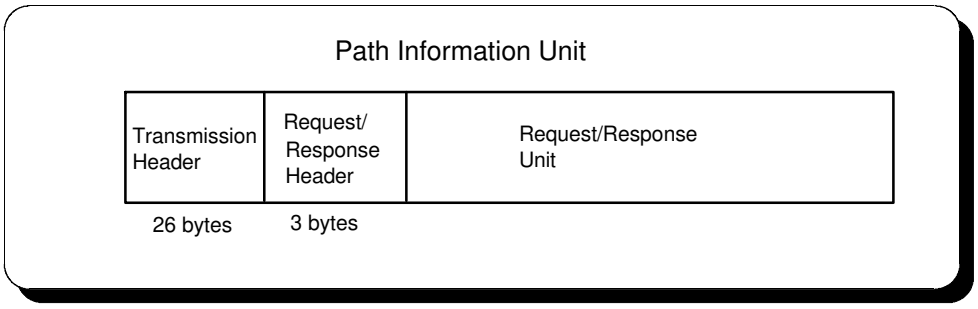
- The RUSIZE parameter in the MODEENT statement for the mode table defines the RU size at the LU level.
  RUSIZE=*xxyy* (for example, xxyy=8989)
  where *xx* is the PIU size from the SLU to PLU and *yy* is the PIU size from the PLU to SLU

  The formula is: *Mantissa * (2 raised to the 'n' power)*
  For instance, in the above example: 8 * (2^9) = 4096 bytes

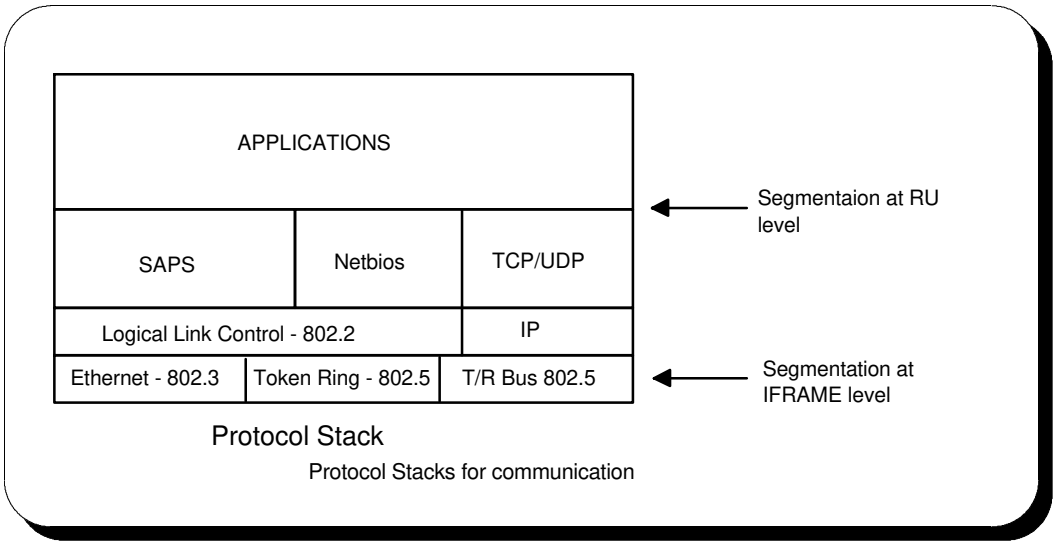
- Summary of parameters that affect PIU size:

KEYWORD	***STATEMENT***	***DESCRIPTION***
MAXDATA	PCCU	Maximum outbound PIU size from Host
TRANSFR	BUILD/LINE	Maximum number of NCP buffers that may be used to contain a single PIU
BFRS	BUILD	NCP buffer size
MAXBFRU	HOST	Maximum number of host buffers that can be used to contain a single PIU
UNITSZ	HOST	Size of host buffers
BFRPAD	HOST	Number of pad characters required by the host access method

If the PIU is too large at any node along the way, the error code '*800A*' is transmitted.



Communication between a PU Type 2.1 and Host uses the FID 2 transmission header format.



.

DDCS Authentication

Scenario for Local Client

Authentication Specified	Client	Server	Server	DCS
Security Specified	Same	Same	Program	Program
Authentication Performed	DDCS server	DDCS Server	DDCS Server and Host	Host

Scenario for Remote Client

Client Authentication Specified	Client	Server or DCS	Server or DCS	Server or DCS
DDCS Authentication Specified	Client	Server	Server	DCS
DDCS CPI-C Nodes Specified	Same	Same	Program	Program
Authentication Performed	Client Workstation	DDCS Server	DDCS Server and Host	Host

AIX and SNA/6000 Considerations

- In the token ring device, the Transmit/Receive Queue Size should be set to 150
- In the Operating System Environment, the *MBUFS* default of 2 MB should be changed from 6 to 8 MB. Use the **netstat** command to query settings.
- Set the SNA Server *IFRAME* to 3900 bytes
- To trace SNA:
  1. start the trace: **sna -t on -l *linkstation-name***
  2. stop the trace: **sna -t off** (the trace is saved in the buffer)
  3. format the trace: **snaformat -l *linkstation-name* -d -s -h**
     (3 files are created: linkstation-name.det, linkstation-name.sum, and linkstation-name.hex)

MVS Considerations

When defining the Performance Group for DB2, ensure that they are defined in the following order where VTAM has the highest service level: VTAM, IRLM, MSTR, SSRV, CICS, DIST.

Notes on DB2 Version 2

Files

The following files are used by DB2 Version 2:

- SQLDBCON - stores tuning parameters for a database
- SQLOGCTL.LFH - tracks and controls all database logs
- Syyyyyy.LOG - database logs
- SQLINSLK - used to ensure that a database is used only by one instance of the database manager
- SQLTMPLK - used to ensure that a database is used only by one instance of the database manager
- SQLSPCS.1 - contains the definition and current state of all table spaces in the database
- SQLSPCS.2 - a copy of SQLSPCS.1 as a backup in case SQLSPCS.1 fails
- SMS files
  - SQLTAG.NAM - used to verify that the database is complete and consistent, with one in each container subdirectory
  - SQLxxxxx.DAT - table file. All data is stored here except LONG VARCHAR, LONG VARGRAPHIC, CLOB, BLOB and DBCLOB data.
  - SQLxxxxx.LF - contains LONG VARCHAR or LONG VARGRAPHIC data
  - SQLxxxxx.LB - contains BLOB, CLOB or DBCLOB data
  - SQLxxxxx.LBA - contains allocation and free space information about SQLxxxxx.LB files
  - SQLxxxxx.INX - index file for a table. All indexes for corresponding tables are stored in this single file.
  - SQLxxxxx.EIX - damaged INX file
  - SQLxxxxx.DTR - temporary data table for a REORG of a DAT file.
  - SQLxxxxx.LFR - temporary data table for a REORG of an LF file.
  - SQLxxxxx.RLB - temporary data file for a REORG of an LB FILE.
  - sqlXXXXX.RBA - temporary data file for a REORG of an LBA file.

To improve performance, cache the directory files, including the database directory. Use the dir_cache database manager configuration parameter.

Tables

- When the default extent size of 32 4KB pages is used, 20 MB is allocated for catalog table space.
- Each page contains 76 bytes of overhead, leaving 4020 bytes for user data. Each row cannot exceed 4005 bytes in length.
- For each user table, the space required is:

  (average row size + 8) * number of rows * 1.5
- LONG VARCHAR or VARGRAPHIC data is stored in 32 KB areas.
- BLOB, CLOB and DBCLOB data is stored in 64 MB areas
- For each index, the space required is

  (average index key size + 8) * number of rows * 2
- When creating indexes, the temporary space required is:

  (average index key size + 8) * number of rows * 3.2
- A database must contain at least 3 table spaces:
  - one catalog table space (SYSCATSPACE)
  - one or more user table spaces (by default USERSPACE1)
  - one or more temporary table spaces (by default TEMPSPACE1)
- An SMS table space is the default:
  - the number of table space containers (segments) must be specified because you cannot add or delete containers after you create a table space
  - the extent size cannot be changed later
  - Specify the value dft_extentsize as the DFT_EXTENT_SZ parameter used with CREATE DATABASE

- For a DMS table space:
  - it can be defined as REGULAR (normal table and index), LONG (for long fields), or LOB (for LOB data)
  - maximum size is:
    - For regular tables and indexes: 64 GB
    - For long fields: 2 TB
    - For temporary data: 2 TB
  - space is preallocated and not extended one page at a time
  - each device can have only one container defined to it. To avoid wasted space, the size of the device and size of the container should be equivalent
  - one page in every container is reserved for overhead
  - three extents in the table space are reserved for overhead. At least two extents are required to store user data.
  - device containers must use logical volumes with a "character special interface". For example , if you create a logical volume with the name "db2lv1", two files are created in /dev: /dev/db2lv1 and /dev/rdb2lv1. The "character special interface" is /dev/rdb2lv1.
    - you must change the owner/group of this device with the command
      chown dbinst1:dbuser /dev/rdb2lv1
      assuming that the instance is dbinst1 and it belongs to group dbuser.
    - SQL0970N will occur if the owner is still root
    - if you get SQL0294, run **db2untag** against the container.
  - containers can be added to an existing table space to increase storage capacity with the ALTER TABLESPACE statement. The contents of the table space are rebalanced across all containers.
  - to find out what the **Status** field means in the output of the LIST TABLESPACES SHOW DETAIL statement, use the **db2tbst** command. For example, to find out about “0x000c”, use **db2tbst 0x000c**
  - to find out about tablespace utilization, use the **tbspstat** command.
  - to remove a load pending state from a table space, use
  - **db2stset <dbpath> <tbspc name> PrettyPlease OFF LOAD_PENDING**
  - This tool replaces LOAD_PENDING with any other state the table space is in after you check it with **db2tbst**. (Note that this tool is not generally available and is provided at the discretion of Service analysts.)
  - to find all locks held, use **db2spec**, then specify **gdbm <databasename>** and **fmlock <output.file>** as options. (Note that this tool is not generally available and is provided at the discretion of Service analysts.)
- Some table space design considerations:
  - you may wish to separate a table space for each very large table, and group all small tables together in a single table space. This separation allows you to select an appropriate extent size based on table space usage.
  - tables less frequently accessed where longer access response time is not an issue (for example, historical data) can be placed in table spaces that are stored on less expensive physical devices that have a slower access rate.
  - using DMS, you can spread your table across three different table spaces: one for indexes; one for LOBs; and one for regular table data.
  - for performance, configure containers to use separate physical devices.
  - back up table spaces instead of databases. This allows you to frequently back up those table spaces that have a large volume of changes, and occasionally back up those table spaces that have a low volume of changes.
  - there must be enough space in a table space container for five extents of the table space.
  - there is very little difference in SMS or DMS table spaces in OS/2.

- Some prefetching considerations:
  - there are two categories: sequential prefetch (which reads consecutive pages into the buffer pool); and list prefetch or list sequential prefetch (which access data pages even when the data pages are not consecutive).
  - the number of pages prefetched can be defined for each table space using the PREFETCHSIZE parameter with CREATE TABLESPACE or ALTER TABLESPACE.
  - PREFETCHSIZE should be a multiple of the EXTENTSIZE. For example, if the extent size is 16 and the tablespace has 4 containers (on 4 physical drives), set PREFETCHSIZE at 64 (that is, EXTENTSIZE multiplied by the number of containers).
  - Version 2.1.1 implements index prefetching, which can be DMS raw I/O

- significant performance implications, particularly for large table scans, can be realized when choosing the right PREFETCHSIZE value. Prefetching can be monitored by looking at the pool_async_data_reads (buffer pool asynchronous data reads) in the database system monitor
- The sequential detection flag (SEQDETECT) must be active in the database configuration. The most recent 8 pages are tracked to detect access that is sequential or near sequential. Access is considered page-sequential if it is within X advancing pages of the current page, where X is the minimum of:
  - 8
  - P/2, where P is the prefetch quantity
  - E/2, where E is the extent size of the tablespace
- list prefetch can be used in conjunction with either single or multiple index access. Steps include:
  - RID retrieval: required pages are compiled by performing matching index scans on one or more indexes
  - RID sort: pages are sorted in ascending page number order
  - Data retrieval: required pages are accessed in order using a sorted RID list
- use the NUM_IOSERVERS database configuration parameter to specify the number of I/O servers to service prefetch request.
- create index table spaces on file containers. No prefetching is done by the database manager for indexes but may take advantage of the operating system or file system prefetching of files.

LOBS

- Turn logging off (use the NOT LOGGED LOB-options clause) when creating a table if a LOB column is larger than 10 MB. If a LOB column is larger then 1 GB, you must turn logging off.
- With restore and rollforward, LOB data that is NOT LOGGED is replaced with binary zeros.
- You can make LOB columns as small as possible with the COMPACT clause of the CREATE TABLE command. However, there is a performance cost (due to storage adjustment) if you append to a compact LOB column in a table and cause the size of LOB values to be increased.

Triggers

- Any trigger dependencies on some other object are recorded in the SYSCAT.TRIGDEP table
- If any dependencies are dropped, the trigger is inoperative.
- If a trigger is dropped, its description is deleted from the SYSCAT.TRIGGERS table. All packages having UPDATE, INSERT, or DELETE dependencies on the trigger are invalidated.

UDFs

- Are external functions which are written in a programming language. They do not contain SQL.
- UDFs are recorded in the SYSCAT.FUNCTIONS and SYSCAT.FUNCPARMS tables.
- If a UDF is dropped, any packages that are dependent on it are marked as inoperative.
- In AIX, a fenced UDF runs by default under a user ID as defined in ~/sqllib/odm/db2udf. If unfenced, it runs under the user ID of the instance.

UDTs

- UDTs are stored in the SYSCAT.DATATYPE table.
- You can only drop a distinct type if:
  - the type is not used in a column definition for an existing table
  - the type is not used in a UDF function that cannot be dropped, if a view, trigger, table check constraint, or another UDF is dependent on it.

Load

- A load has the following phases:
  - load, where data is written into the table
  - build, where the indexes are created
  - delete, where rows that cause a unique key violation are deleted from the table.
- DB2 for Version 2 loading almost eliminates logging
- Creating the primary key before loading the data is more efficient than using CREATE INDEX
- You must run the SET CONSTRAINTS statement after loading a table if the table is left in check-pending state. This will happen if the table has referential integrity and check constraints.
- Use the SAVECOUNT parameter at established intervals. If a failure occurs, you can restart at the last consistent point with the RESTARTCOUNT parameter.
- If failure occurs at the beginning of a load, use the REPLACE option
- If two or more loads are being run, use the REMOTE FILE option to specify the temporary file name, and USING to specify the directory.
- Temporary files are created during a load. Use the USING parameter to specify a directory other than the default in ~/sqllib/tmp.
- Performance considerations
  - increase the UTIL_HEAP_SZ database configuration parameter and reduce the BUFFPAGE, SORTHEAP and DBHEAP parameters, since LOAD does not use any of the engine buffers.
  - if indexes are involved, specify the maximum SORT BUFFER option (in number of 4 KB pages)
  - use multiple sort directories for a temporary space
  - to gather statistics about a load, use the STATISTICS YES option. If data is appended into a table, statistics are not gathered.
  - using the SET CONSTRAINTS command lengthens the total time needed for a load.
  - LOAD performs best with the REPLACE option, unless the table is empty (when INSERT is marginally faster)
  - use the SORT BUFFER and DATA BUFFER parameters if you have sufficient memory. DATA BUFFER can reduce I/O waits when loading long fields or LOBs.
  - specify the USING option for multiple directories in different file systems
  - do not specify COPY YES, for a performance improvement of approximately 3%
  - beyond 10000 rows, LOAD is a better alternative to IMPORT
  - Every time a LOAD is performed, existing indexes for the table are dropped and rebuilt. Hence, if you are LOADing a table that is made up of 2 or more equal size source files and you have a primary key and/or indexes defined to that table, the elapsed time of the LOAD may increase in each subsequent LOAD due to the sort time required to build the indexes
- The LOAD exception table is identical to the original table except that:
  - there are no constraints (NOT NULLS are not considered a constraint)
  - there are two more columns: a timestamp (c1), and a 32-KB CLOB (c2)

Recovery

- Table space roll-forward recovery must roll forward to the end of logs. There is **no** point-in-time recovery.
- To reduce the amount of time to do a backup/restore, increase the number of multiple buffers by using the WITH n BUFFERS and BUFFERS options in the BACKUP or RESTORE commands. If BUFFERS is not specified, the database manager configuration parameters BACKBUFSZ and RESTBUFSZ are used. The ideal size is a multiple of the extent size.
- Use redirected restore to redefine tablespace containers. Redirected restore can only be done through the Database Director. (Select the **Pause to allow table space container redefinition** check box in the Table Spaces - Recover notebook.). A regular restore will fail if any containers within any table space have changed.

Event Monitor

- Use the following command to set the event monitor to track events that occur in the database:

- `CREATE EVENT MONITOR ev1 FOR STATEMENTS WHERE APPL_ID = 'mypgm'`
  `WRITE TO FILE /tmp MAXFILES3 MAXFILESIZE 1000 BUFFERSIZE 125 NONBLOCKED`
  `REPLACE MANUALSTART`
  - the WHERE clause can specify AUTH_ID or APPL_NAME
  - the FOR clause can be one of the following:

Event Type	*Description*	*When*
DATABASE	Database information summary	Last application DISCONNECTs
CONNECTIONS	Connection information sumary	Every application DISCONNECTs
TABLES	Table information sumary	Last application DISCONNECTs
STATEMENTS	SQL statement information sumary	Each SQL statement is invoked
TRANSACTIONS	Transaction information sumary	COMMIT or ROLLBACK occur
DEADLOCKS	Deadlock information sumary	A deadlock occurs (-911)
TABLESPACES	Table space information sumary	Last application DISCONNECTs

- files can be REPLACE or APPEND
- the files generated are *nnnnnnnn.EVT*
- BUFFERSIZE memory is allocated from the DBHEAP
- To turn on event monitor *ev1*, use the command: SET EVENT MONITOR *ev1* STATE = 1

Miscellaneous Performance Considerations

- Increase the MINCOMMIT value in the database configuration. This parameter delays the writing of log records to disk until a minimum number of commits has been performed.
- The query optimization class is the level of optimization used by the optimizer to generate access plans:
  - For static queries: use the QUERYOPT *n* option in the BIND command
  - For dynamic queries: use SET CURRENT QUERY OPTIMIZATION = *n*, where *n* is:
    - 0, for a minimal amount of optimization
    - 1, for a level that approximates that used in Version 1
    - 3, for a moderate amount of optimization that approximates that used in DB2 for MVS.
    - 5, for optimization that is sensitive to the operating environment and to dynamic or static queries. This is the default class.
    - 7, for a significant amount of optimization.
    - 9, to use all available optimization techniques.
  - use a low optimization class (0 or 1) for very short run times of less than one second.
  - use class 1 if there are many tables with many join predicates on the same column.
  - use classes 3 or 5 for long-running queries of more than 10 seconds
  - do not use class 9 under normal circumstances because it will affect performance
- In Version 2.1.1 the number of rows specified in the OPTIMIZE FOR clause will be used by the optimizer to determine the best access path
- Local predicates are assessed in reverse order. Hence, if you have a highly selective local predicate in the WHERE clause, place it last. This assumes that you have an index on that column.
- If you use functions such as TRANSLATE or UCASE on any character column, the optimizer cannot use the start/stop index key searches. That is, **all** entries in the index must be compared. Start/stop key searches only select index rows that meet the predicate value. (See the appendix on differences in predicate evaluation for a detailed discussion.)
- Use the explain utility to analyze a projected access plan:
  - run EXPLAIN.DDL from /sqllib/misc to create explain tables
  - use SET CURRENT EXPLAIN SNAPSHOT YES to capture dynamic SQL in tables and process SQL
  - use SET CURRENT EXPLAIN SNAPSHOT EXPLAIN to capture SQL in a table without processing it
  - the Visual Explain tool provides a graphical interface for viewing access plans based on explains.

- Use **db2look** to get information from the system catalogs, and output information on table spaces, tables, indexes, and columns. This information can then be sent together with a database's DDL to simulate a user's environment. Example: **db2look -d mydb -m -a -o mydb.mimic**
- There are two tools available that will give a detailed output of the SQL statement access path
  - **.db2service**
    - create a file (e.g. explain.out) with the SQL statement
      e.g. .db2service <SQL statement>;
    - use the CLP with the input file just created above
      e.g. db2 -tvf explain.out
      this will result in a detailed output in file *serv.log* in the sqllib/db2dump directory
  - **db2exfmt**
    - you will first need to create the Explain tables (2 bullets up)
    - create a file (e.g. db2exfmt.sql) with the SQL statement
      e.g. explain plan for <SQL statement>
    - issue the following command from the CLP
      select explain_time, source_name, source_schema from explain_instance
    - using the information obtained above (last SQL statement should have the latest time stamp), issue the command: db2exfmt and answer the prompts
    - the output will be in the file you specified
- Use the following database manager configuration parameters:
  - ASLHEAPSZ on the DB2 server to allocate a cache for row blocking
  - RQRIOBLK on a client workstation to cache rows in remote applications
  - The optimizer uses the following, along with other factors, to estimate how much of the buffer pool will be available at run-time for an access plan:
  - The database configuration parameter AVG_APPLS
  - The database manager configuration parameter CPUSPEED
- If you use the WITH DISTRIBUTION clause with the RUNSTATS command, the NUM_FREQVALUES and NUM_QUANTILES database configuration parameters determine if distribution statistics are collected. The size of the statistics heap (STAT_HEAP_SZ) may limit the number of statistics computed and stored.
- Pages can be written from the buffer pool to disk if the percentage of space occupied by changed pages in the buffer pool exceeds the value of the database configuration parameter CHNGPGS_THRESH. For this to occur, you must specify asynchronous page cleaners using the database configuration parameter NUM_IOCLEANERS.
- Connect times can be reduced by changing the values of the database manager configuration parameters DIR_CACHE and MAX_IDLEAGENTS.
- If you want prefetch quantity greater than 16 pages, ensure that the database configuration parameter BUFFPAGE is above 1100. Note that there is little benefit in specifying a large buffer pool for table scans. Use any additional memory for a larger sort heap.
- Ensure that the database configuration parameters PCKCACHESZ and APPLHEAPSZ are set in proportion to each other. To determine the package cache hit ratio, look at the database system monitor values: it should be the quotient of package cache inserts / package cache lookups
- In complex queries, static SQL may perform worse than dynamic SQL because it is hindered by host variables. The optimizer doesn't have all the information it needs to find the best access plan. In simple OLTP queries, static SQL is better then dynamic because of the dynamic prepare time. Embedded dynamic SQL, when loaded with parameter markers will have the same access plan as embedded static SQL that uses host variables. Parameter markers are therefore not recommended.
- In Version 2.1.1, it is possible to release read locks when closing cursors that are operating under either the Repeatable Read or Read Stability isolation levels. In Embedded SQL, use the RELEASE clause with the CLOSE statement.
- Use the ACTIVATE DATABASE command after starting DB2 to improve initial connection time. The database will remain active until explicitly deactivated with the DEACTIVATE DATABASE command.
- If you notice considerable locking contention due to next key locks and are sure that RR isolation level is not required for your DB2 instance, you can set the environment variable DB2_RR_TO_RS=ON. For example, in AIX you would issue the commands
  - export DB2_RR_TO_RS=ON

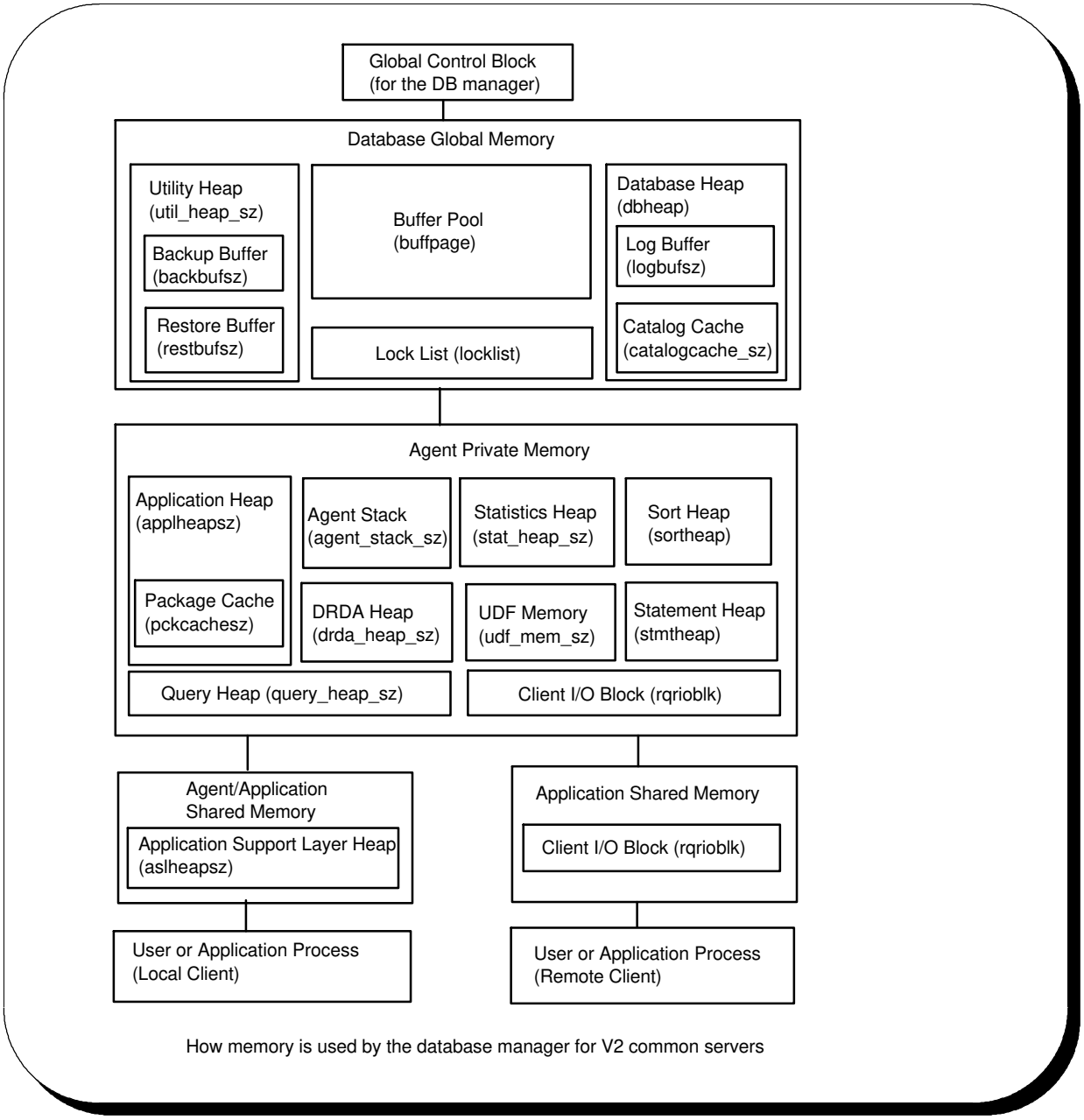
- db2start
- If you need to define table constraints after the table had been defined (CREATE TABLE), **do not** create each constraint individually (ALTER TABLE ADD CONSTRAINT). They should be defined within a single ALTER TABLE SQL statement. Degredation of INSERTS of up to 100% may result if you add 2 or more such constraints individually.

Distributed Unit of Work (DUOW) and 2-Phase Commit

- DUOW (or CONNECT Type 2) is supported in application programs written with either static or dynamic SQL or CLI. DB2 CLP does not support this function.
- Type 2 CONNECT is set with the CONNECT option in the **PREP** command.
- Use the SQL statement SET CONNECTION *hostname* to change the state of a connection from dormant to current.
- In CLI, the SQL_CONNECTTYPE option in the SQLSetConnectOption() function determines the scope of a connect. Valid values are: SQL_CONCURRENT_TRANS (for individual commit scopes); and SQL_COORDINATED_TRANS (corresponding to Type 2 CONNECT and working in conjunction with the SQL_SYNC_POINT connection option).
- During installation of DDCS, choose DRDA 2PC if the gateway will be used as a Sync Point Manager
- Use SYNCPOINT(TWOPHASE) in the **PREP** command if the application does updates to two databases in a single Unit of Work (UOW)
- In a 2PC scenerio, the client coordinates the UOW and uses a transaction manager to track the completion of work. The transaction manager database is determined on the client from the database manager configuration parameter TM_DATABASE.
- A transaction manager (TM) database instance must be able to connect to all other databases participating in a distributed transaction, and vice-versa.
- Once the client receives a positive PREPARED response from all databases, all work can be committed. If any database fails to commit work, the transaction manager retries the commit. The database manager configuration parameter RESYNC_INTERVAL determines how long the TM database waits between attempts.
- Sync Point manager is only required for host DRDA databases
- In VTAM, use SYNCLVL=SYNCPT in the DB2 APPL statement

CAE/Win Performance Parameters

- The environment for CAE/Win is defined in the file \Windows\db2.ini.
  - DB2CODEPAGE must be set if the target database is DB2/MVS. For example
    [DB2 Client Application Enabler]
    .....
    DB2CODEPAGE=437
- The CLI configuration options file is defined where CAE/Win is installed e.g. \sqllib\win. Within this file, a section is defined for each database alias cataloged:
  - SCHEMALIST="'schema1','schema2',....." restricts the scope of table being queried by the application for table information. This is particularly crucial if e.g. MS Access is the end user application and DB2/MVS is the target DBMS. You may not want the application to retrieve table information for any SYSIBM tables. Such applications commonly query table information before actually executing the intended SQL DML. This keyword can be used in conjunction with the DBNAME and TABLETYPE parameter to further restrict the number of tables for which information is returned.
  - SYSSCHEMA=sysschema. This keyword indicates an alternative schema to be searched in place of the SYSCAT (or SSIBM, SYSTEM) schemas when the DB2 CLI and ODBC Catalog Function calls are issued to obtain catalog information. This is a major performace item when the target database has a large number of tables within a database or subsystem.
  - TABLETYPE="'TABLE'|,'ALIAS'|,'VIEW'|,'SYSTEM TABLE'|,'SYNONYM'". This keyword can be used to reduced the time it takes for the application to query table information.
  - ASYNCENABLE=0 | 1. To enable asynchronous features. 0 - disabled, act ssimilarly to previous versions. 1 - enabled.



How memory is used by the database manager for V2 common servers

Notes on DB2 for NT

- The **winmsd** command reports on all I/O, IRQ and DMA channels
- The **regedt32** command displays the Registry
- Use **snacfg /display** to dump SNA configuration
- Use **\system32\repair\rdisk.exe** to make the repair disks
- The **net config rdr** command reports on the active MAC address (for example, token ring UAA or LAA)
- To add your own Locally Administered Address (LAA):
  1. Use **regedt32** and then select the following choices to go to the ibmTokM1 window: *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\ibmTokM1*
  2. From the **Edit** menu, choose **Add Value**. For **Value**, use your Network address and select *REG_SZ* for data
  3. Enter your UAA (it's 12 hex digits)
  4. Exit
- DB2 for NT requires Windows NT Version 3.5.0 or higher, and SNA server Version 2.1 or higher
- The database code page is derived from the value of the DB2CODEPAGE environment variable if it is set, or taken from the ANSI code page in the Registry
- DB2 supports raw I/O on NT for both logical and physical drives. Prior to Version 2.1.1, only entire physical drives were eligible for raw I/O. For example, to use the following as raw I/O:

  \\.*PhysicalDrive***x**

  where ***x*** is an integer that is the disk ID obtained from Disk Administrator. To use logical drives (partitions), use the following format

  \\.\e:
- You can eliminate file caching on DB2 by setting the DB2NTNOCACHE environment variable to 1. This applies to all files except LONG FIELDS and LOBS. More memory will then be available for use by the database.
- In a non-paging situation, a process's working set can grow as large as needed. When paging occurs, the maximum working set is limited to 1 MB. To explicitly set the working set for DB2, specify a range of values with the environment variable DB2NTWORKSET=*min,max*. On a system with 32 MB of RAM, a good minimum working set is 15-20 MB. The maximum value should be less than the the physical RAM size.
- An application must explicitly COMMIT all work before terminating. Unlike OS/2 or AIX, changes to the database are not automatically committed on normal termination of the application.
- To ensure that DB2 is not shut down when you log off, start it automatically from the **Control Panel - Services** panel. If the DB2 security services are not started, SQLCODEs -1042 or -1402 are encounted on a CONNECT.
- LOAD from a tape device is not supported.

Memory Requirements

NT requires a minimum of 16 MB RAM for basic file and print services. The following tables show the additional memory overhead for DB2 (in MB):

(Size in MB)	Base	1st connection	Additional connection	Additional concurrent DB	Additional concurrent instance
Single	8	3	0.5	2	8
Server	8	3	0.5	2	8
CAE			2		

(Size in MB)	Base	1st connection	Additional connection	Local Client
DDCS single user				7
DDCS multi-user	3	2	0.5	2

Security

- When a user connects to DB2, authentication is done as follows (if **Authentication=Server**)
  - on the primary domain controller (PDC), unless DB2DMNBCKCTRL is set
  - if the environment variable DB2DMNBCKCTRL=*xxx* is set (where *xxx* is the backup domain controller name), authentication is done on the backup domain controller (BDC). The BDC could either be the same system as the DB2 server or another system on the LAN segment. If it happens to be on a remote LAN segment, authentication on the PDC is avoided.
  - if the client is coming in from a trusted domain (take, for example, domain A) and the DB2 server is on domain B, the client will be authenticated in its own domain (A) and allowed to connect to DB2.

Performance Considerations

- Use the **DB2PERFI.EXE** program to install and un-install the performance DLL. When you do this, the Performance Monitor-Add to Chart-Object window will contain the entry “DB2 NT Database Manager”. The counters will essentially cover similar items in the Snapshot Monitor - Database Manager Snapshot window.
- To monitor performance of DB2 from a remote client, you will need to explicitly connect to the database with a user ID and password.
- To clear the performance values, use the **DB2PERFC.EXE** program. By default, it resets all databases in the instance executing the program. You can specify the database to reset by passing the database name as a parameter.
- Turn on the following performance counters (in addition to DB2's) to monitor the system:

Memory: Pages/sec	specifies pages read/written to disk to resolve memory swapping
Memory: Available Bytes	displays free physical memory. Should not stay constantly below 1 MB.
Processor: % Processor Time	indicates a CPU bottleneck when high (for example, over 90%)
Process: % Processor Time	determines which process is "chewing" up CPU
Process: Page Faults/sec	specifies pages that had been in memory but are now in the swapper
Processor: Interrupts/sec	determines if CPU activity is caused by applications or by service hardware interrupts
Physical Disk: %Disk Time	determines if I/O causes a bottleneck
Physical Disk: Disk Queue Length	specifies pending I/O requests. If it is greater than 2, there is disk congestion

- To determine if NT configuration is best set for up to 64 remote users:
  1. From the **Control Panel - Network** panel, choose **Server**. This option lets you configure the server's resource allocation.
  2. Specify one of the following:
     Minimize Memory Used - for up to 10 remote users
     Balanced - for 10 to 64 remote users
     Maximized Throughput for File Sharing - for 64 or more remote users. This is the default.
- NTFS is best used on volumes greater than 400 MB

General Notes

- The database administrator's ID must be 8 characters or less (note that the user ID “ADMINISTRATOR” is not valid)

- In the services file (typically \winnt35\system32\drivers\etc\services), you need to add two entries for the connection and interrupt ports for TCP/IP. Ensure that you make a comment after each entry to distinguish them.
- Example:

```
db2inst1        3700/tcp        # DB2 connection port
db2inst1i       3701/tcp        # DB2 interrupt port
```

Notes on DB2 for Solaris and HP-UX

The following configurations are for Solaris machines. For HP-UX, remove the prefixes “msgsys:msginfo_” or “semsys:seminfo_”.

Kernel Configurations

- **set msgsys:msginfo_msgmni = xxx** (determines the number of possible local connections)
  In Version 1.2, every local connection requires 6.5 possible connections, and every remote connection requires 1. In Version 2, every local connect requires 3 possible connections, and remote connections don’t require any.
- **set msgsys:msginfo_msgtql = xxx**
- **set msgsys:msginfo_msgmap = xxx**
- **set semsys:seminfo_semmni =xxx**
  In Version 1.2: 2 for local and 1 for remote; In Version 2: 4 for local and 1 for remote.
- **set semsys:seminfo_semmns = xxx**
- **set semsys:seminfo_semmap = xxx**
- **set shmsys:shminfo_shmmax = xxxxxxxxxxxxx** (determines the maximum amount of memory that can be used by DB2)

Nodelock file

- The NetLS nodelock file should be in the following directories
  - Solaris: /var/netls
  - HP-UX: /usr/netls

Messaging

- **msgsys:msginfo_msgmni** - the number of message queues that are active. Message queues are used to communicate between processes, and many are used with CLP.
- In Version 2.1.1 message queues are used for things like page cleaners and prefetchers. This is not an issue for connections.
- -*SUGGESTED VALUE*: A minimum of 400, as recommended in the *Installation Guide.*
- **msgsys:msginfo_msgtql -** the number of active messages in the system that were sent but are not yet received. Since DB2 does not typically send more than one message per queue, we recommend **at least** msgtql = msgmni. (The *Installation Guide* recommends that msgtql be greater than msgmni, with a minimum of 600.
- **msgsys:msginfo_msgmap** - the size of the structure used to hold the mni. A value of (value of msgtql + 2) is recommended.
- **msgsys:msginfo_msgmax** - must be set to 64 KB for DB2 (specifically CLP) to start
- **msgsys:msginfo_msgmnb** - must be set to 64 KB for DB2 (specifically CLP) to start
- **msgsys:msginfo_msgseg** - the number of segments used to hold messages. A value of 256 KB is recommended. This value multiplied by the value of **msgssz** should be larger than 64 KB. (This is only an issue if you have local access via the CLP.)
- **msgsys:msginfo_msgssz** - the size of the message segments. This value should be small: 16 or 32 bytes.
- **msgsys:msginfo_msgseg** - should be a large value. This value multiplied by msgseg should be larger than 64 KB. (This is only an issue if you have local access via the CLP.)

Semaphores

- **semsys:seminfo_semmni** - the number of semaphore identifiers. One semaphore identifier can have more than one semaphore associated with it. The suggested value is 512.

- **semsys:seminfo_semmns** - the maximum number of semaphores. This value should be at least `2 * semmni`
- **semsys:seminfo_semmap** - the number of semaphore MAP entries. This value should be `semmns + 2`
- **semsys:seminfo_semmnu** - the maximum number of processes having a pending semaphore undo request. This value should be set to **at least** `maxagents * 2`. The *Installation Guide* recommends a minumum of 1400.
- **semsys:seminfo_semum**e - The Solaris system default is "OK".

Shared Memory

- **shmsys:shminfo_shmmax** - the maximum number of bytes of shared memory that an application is allowed to take. This value should be as big as the largest shared segment to be allocated. Very often this value will be the buffer pool, so the rule of thumb is that it should be **at least** 1.11 times the size of the buffer space (BUFF PAGES * 4096 * 1.11). Note that this space is not preallocated, so the definition can be large.
- **shmsys:shminfo_shmseg** - As per the *Installation Guide*, this value should be 10
- **shmsys:shminfo_shmmni** - As per the *Installation Guide*, this value should be 256

General Notes on Solaris

- With the default DB2 for Solaris configuration, DB2 takes 5 MB of memory just for DB2START to be invoked.
- For the first connection to a database, with all default configuration parameters, DB2 takes an additional 33 MB. Subsequent connections require much less additional memory.
- Redirected restore via db2dd is not supported on the Solaris and HP-UX platforms due to the absence of the db2dd GUI utility. A future release of DB2 for NT/95 (due in Sept) is expected to support remote backup/restore to a DB2 for Solaris database.

Corrections to the Version 2.1.0 DB2 for Solaris README File

- The discussion on using raw devices for logs provides incorrect instructions. (This is not a common situation and is not expected to be used often.) The corrected instructions are in the Version 2.1.1 refresh readme file.
- The readme file states that for maximum performance, you need a particular patch from Sun. However, note that **THIS PATCH IS REQUIRED** not just for performance but to allow DB2 for Solaris to run. **Get this patch from SUN.**

Appendix: Useful UNIX Commands

The following commands will help you to monitor the performance of your system. They are available for AIX and other UNIX systems:

- **genkld** - view all libraries loaded in shared segment
- **ps eww <pid>** - view all environment variables defined for a process
- **sar -r <interval> <number>** - view paging statistics
- **sar -u <interval> <number>** - view CPU activity
- **vmstat <interval> <number>** - view statistics on processes, virtual memory, disk and CPU
- **vmstat -s** - view summary info
- **iostat hdisk0 <interval> <count> -** disk usage for a specific drive
- **svmon -i <interval> <count>** - capture and analyse virtual memory
- **filemon -** give detailed information about file accesses
- **iostat -d** - view disk usage since boot time
- **netstat [-A | -a | -i | -d | -n | -p | -r | -s | -u | -v]** - view network information on sockets or adapters (a,i,u,v), protocol (d,p,s), address (A, n) and routing (r).
- **ipcs -a -m -q** - list all IPCS message queues and shared memory on the system
- **ipcrm -q <messageid>** - remove the IPCS message queue *messageid*
- **ipcrm -m <sharememoryid>** - remove the IPCS shared memory *sharememoryid*
- **tprof -p ksh -x sleep 100** - monitor the system for 100 seconds
- **tprof -p db2sysc -x sleep 100** - monitor DB2 for 100 seconds. Must be run in ~/sqllib/adm
- **filemon -o <out.file> -v** - start monitoring the performance of the filesystem.
  trcoff - stop monitoring the filesystem performance.
- **slibclean** - remove modules in the kernel and library memory that are not currently used
- **dump -H** - dump the header of the loader section. Valid for executable files.
- **dump -a** - dump the archive header of each member
- **xmperf** - this is a graphical tool that is useful for monitoring all system activity. Available only with AIX Performance Toolbox (PTX) (5696-900).

Appendix: Notes on RAID

RAID (Redundant Array of Independent Disks) combines a number of disk drives into a single unit. The basic idea behind a RAID is to provide disk parallelism, without increasing the probability of losing data due to the failure of any single drive. Note that proper DB2 backup/recovery procedures are also designed to protect the database against the failure of one or more drives. The advantage of a RAID is that the failure of a single drive does not require the system to be shut down. Note that after a failure, a RAID performs in a degraded mode until the problem disk is replaced and its data is automatically reconstructed. Unfortunately there is a price to be paid for this fault tolerance -- typically a combination of additional disk space and write performance.

RAID Levels

There are multiple RAID levels to choose from:

RAID 0

Level 0 stripes data uniformly across a set of disks without any redundancy. This provides I/O parallelism, but not fault tolerance. DB2 table spaces can be configured to run in this mode without requiring a RAID system. They will also typically perform better since DB2 gets more control over data placement, and can exploit it.

RAID 1

Level 1 mirrors the data on different drives for fault tolerance and high performance. This is the highest performing RAID configuration, but also has the highest space overhead. It is a good choice when storing DB2 logs.

RAID levels 3 and 5 use data encoding to reduce space overhead, but do affect the cost of performance.

RAID 3

Level 3 stripes data across multiple disks and uses an additional disk to store parity information. In the event of a disk failure, this parity information can be used to reconstruct the data from a lost drive. The data is striped finely, so all drives need to be accessed for every database I/O. This makes it well suited for a large amount of sequential I/O, but makes it totally unsuitable for transaction workload.

If you do decide to use this RAID mode, a good configuration will have the table space prefetchsize equal to the extentsize. 8 to 48 pages are good values to use for these. You may want to divide the RAID storage into multiple pieces and allocate them to multiple containers. This provides good flexibility for sharing disk space among table spaces and adding containers to table spaces as space requirements grow over time. Note that RAID 2 is similar to RAID 3 except that it uses a different error correction technique which is complex and expensive. Therefore it's not a good choice for DB2.

RAID 5

Level 5 stripes data in larger block sizes. This allows I/O to be satisfied without accessing all the drives, thus making it possible to service multiple concurrent I/O requests on the drives. For this reason it's more suitable for transaction processing than RAID 3. It also distributes the parity disk on multiple drives, which helps to alleviate the workload on parity data. This makes RAID 5 the most popular RAID level. Note that every time data is updated on any of the drives, the parity needs to be updated as well and performance is degraded. Also, during data reconstruction after a failure the performance is further degraded since all the drives must be read in order to service requests to the failed drive. For these reasons, the choice of using this level should be made purely for fault-tolerance. If I/O parallelism is what you're looking for, DB2 provides excellent support for it without the need for a RAID.

RAID 5 systems are usually fairly configurable. They allow the disk space to be divided into multiple RAID groups (typically about 5 disks per group). The block size can be configured to suit the workload. A RAID group can be further subdivided for better flexibility of space allocation. On AIX this can be done using the logical volume manager.

RAID Block Size

Choosing a proper RAID block size is very important. The choice depends on the type of workload:

For workloads that involve large amounts of sequential write I/O (such as database restore and load), a large block size helps reduce the number of times parity information needs to be written. This can usually make a significant difference to performance. Blocksize values of 128 KB to 256 KB are good values to use if you're looking for good load/restore performance. Spreading the containers of a table space over multiple RAID groups can improve performance significantly, since DB2 backup-restore can exploit this parallelism. You should choose a backup-restore buffer size that is large enough to read or write at least one table space extent from each of the containers in parallel (C times E, where C is the number of containers and E is the extentsize). As in RAID 3, the prefetchsize must equal the extentsize. This discussion also applies to transaction workload where the database size is much larger than the RAID block size and the access is fairly random.

For good query performance that typically involves sequential scans of data, a small block size helps exploit parallelism between disks. Setting the extentsize and prefetchsize to a multiple of the blocksize enables DB2 to exploit that parallelism. 4 to 8 pages of RAID blocksize and 20 to 40 pages of extentsize/prefetchsize are good values to use for a 5-disk RAID group. Note that this performance is obtained at the cost of restore/load performance.

Other important things about RAIDs

- RAIDs protect against single drive failures only. DB2 backup/restore procedures must be used to obtain complete data protection.
- RAIDs provide tolerance for disk failures only, and you must consider the other parts that can fail and bring the system down. For example, consider dual disk controllers which usually enhance performance.
- Use fast or wide disk controllers.
- Consider using a recoverable disk cache to recover some of the performance lost to maintain parity information.

Appendix: Differences in Predicate Evaluation After Migration

The description below may assist users who perceive a preformance degradation in some SQL queries after migrating from DB2 Version 1.2 to Version 2.1.

Version 2.1.x

- For disjuncts:
  - Given "WHERE P1 or P2 or P3":
    - Version 2 evaluates predicates in the order of P1, P2, and P3. (DB2 terminates the evaluation as soon as TRUTH is found.)
    - This clause is the same as "WHERE (P1 or P2) or P3".
  - If index access can be used for "WHERE P1 or P2 or P3":
    - for index ORing, the order is P3, P2, and then P1.
    - otherwise it is P1, P2 and then P3.
  - Given "WHERE C1=0 or C1=1 or C1=2":
    - QRW will convert the predicate into "C1 in (0, 1, 2)".
    - QRW may or may not generate an equivalent OR predicate:
    - "WHERE (C1=0 or C1=1) or C1=2"
    -
- For conjuncts within the same category or bucket:
  - Given "WHERE P1 and P2 and P3":
    - Version 2 evaluates predicates in the order P3, P2, and then P1. (The evaluation is terminated as soon as FALSE is found.)
  - If index access can be used for "WHERE P1 and P2 and P3", the order of evaluation of predicates is:
    1. Start/stop predicates. (For example, C1=1 where C1 is the major column of an index.)
    2. Index sargables
    3. Fetch sargables
    4. Residuals

- Predicates (conjuncts) are classified into 4 categories (or buckets) based on their operators. Predicates with the highest priority are evaluated first.

1. priority 1 (highest) is for the comparison operations including =, <>, >, >=, <, <=, like, between, is null, is not null
2. priority 2 is the default category (for example, cheap UDFs)
3. priority 3 is for expensive UDFs, case statements, or disjuncts
4. priority 4 (lowest) is for subqueries and LOB type operations

Within a category, predicates (or Boolean terms) are evaluated as described in the above bullet on disjuncts.

- Miscellaneous
  - In general, the optimization level does not affect the predicate evaluation order, although we may get an entirely different plan (and hence predicates may be evaluated in a different order).
  - Applying DeMorgan's law does not affect the predicate evaluation order.

Version 1.2

How does Version 1 order predicates? The short answer: unpredictably (almost).

- For pure local Boolean functions (those involving just one table) in a relation scan, evaluation is from right to left

- For pure local Boolean functions (BFs) in an index scan, evaluation is left-to-right. (The LIFO walk of the WHERE clause's BFs to attach the eligible ones to the greedy node, which DB2 then attaches to each index scan. This is true of ipreds and dpreds, separately.)
-
- For example, in an index scan of x(a,b,c) for:

```
SELECT * FROM t WHERE b=1 AND d=3 AND c=2 AND e=4
```

the ipred (index sargable) would apply b=1 first, then c=2. The dpred (sargable) would apply d=3 first, then e=4. If a relational scan were chosen, the dpred would apply, in order, e=4, c=2, d=3, and then b=1.

- Within a BF that's a tree of ORs and possibly ANDs (under the OR somewhere), it's strictly right to left. (The codegen builds operands of ORs and ANDs in the order it is given them, but semantics builds them backwards as a result of the way it pushes and pops ORs' and ANDs' operands: for example, it pushes the left, then pushes the right, then pops the right but calls it the left operand, then pops the left but calls it the right.)
-

This applies only to "Natural" BFs, ORs, and ANDs, and not to those resulting from applying deMorgan's law or otherwise internally created (for example, INs transformed to ORs for index ORing).

Silly parentheses can complicate things, too. For example, for the following clause:

```
where (p1 and p2) and (p3 and p4)
```

A thread like the following may be returned on a relational scan:

```
p4
p2
p1
p3
```

(This is possibly an outcome of the recursive BF-identifier in qmod.)

Multi-table Boolean functions (join predicates) are difficult to predict.

Appendix: Sort Heap memory changes in v2.1.2

A change that was made in DB2 Version 2.1.2 that affects how DB2 utilizes memory for sorts, can potentially affect performance. This change applies to all platforms.

The bottom line is that sorts will start using more memory if **SORTHEAP** is larger than the optimizer estimation and the optimizer estimation is less than 1MB. You can fix small sorts by lowering **SORTHEAP** (as in TPC-C-like transactions), but this could impact large sorts.

Several reported instances of poor TPC-C performance were due to an incorrect database configuration setup. When tuning for performance, look for parameters that can be reduced, not just ones that can be increased.

A specific problem occurs when **sortheap** is set too big or is left at the default of 256 pages. For TPC-C (or any true OLTP workload) there will be few or no sorts, and any that occur will be small. Therefore the sortheap should be small (that is, as small as it can be which is 16 pages right now).

Note that in Version 2.1.0 this situation wasn't a problem because the optimizer told sort how much memory was needed. However, we discovered that *sometimes* the optimizer does not estimate correctly the number of rows to be sorted. In Version 2.1.0 it was possible for the optimizer to estimate 1 or 2 rows when in fact there were 100K rows. Disaster resulted! While we continue to make the optimizer smarter, we decided to make the worst case scenario a little less severe and implemented the following formula for the size of the sortheap allocated:

sort area acquired = min (sortheap, max(256, opt estimate))

In summary, reduce the sortheap configuration parameter if you want less than a 1-MB sortheap.

Appendix: AIX Example of a UDF

```
/*****************************************************************************/
/*  Program: view.c                                                          */
/*  Date:    Jul 24th 1995                                                   */
/*  Author:  Finn McCann, Liwen Yeow                                         */
/*  Description: To test UDF usage to view a AVI, MPEG file                  */
/*                                                                           */
/*  Compile/Link: xlc -I/usr/lpp/db2_02_01/include -O -c view.c              */
/*                xlc -o view view.o -ldb2 -lc -L/usr/lpp/db2_02_01/lib -bE:view.exp */
/*                    -e view                                              */
/*****************************************************************************/
#include <stdio.h>
#include <sqlcli.h>
#include <sqludf.h>
#include <stdlib.h>
void SQL_API_FN view(SQLUDF_BLOB *movie_struct, long *out,short * in1null, short * outnull,
SQLUDF_TRAIL_ARGS)
{
  FILE *temp_file;
  long rc;
  *out = 0;
  temp_file = fopen("/tmp/movie_file","w+b");
  if (temp_file == NULL)
  {
    strcpy(sqludf_sqlstate,"38999");
    strcpy(sqludf_msgtext,"open error in function fopen");
    rc = 1;
    goto end_pgm;
    }

  rc = fwrite(movie_struct->data,1,movie_struct->length,temp_file);
  if (rc < 0)
  {
    strcpy(sqludf_sqlstate,"38999");
    strcpy(sqludf_msgtext,"write error in function fwrite");
    goto end_pgm;
  } /* endif */

  fclose(temp_file);
  putenv("DISPLAY=hellcat:0");  system("/u/bio/avis/xanim /tmp/movie_file");
  remove("/tmp/movie_file");
  end_pgm:  return;
 }
```