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

May 17th 1996

Note: This document can be distributed freely to anyone who has a need to configure and tune DB2/2, DB2/6000 V1.x or DB2/2, DB2/6000, DB2/NT, DB2/Solaris and DB2/HP-UX v2.1. Distribution of this document must be made in whole. Contents of this document are the personal notes of the author and are **not** that of IBM. The author does not guarantee accuracy of the information herein and may make changes at any time.

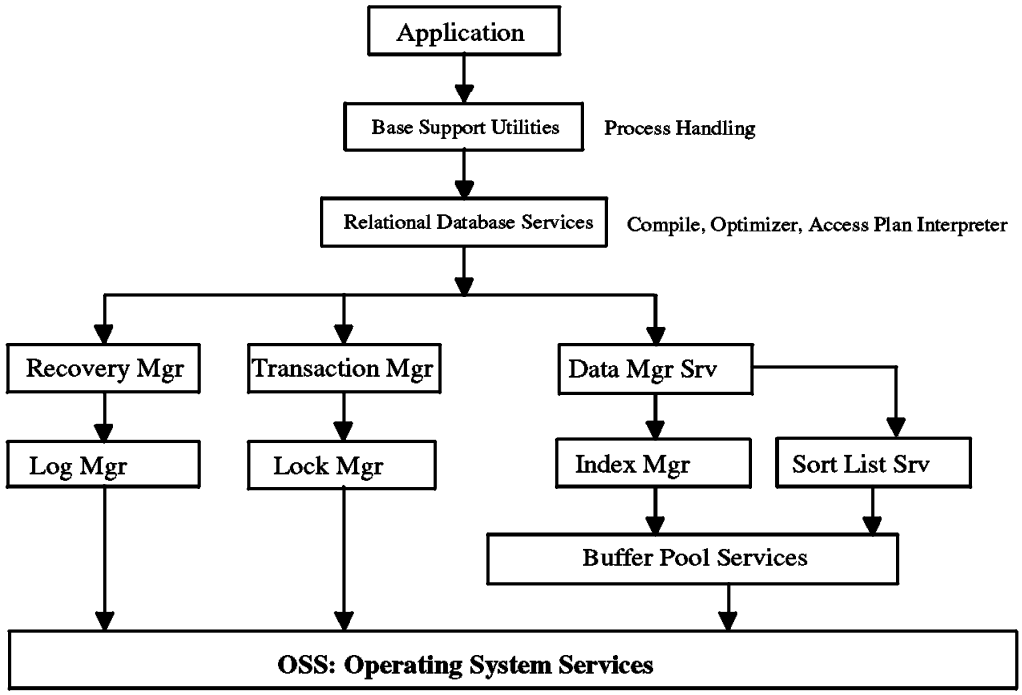
Acknowledgement of contributors

- Toronto DB2 Performance Dept
- Surendra Verma
- Bob Harbus

After reviewing this document, if you would like to have changes, additions and corrections made, please send your note/e-mail to me at the address on the cover page. Your comments on this document, whether positive or negative is greatly appreciated.

Version 1.x

Component Overview



Database Design

- Normalized database design
  - no data redundancy - simple maintenance
  - more table - more i/o
  - more joins - complex SQL
  - preferred for Transaction oriented Apps
- Denormalized database design
  - some redundancy - harder to update
  - fewer table - less i/o
  - fewer joins - simpler SQL
  - preferred for Query Apps
- Additional summary data tables
  - more tables - more disk space
  - use to offload from production - especially decision support
  - duplicated data - loss of data consistency
- Small tables may have contention problems. Large tables rarely have such problems.
- Large tables may impact utilities - IMPORT. On-line apps may perform better on large table because page contention is less likely.
- Table splitting to reduce scan cost
  - horizontally - by key range or current/history

- vertically - by sets of columns to increase rows per page - duplication of matching key columns. Must consider JOIN overhead.

- Consider row length and positioning of columns
  - row lenght vs page size - longest row is 4K; max of 127 rows per page
  - rows less then 24 bytes waste space
  - rows over half or third waste space
  - important to minimize number of pages
  - minimize variable lenght columns. Do not use LONG VARCHARS unless intent is to have max row lenght. - increase CPU consumption and logging. Requires 2 extra byte. Columns following var lenght columns do not have fixed position in row - CPU cycles required to calculate position.
  - for read only tables, place var lenght columns at end of row. For updated tables, place frequently updated columns, whether fixed or var at end of row.
  - do not define columns that are < 18 bytes as VARCHAR.
  - DECIMAL columns consume more CPU resources
- NULL values
  - useful if numeric data with missing value is computed
  - more complex programming
  - additional disk storage - consider NOT NULL WITH DEFAULT. NULL requires 1 extra byte
  - in comparison, NULL values are treated as unequal; selecting DISTINCT values, NULL is presented only once; if counting DISTINCT, NULL values are not counted; NULL values are ordered last.

- Index Selection
  - required to enforced Unique Key Values; support Primary keys
  - desirable for
    - WHERE predicate evaluation
    - ORDER BY and GROUP BY
    - index only access
    - foreign keys
    - support JOINS
  - balance retrieval benefits vs update cost of indices
  - indices with Low Cardinality hurt index maintenance performance
  - DB2 Optimizer users statistics gathered by RUNSTATS
  - additional indexes increase the possibility of deadlocks as they provide alternative update paths to same data page

Application Design

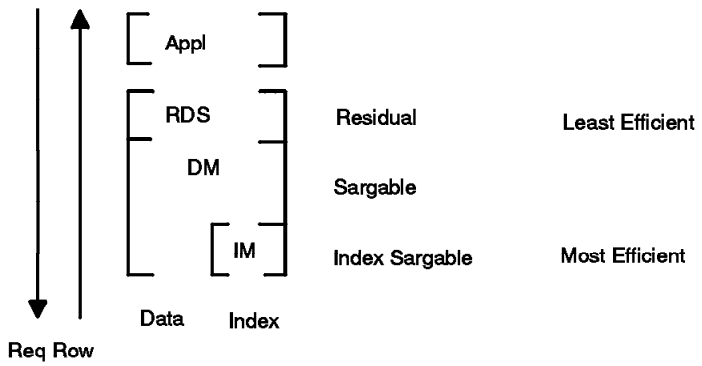
- Lock waits and Deadlocks problems
  - lock escalation hurt concurrency level - whoever gets escalated prevents other from accessing and /or updating data
  - inappropriate isolation level
  - updates in different order by different programs - update in standardize order
  - large number of table indices
  - inappropriate use of TABLE LOCK
  - examine lock type chosen by Optimizer via EXPLAIN
  - review Lock List Limit
  - do insert/update/delete at end of UOW
  - consider forcing U Locks on data
    - cursor updates - FOR UPDATE OF on cursor

- cursor deletes - FOR UPDATE OF dummy non-index column
- include retry logic in application
- use FETCH to hold row lock
- use SELECT to read data and free row

- Transaction management
  - UOW not clearly defined
  - excessive/insufficient COMMITS - frequent commits enhance concurrency, requires more CPU shorten recovery.
  - holding a lock while waiting for a user to respond can result to excessive lock time
  - unneccessary CONNECT statement is also expensive.
- Data handling
  - every column and row touched increses CPU and elapsed time
  - reduce number of rows searched and returned - selective predicates
  - reduce number of columns retrieved and updated - SELECT * only if required
  - REORG so clustering improves

Optimizer

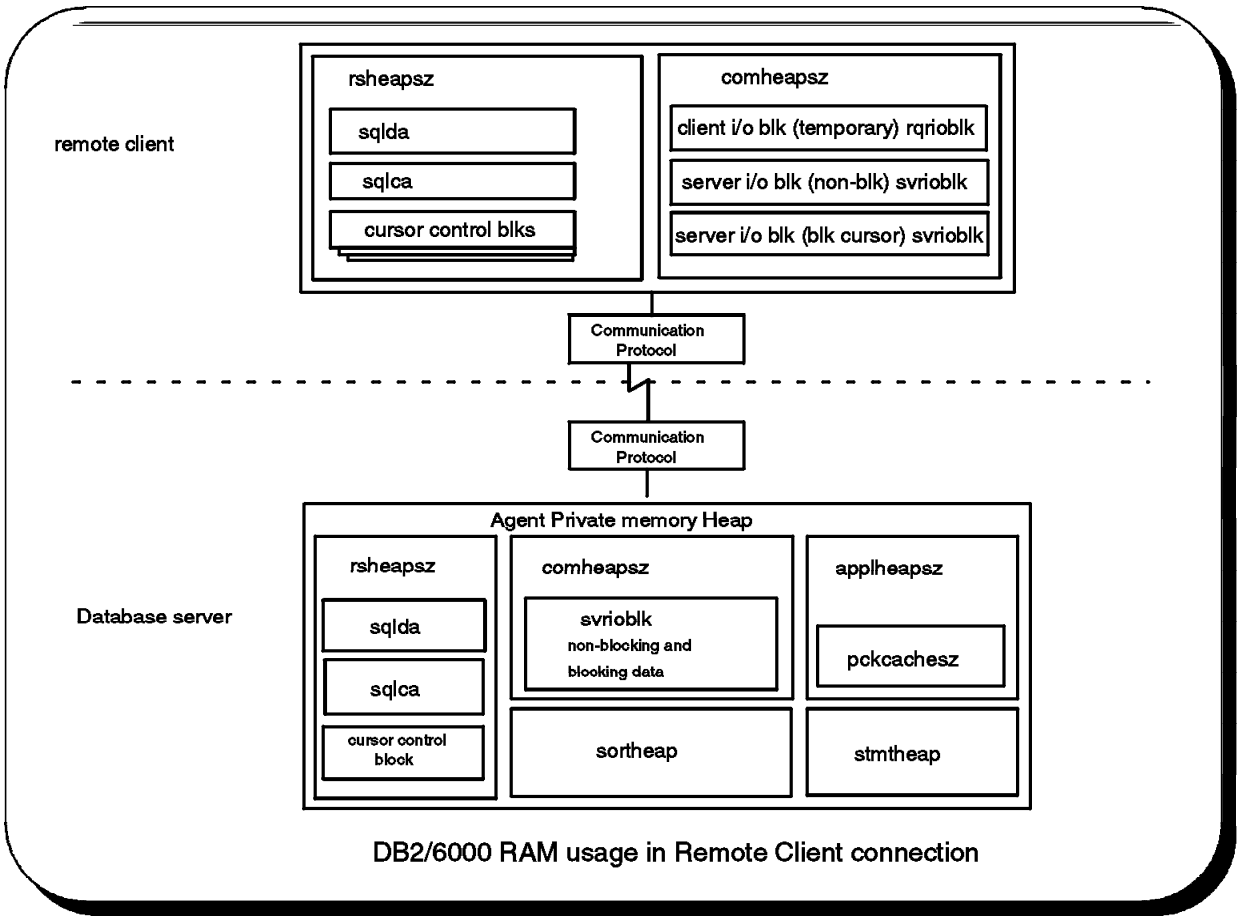
- considers the following in join method (Nested loop, merge), order (inner vs outer table) and access path
  - buffer pools size (rebind if size changes)
  - indices - 1st key and secondary key column statistics
  - restrictiveness of predicates
  - table size
  - data ordering (GROUP BY, ORDER BY, DISTINCT)



- use EXPLAIN to view decisions
  - watch for join method, access path
  - use JOINS instead of sub-SELECTS where possible

- when coding SQL, the optimizer works better if you use
  - EXIST instead of IN
  - BETWEEN instead of LIKE
  - JOINs instead of subqueries
  - try not to use NOT

- in DB2/6000, static SQL packages can be cache only if they are granted PUBLIC access. Use the PCKCACHESZ database configuration to determine size.

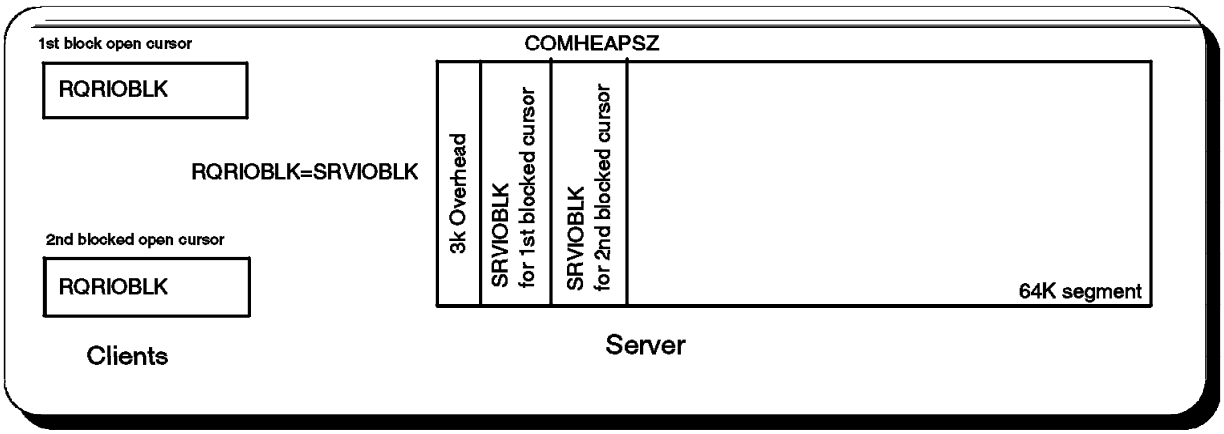


Client/Server Database

- cursor record blocking
  - avoid CURSOR Update/Delete - prevents target cursor from being blocked. Use a cursor and a separate update statement.
  - avoid ambiguous cursors - e.g. EXECUTE IMMEDIATE that issue and UPDATE against a static cursor. Cursor is ambiguous because bind process cannot predict type of statement in EXEC IMM
  - use FOR FETCH ONLY or FOR UPDATE clause on cursors
  - the /K=ALL bind parm
  - use DARI. Can reduce network messages. Benefit to the applications
    - that process large amounts of data between the remote procedure and the database but only small answer set.
    - there are logical sets of SQL which are always executed together.

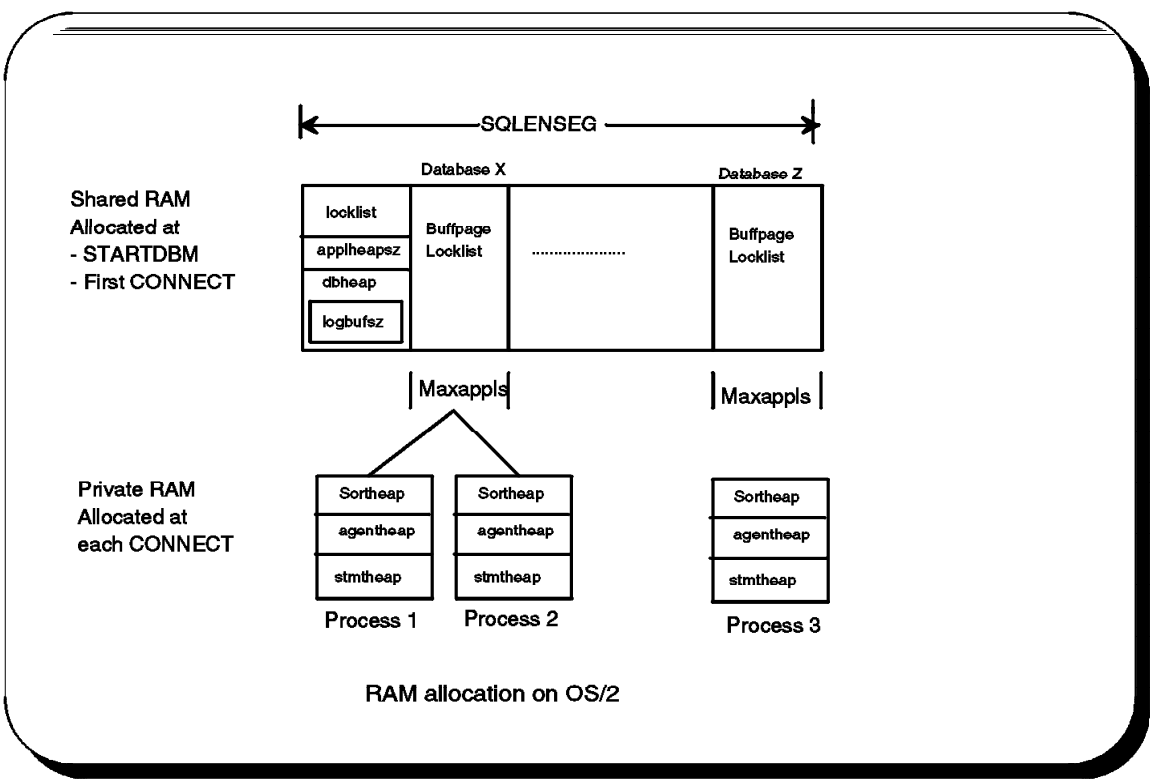
System Tuning

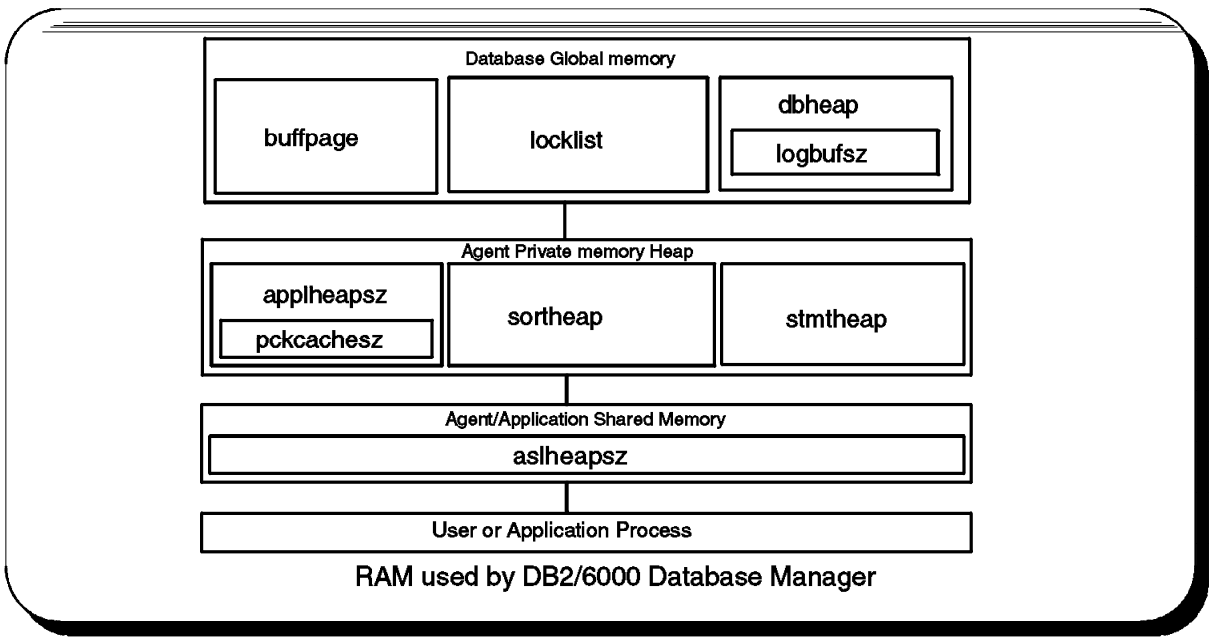
- DB2 has two main tuning paramaters
  - Database Manager paramaters - affect the amount of system resources affecting all databases and applications using DB2
  - Database paramaters - specific to one database. Paramaters specify the amount of resource allocated to that database.



- DOS/Windows client paramaters - stored in DBDRQLIB.CFG. Relates to RDS environment such as NETBIOS name and size of communication area.

- Shared and private RAM - shared RAM: available for all processes connected to database; private RAM: available for each process





- SQLNETB: *in config.sys*
  defines the number of sessions that DB2/2 will request from NETBIOS at STARTDBM time. It represents the number of concurrent sessions, plus DB2/2 overhead.
  SQLNETB = 6 + inbound + outbound sessions
- BUFFPAGE: one pools per database
  - sets the number of 4K pages of Buffer Pool
  - data read/write on disk are transferred to/from this buffer
  - data stays in the buffer for next request causing less i/o
  - size affects access plan
  - if buffer pool is too big, some data may be paged out to swapper
  - need to consider space in pages most frequently accessed tables and indexes occupy
  - pages are written back to disk when one of the following occurs:
    - all process disconnect from the database
    - the database quiesces (i.e. all connected processes have committed)
    - room is needed for another incoming page
  - Warning: If you notice that commits take a very long time to complete and there is very high disk activity, the dbm is flushing the buffer pool and the time taken is dependent on the size of the BUFFPAGE i.e. the larger it is, the longer the time it appears to "hang". This phenomena has been observed in one situation and reportedly a combination of log file being full (frequency of flushing), buffersize and commits taking place.
- SORTHEAP: one per process connected to database
  - allocated as needed and freed when sorting completes
  - used during index creation
  - size should not exceed size specified in SHEAPTHRES in database manager configuration.
  - recommendation: 2.5 x (max table to be sorted) / 64
  - if SHEAPTHRES is exceeded, any subsequent sorts will sort to temporary files/have reduced maximum sort heap size.
- DBHEAP - one per database
  - primarily contains cursor control block information. Size id dependent on number of simultaneous open cursors. Increase if application receives SQL code -956.

- APPLHEAPSZ - one per process
  - working area for process. Increase if SQL code -954.
- MAXAPPLS - maximum number of concurrent processes that can be connected (locally and remotely) to a database
- LOCKLIST - one per database
  - sets number of pages for keeping locks
  - each locks requires 25 bytes of RAM