A DB2 Performance Tuning Roadmap:
A High-Level View on Managing the Performance of DB2 for z/OS

Craig S. Mullins
Mullins Consulting, Inc.
15 Coventry Court
Sugar Land, TX 77479

http://www.craigmullins.com

1. The Tuning Progression

Problem Resolution

- Application
  - SQL
  - Host Language Code
- Database
  - Indexes
  - Database and Index Organization
  - Database Design (normalization / denormalization)
- DB2 Subsystem
  - ZPARMs, Pools, Locking, IRLM, DDF, etc.
- Environment
  - Network
  - TP Monitor (CICS, IMS/TM)
  - Operating System
2. Basic Tuning Rules

- 80% of the results of tuning come from 20% of the tuning effort -and-
  - 20% of your DB2 applications cause 80% of your problems
- Tune one thing at a time
  - How else do you know whether the action helped or not?
- All tuning optimizes:
  - CPU, I/O or concurrency
Application Code and SQL

- Most relational tuning experts agree that the majority of performance problems with applications that access a relational database are caused by poorly coded programs or improperly coded SQL…
  - as high as 70% to 80%
3. Application Tuning: SQL

- Simpler is better, but complex SQL can be efficient
- In general, let SQL do the work, not the program
- Retrieve the absolute minimum # of rows required
- Retrieve only those columns required - never more
- Always provide join predicates (i.e. no Cartesian products)
- Favor Stage 1 and Indexable predicates
  - Host variable data type/length should match column
- Avoid tablespace scans for large tables (usually)
- Avoid sorting when possible:
  - indexes for ORDER BY and GROUP BY
  - judicious use of DISTINCT
  - UNION ALL versus UNION (if possible)
3. Application Tuning: Stage 1 and 2

Stage 2 - Evaluated after data retrieval (non-sargable) via the RDS (Relational Data Services) which is more expensive than the Data Manager.

Stage 1 - Evaluated at the time the data rows are retrieved (sargable). There is a performance advantage to using Stage 1 predicates because fewer rows are passed to Stage 2 via the Data Manager.
Avoid Black Boxes
4. Application Tuning: Optimization

- SQL Statement(s)
- Other System Information
- DB2 Catalog
- Optimization Hint

DB2 Subsystem

DB2 Optimizer

Plan Tables
- PLAN_TABLE
- DSN_STATEMNT_TABLE
- DSN_FUNCTION_TABLE

DBAS (DSNDBM1)

Optimized Access Path(s)

Query Cost Formulas
4. Application Tuning: *EXPLAIN* Analysis

- Hint used?
- Index used?
  - Single, Multiple
- Matching column(s)?
- Index only?
- TS scan (*page range*)
- Type of Join?
  - Nested Loop
  - Merge Scan
  - Hybrid
- SQL Text
- Table & Index Information
  - DDL
  - Stats
- Cardinality
- Other Stuff
  - Triggers
  - RI
  - Constraints
- Prefetch?
  - Sequential
  - List
- Parallelism used?
  - I/O, CPU, Sysplex
  - Degree
- Sort required?
  - Join, Unique, Group By, Order By
- Locking

*SQL Text*  
*SQL Analysis*  
*Catalog Stats*  
*PLAN_TABLE*  
*Analysis & Recommendations*
5. Application Tuning: Locking

- Avoid deadlocks by coding updates in the same sequence regardless of program.
- Issue data modification SQL statements as close to the end of the UOW as possible.
  - The later in the UOW the update occurs, the shorter the duration of the lock.
- Encourage Lock Avoidance.
  - ISOLATION(CS) / CURRENTDATA(NO).
  - Can be used only by read only cursors.
- Use LOCK TABLE judiciously.
- Consider ISOLATION(UR) to avoid locking.
6. Application Tuning: Commit

- Avoid Bachelor Programming Syndrome

- Plan and implement a COMMIT strategy
  - or experience TIMEOUTs and DEADLOCKs
7. Application Tuning: Program

- Do not embed efficient SQL in inefficient program logic
  - **Classic example**: finely tuned, efficient SQL inside of a program loop that executes 3,510,627 times!
- Let SQL do the work when possible
  - e.g.) do not code “program” joins
- Sign of trouble: SQL followed by lots of IF...ELSE or CASE statements
- If you are only going to retrieve one row, consider coding a singleton SELECT (**usually**)
8. Application Tuning: 
*Online vs. Batch*

- When designing online transactions, limit the amount of data to be retrieved to a reasonable amount
  - No one reads hundreds of pages/screens online!
- Limit online sorting and joining (*but be reasonable*)
- Consider OPTIMIZE FOR 1 ROW to disable list prefetch
  - With LP DB2 acquires a list of RIDs from a matching index, sorts the RIDs, & accesses data by the RID list
  - Can be very inefficient for a multiple page transaction
Database Object Tuning

DATABASE

- Segmented Tablespace
- Simple Tablespace
- Partitioned Tablespace

- Table
- Table
- Table

- Table w/LOB
- Table
- Table

- LOB Tablespace
- Auxiliary (LOB) Table

- Index
- Index
- Index

- Separate LOB Tablespace per Partition (for each LOB column)

DB2 Performance Tuning Roadmap
© 2005, Craig S. Mullins (Mullins Consulting, Inc.)
9. Database Organization

- Be sure to run RUNSTATS
  - as data volume changes, new data structures added
  - followed by (RE)BIND with /EXPLAIN(YES)
- Review statistics to determine when to REORG
  - NEARINDREF and FARINDREF
  - LEAFDIST, PERCDROP
  - For clustering indexes
    - NEAROFFPOSF and FAROFFPOSF
    - CLUSTERRATIOF
  - Analyze access patterns before reorganizing
    - Random vs. sequential
    - Consider automation
10. Database Design: The Basics

- As normalized as possible, but performance before aesthetics; normalization optimizes “update” at the expense of “retrieval”
  - Don’t let data modelers dictate “physical” design
- One table per tablespace \textit{(usually)}
- Partitioned or segmented over simple TS
  - DSSIZE < 4GB unless you definitely need large TS
- Do not create base table views
- Avoid the defaults - they are usually wrong
- Determine amount of free space \textit{(PCTFREE & FREEPAGE)}
  - Based on volatility – don’t just let everything default to 10.
11. Database Design: Indexes

- A proper indexing strategy can be the #1 factor to ensure optimal performance
- First take care of unique & PK constraints
- Then for foreign keys *(usually)*
- Heavily used queries - predicates
- Overloading of columns for IXO
- Index to avoid sorting
  - ORDER BY, GROUP BY
- Consider I/U/D implications
- Sequence columns wisely in multi-col indexes
- Avoid indexing variable columns
12. Database Design: Data Types

- Use NULL sparingly
- Use appropriate DB2 data types
  - Use DATE instead of CHAR or numeric for dates
  - Store numeric data using a numeric data type
    - INTEGER, SMALLINT, DECIMAL, etc.
  - INTEGER versus DECIMAL(x,0)
    - Control over domain vs. storage requirements
  - “DATE and TIME” versus TIMESTAMP
    - Ease of use/storage vs. precision/arithmetic
- Compression versus VARCHAR
  - Compression = less overhead (no 2 byte prefix)
  - Compression requires no programmatic handling
13. Database Design: Integrity

- Use DB2 declarative RI instead of program RI (usually)
  - performance and ease of use
  - ensure integrity for planned and ad hoc database modification
- Do not use RI for lookup tables (overkill)
  - consider CHECK constraints vs. lookup tables
- Use triggers only when declarative RI is not workable
  - Triggers are less efficient (usually) than RI
    - but usually better than enforcing in application programs
- Specify indexes on foreign keys
14. Database Design: BP, part one

- Bufferpool allocations - do not default everything to BP0
  - Explicitly specify a buffer pool for every TS and IX
- Ideas:
  - isolate the catalog in BP0
  - separate indexes from table spaces
  - isolate heavily hit data
  - isolate sort work area
  - optimize BP strategy for your data & app processing mix: sequential vs. random
  - there is no “silver bullet” approach
    - more on bufferpool tuning coming up!
15. Database Design: Rows & Columns

- Avoid wasted space *(page size?)*
  - Row length > 4056 requires larger page size
  - Row length 2029 - 4056 = one row per page
  - Row length < 15 wastes space (max 255 rows/page)

- Sequence columns based on logging
  - Infrequently updated non-variable columns first
  - Static (infrequently updated) variable columns
  - Frequently updated columns last
  - Frequently modified together, place next to each other
System & DB2 Subsystem Tuning

Allied Agent Address Spaces
- CICS
- IMS
- TSO

SSAS
System Services Address Space
Logging
Recovery
Access to TP

DBAS
Database Services Address Space
RDS - Optimizer
Data Manager
Buffer Manager

IRLM
Internal Resource Lock Manager
Locking

SPAS
Stored Procedure Address Space

VSAM Media Manager

DDF
Distributed Data Facility

Data

DB2 Performance Tuning Roadmap
© 2005, Craig S. Mullins (Mullins Consulting, Inc.)

- DB2 uses four types of pools; memory structures to cache data and information to avoid costly disk I/O

  1. **Buffer Pools** - used to cache data in memory when it is read from disk.

  2. **RID Pool** - used to sort RIDs (record identifiers) for List Prefetch, Multiple Index Access, and Hybrid Joins.

  3. **EDM Pool** - used to cache program details (access paths, dynamic PREPARE, authorization) and database structural information (DBD).

  4. **Sort Pool** - used when DB2 must sort data.
16. Subsystem: Buffer Pools, part two

- DB2 provides up to 80 buffer pools - USE THEM!
  - 4K: BP0 thru BP49
  - 8K: BP8K0 thru BP8K9
  - 16K: BP16K0 thru BP16K9
  - 32K: BP32K thru BP32K9

- Consider reserving a bufferpool for tuning
  - Move problem objects there to isolate for tuning

- DB2 V8 significantly increase buffer pool storage
  - 1TB new limit for buffer pools
  - No more hiperpools
  - No more bufferpools in data spaces

- Monitor hit ratio: % times a page is found in the buffer pool
  - The higher the ratio the better

\[
\frac{\text{GETPAGES} - \text{PAGES READ}}{\text{GETPAGES}} = \frac{\text{SYNC I/O} + \text{ASYNC I/O}}{
\]
16. Buffer Pools: Tune Thresholds

- Variable Thresholds

VPPSEQT: Parallel Sequential (50% of VPSEQT)

VDWQT: Vertical Deferred Write (10%)

VPXPSEQT: Assisting Parallel Sequential (0% of VPPSEQT)

DWQT: Deferred Write (50%)

VPSEQT: Sequential Steal (80%)

Immediate Write (97.5%)

Prefetch Disabled (90%)

Data Manager Critical (95%)
16. Buffer Pools: Monitor Thresholds

- Fixed Thresholds
17. Subsystem: RID Pool

- One RID pool for all processing.
- The default RID pool size is 4 MB. Can be changed.
- RID Pool is used for:
  - enforcing unique keys while updating multiple rows
  - sorting RIDs during the following operations:
    - List prefetch, including single index list prefetch
    - Access via multiple indexes
    - Hybrid Joins
### 18. Subsystem: EDM Pool

**DB2 Database Services Address Space**

<table>
<thead>
<tr>
<th></th>
<th>DBD</th>
<th>DBD</th>
<th>CT</th>
<th>SKPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKCT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DS</td>
<td>PT</td>
<td></td>
<td>CT</td>
<td>CT</td>
</tr>
<tr>
<td>CT</td>
<td>DS</td>
<td>DS</td>
<td>SKPT</td>
<td></td>
</tr>
</tbody>
</table>

**What’s in EDM Pool**
- DBDs
- SKCTs
- CTs
- SKPTs
- PTs
- Auth Cache
- Dyn SQL Prep
- Free pages

V8 breaks each out into separate “pools”

**General ROT**: shoot for 80% efficiency; (1 in 5 DBD/SKPT/SKCT needs to be loaded)
19. Subsystem: Sort Pool

- Sort Pool value is the maximum size of the sort work area allocated for each concurrent sort user.
- The default Sort Pool size is 1 MB. It also can be changed on install panel DSNTIPC.
- In general, the larger the Sort Pool, the more efficient sorting will be.
- Need to understand the way DB2 sorts to understand tuning.
19. DB2 Tournament Sort

- DB2 uses tournament sorting...
  - rows to be sorted are passed through a tree structure, entering at the bottom; each row is compared to rows already in the tree and the lowest (ASC) or highest (DESC) are moved up the tree
  - when a row emerges from the top it is placed in an order set of rows in memory (Sort Pool)
  - when a row does not fit in the current ordered set (out of range) the complete ordered set (called a run) is moved to a work file
  - logical work files are in memory; if too big, moved to DSNDB07
  - all runs are merged to form a sorted results set
20. Subsystem: Logging

- DB2 will only run as fast as the log

Log Tuning Parameters
- Input Buffer Size
- Output Buffer Size
- Write Threshold

Waits occur if OUTBUFF is too small
At the Write Threshold, data is written to the Active Log

Log Configuration
- Dual Active Logging is the preferred configuration
- Preferably with each log defined to separate devices and on separate channels

DB2 rollbacks from log data on DASD
- Consider keeping archive logs on DASD
- Migrate archive logs to tape after a specified period of time (HSM)
20. Subsystem: System Checkpoint

- Periodically DB2 takes a checkpoint, containing:
  - currently open unit of recoveries (UR) within DB2, all open page sets, a list of page sets with exception states, and a list of page sets updated by any currently open UR
- Specified in the CHKFREQ parameter in DSNZPARM
  - Number of log records written
  - Or, as of V7, number of minutes
- Can be changed dynamically using:
  - SET LOG or (temporary)
  - SET SYSPARM (V7) (permanent)

15 minute intervals for checkpoints during peak processing times. CHKFREQ replaced LOGLOAD in V7
21. Subsystem Tuning: IRLM

- **MAXCSA** = \(250 \times (\text{LOCKS PER USER}) \times (\text{MAX USERS})\).
  - 250 bytes of storage for each lock.
- **ITRACE=NO**
  - do not use ITRACE; instead, use DB2 lock traces
- **PC=NO vs. PC=YES**
  - YES: locks held in the IRLM private address space; DB2 uses cross memory for IRLM requests
  - NO: locks held in the extended common storage area
- **DEADLOK**
  1. The number of seconds between two successive scans for a local deadlock
  2. The number of local scans that occur before a scan for global deadlock starts

DB2 V8 eliminates PC parameter
22. Environment

- Operating System
  - version, memory, JCL, RACF, etc.
- TP Monitors
  - CICS, IMS/TM, C/S GUI, web, etc.
- Networking
  - TCP/IP, SNA, DRDA, stored procedures, etc.
- DASD
  - storage, ESS/Shark, placement, etc.
Summary

Application
Database
DB2 Subsystem
Environment
Do one thing at a time and;
You can tune DB2!
Announcing DB2PORTAL.com

- New, free web portal for DB2 DBA, programmers, & users
  - Useful articles and links to DB2 information
  - The only site devoted completely to mainframe DB2

- [http://www.DB2portal.com](http://www.DB2portal.com)