



Performance Tuning Recommendations When Migrating to DB2 for z/OS V8

Dr. Jim Teng

IBM Distinguished Engineer



DB2 for z/OS Technical Forum
July 23-25, 2007
Taipei, Taiwan

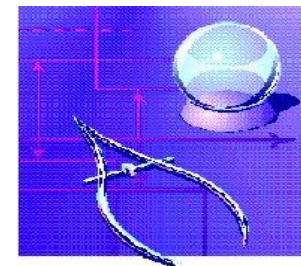
Disclaimers & Trademarks*

Information in this presentation about IBM's future plans reflect current thinking and is subject to change at IBM's business discretion. You should not rely on such information to make business plans. Any discussion of OEM products is based upon information which has been publicly available and is subject to change. The opinions expressed are those of the presenter at the time, not necessarily the current opinion and certainly not that of the company.

The following terms are trademarks or registered trademarks of the IBM Corporation in the United States and/or other countries: AIX, AS/400, DATABASE 2, DB2*, Enterprise Storage Server, ESCON*, IBM, iSeries, Lotus, NOTES, OS/400, pSeries, RISC, WebSphere, xSeries, z/Architecture, z/OS, zSeries, System p, System I, System z

The following terms are trademarks or registered trademarks of the Microsoft Corporation in the United States and/or other countries: MICROSOFT, WINDOWS, ODBC

For more copyright & trademark information see ibm.com/legal/copytrade.phtml



Notes

- **Abstract:** This presentation covers the performance considerations, such as catalog migration time and a potential change in the use of CPU time, I/O time, and virtual/real storage as a DB2 for z/OS subsystem is migrated to V8 compatibility mode and then to new function mode. It also provides performance monitoring and tuning tips on CPU and virtual storage usage of applications migrating to V8.



Agenda

- **Catalog migration performance**
- **Change in CPU usage and reporting**
- **Change in DBM1 address space virtual storage usage**
- **Change in real storage usage**
- **Performance monitoring and tuning tips**



Catalog Migration

- **V7 to V8 Compatibility Mode (CM)**
 - 0.2 to 10 minutes, depending on the size of catalog/directory (medium to large)
 - Time heavily depends on DASD and channel model used also
 - No significant change in catalog size from V7 to V8 Compatibility Mode
 - Compatibility Mode
 - DB2 catalog in EBCDIC
 - Fallback to V7 allowed
 - Check for type 1 index
 - If any found, catalog migration rolled back



Catalog Migration ...

- **V8 Compatibility Mode to New Function Mode (NFM)**
 - **0.1 to 2 hours depending on the size of catalog/directory (medium to large)**
 - **1 to 10% increase in the size of catalog for both data and index**
 - **New Function Mode**
 - **DB2 catalog in unicode**
 - **No fallback allowed**



Catalog Migration ...

- Online Reorg Sharelevel Reference of SPT01 and 17 catalog tablespaces the most time-consuming
- Very Rough Rule-of-Thumb on estimating the time for medium to large catalog/directory = 6min + 5min/GB of SPT01, SYSPKAGE, SYSDBASE, etc.
 - Example: If 10GB SPT01, SYSPKAGE, SYSDBASE, then $(6+5 \times 10) =$ roughly 1 hour
 - Most catalogs are smaller than 10GB and thus faster
- Time also depends on DASD and channel model used
- Catalog reorg can make catalog migration faster, especially if no reorg for a long time



Major Performance Highlight of V8

- **10 to 100 times improvement possible from**
 - **Materialized Query Table**
 - **Stage 1 and indexable predicate for unlike data types**
 - **Distribution statistics on non-indexed columns**
 - **Other access path selection enhancements**
- **2 to 5 times improvement possible from**
 - **Multi-row Fetch, cursor Update/Delete, Insert**
 - **Star join with work file index, in-memory work file, more parallelism**
 - **DBM1 virtual storage constraint relief**
 - **Partition Load/Reorg/Rebuild with DPSI**



For applications not taking advantage of V8 performance features

- **I/O time**
 - No change for 4K page I/O
 - Significant sequential I/O time improvement possible for 8K, 16K, or 32K page because of bigger Vsam Control Interval size (NFM)
 - Up to 70% i/o data rate (MB/sec) improvement
 - Also Vsam i/o striping now supported
- **Some CPU time increase is expected in order to support a dramatic improvement in user productivity, availability, scalability, portability, family consistency,..**
 - DBM1 virtual storage constraint relief with 64bit instructions
 - Long names, long index keys
 - Longer and more complex SQL statements



CPU change based on laboratory measurements

with no application nor aggressive configuration/environment change

- **'+' means cpu increase, '-' means reduction, compared to V7**
- **-5 to +15% online transaction**
- **-10 to +10% online transaction in data sharing (NFM)**
- **-5 to +20% batch**
 - **-5 to +5% insert**
 - **-5 to +20% select**
 - **+5 to 20% fetch, update**
- **-10 to +15% batch data sharing (NFM)**
- **-20 to +15% batch DRDA**
- **-5 to +5% utility**
- **-20 to +15% query**



CPU change - continued

- ➔ Typically, no difference between CM and NFM except in data sharing for workloads with
- No application change
 - No aggressive configuration/environment change
 - Examples of what CM supports
 - Most access path selection enhancements
 - DBM1 virtual storage constraint relief
 - Lock avoidance in Select Into with CurrentData Yes, overflow rows
 - 180 CI limit removal in list prefetch and castout I/O
 - Long-term page fix option by buffer pool
 - SMF 89 performance enhancement (usage pricing)
 - Data sharing immediate write default change
 - Implicit multi-row operation in DRDA



More Indexable Predicates

- **For column comp-op value with unlike type or length**
 - ▶ **4byte char column = 8byte host variable**
 - ▶ **Integer column = decimal host variable**

 - ▶ **Stage 2 and non indexable in V7**
 - ▶ **Stage 1 and indexable in V8**
 - **So index on char or integer column here can be used in V8 but not in V7**

 - ▶ **Also useful where a programming language does not support all SQL data types. For example,**
 - **No decimal type by C/C++, no fixed-length char by Java**



Multi-row Fetch

- **FETCH NEXT ROWSET FROM cursor FOR N ROWS INTO hva1, hva2, hva3**
- **Up to 50% CPU time reduction by avoiding API (Application Program Interface) overhead for each row fetch**
 - ▶ **%improvement lower if more columns and/or fewer rows fetched per call**
 - **Higher %improvement if acctg class 2 on**



Multi-row Insert

- **INSERT INTO TABLE FOR N ROWS
VALUES(:hva1,:hva2,...)**
- **Up to 30% CPU time reduction by avoiding API overhead for each row insert**
 - ▶ **%improvement higher with fewer indexes, fewer columns, and/or more rows inserted per call**
- **Similar improvement for multi-row cursor Update and Delete**



Automatic exploitation of multi-row Fetch

- **DRDA as discussed previously**
- **DSNTEP4 = DSNTEP2 with automatic multi-row fetch**
 - ▶ **Up to 35% CPU reduction in fetching 10000 rows with 5 and 20 columns**
- **DSNTIAUL (sample Unload utility)**
 - ▶ **Up to 50% CPU reduction in fetching 10000 rows with 5 and 20 columns**



Varying Length Index Keys

- **VARCHAR index key no longer needs to be padded to maximum**
 - ▶ **V7: Always padded to maximum length**
 - ▶ **V8: Option of either padded or not**
 - ▶ **Especially useful for a large VARCHAR, e.g. DB2 catalog with 128byte VARCHARs**
 - ▶ **In such a case, more index entries per index page, fewer index pages and index levels, less DASD space and buffer pool needed**

- **Further enablement of index-only access**
 - ▶ **SELECT varchar1 FROM T WHERE char1=x**
 - **with index on char1,varchar1**



Partitioned Tablespace Without Index

- **Table-controlled versus index-controlled partitioning**
 - ▶ **V7: Partitioning index always required**
 - ▶ **V8: Partitioning index is optional**
- ▶ **Partitioning without index could reduce the number of indexes by 1**
 - **CPU and I/O reduction in Insert, Delete, and possibly Update**
 - **Especially useful if partitioning index is not referenced in predicates**

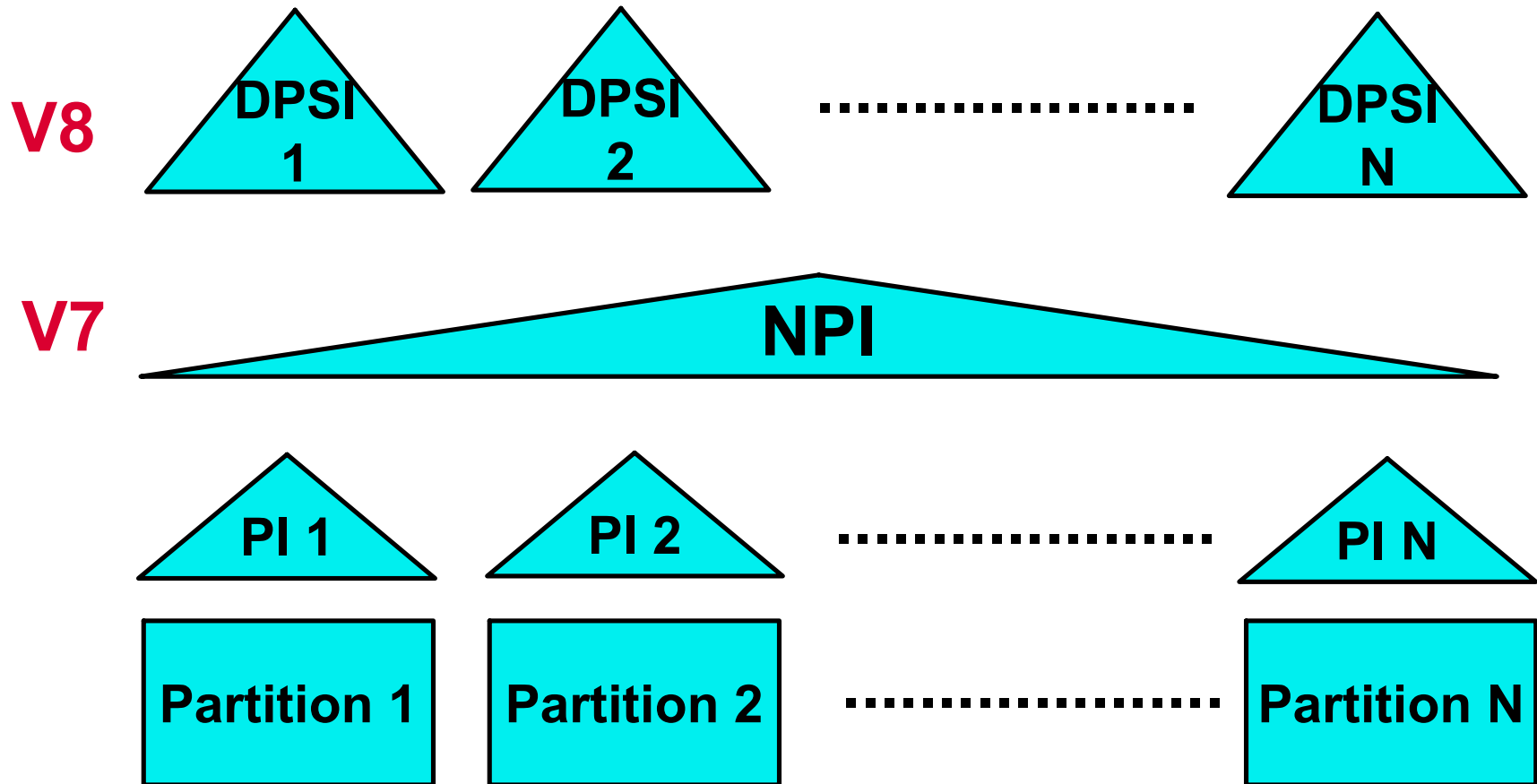


Reading Index Backward

- **Read multiple rows via index backwards to avoid sort**
 - ▶ **SELECT FROM TABLE ... ORDER BY c1 DESCENDING**
 - with an ascending index on c1
 - Dynamic prefetch of index to make backward scan as efficient as forward scan
 - ▶ **Supported with or without scrollable cursor**
 - ▶ **Accesstype = IR**



Partition Load/Reorg/Rebuild with DPSI



DPSI Usage Considerations

- **Not for unique index**

- **Query performance impact**
 - 1 **Depending on PI predicates available, some or all DPSI partitions may have to be scanned because the same DPSI key values may be in multiple partitions**
 - **More % overhead with fewer rows scanned and/or more partitions scanned**

 - 2 **ORDER BY DPSI column may require extra processing**

 - 3 **Trade-off between partition tablespace utility and some query performance**



Tuning for CPU Usage in V8

- **Rebind plans/packages**
 - **Better access path selection, especially beneficial for complex query**
 - **Enable SPROC (fast column processing) for 64bit**
 - **Reduce package overhead, especially when small number of short-running SQL calls/package**
 - **Take advantage of some ALTERed objects; for example**
 - **Matching index access or index-only after Alter Index Add Column (NFM)**
 - **Index-only after Alter padded to non padded index (NFM)**
 - **Consider Alter to less expensive column type (e.g. VARCHAR to CHAR)**



Long-term page fix option for buffer pool

- **DB2 BPs have always been strongly recommended to be backed up 100% by real storage**
 - **To avoid paging which occurs even if only one buffer short of real storage because of Least-Recently-Used buffer steal algorithm**
 - **Given 100% real storage, might as well page fix all buffers just once to avoid the cost of page fix and free for each and every i/o**
- **Up to 8% reduction in overall IRWW transaction cpu time**
- **New option: ALTER BPOOL(name) PGFIX(YES/NO)**
- **Recommended for BPs with high buffer i/o intensity = [pages read or written]/[number of buffers]**



CPU Tuning in Data Sharing

- **Group-wide shutdown and restart to reduce global and false contentions for pageset/partition locks when release commit in data sharing (NFM)**
 - **-6% overall cpu for 2way data sharing IRWW**
- **CF Batching**
- **Remove 180 VSAM Control Interval limit in list prefetch and castout i/o**
 - **2 to 3% overall CPU time reduction for IRWW**



Caution on observed CPU time increase in V8

- In general, higher %cpu increase in acctg
 - But lower %cpu increase, or even reduction, in MSTR, DBM1, and IRLM address spaces possible
 - For lower %cpu increase overall



	Non data sharing	Data sharing
Acctg class2	+3%	+11%
MSTR/ DBM1/ IRLM	-15%	-52%
Total	-1%	-11%

- **Example of IRWW**

- Also, usage pricing improvement in class1, but not class2, acctg



Notes

- **Less DBM1 SRB time from**
 - Long term page fix option especially in prefetch and write i/o
 - 180 VSAM CI limit removal in list prefetch and castout i/o
 - z/OS 1.4 CF Level 12 batching of multiple CF write and castout read requests into 1 CF access
 - Bigger benefit for Insert/Update/Delete-intensive application
- **Less MSTR SRB time from**
 - Default immediate write change from NO(Phase2) to Phase1
 - MSTR SRB shifted to Acctg TCB time – no net change
- **Less IRLM time from**
 - Reduced global and false contention for table space and partition locks when release commit in data sharing
 - Less need for release deallocate bind option



V8 Virtual Storage Usage in DBM1 Address Space

'typical' V7 below 2GB storage usage shown

EDM DBD pool 10 to 250MB

Buffer pool 0 to 1GB

[Dataspace lookaside buffer 0 to 100MB]

Buffer control blocks 1MB to 300MB

RID pool 4 to 80MB

Castout engine work area 0 to 80MB

Compression dictionary 0 to 500MB

[RDS OP pool 5 to 500MB]

Dyn Stmt Cache control blocks 0 to 200MB

BufMgr/DataMgr Trace Table 10 to 100MB

2GB

2GB

Other EDM pool 20 to 300MB

Local Dynamic Statement Cache 0 to 300MB

Thread and stack storage 50 to 800MB



Virtual storage ...

- **DBM1 virtual storage constraint relief improves scalability of performance**
 - As the processor power continues to grow, linear scalability, or ability to exploit increasing processor power without encountering a bottleneck which prevents the full CPU usage, becomes more important.
 - Bigger buffer pool and cache to reduce i/o bottleneck and CPU overhead
 - ➔ Without 64bit support, it was difficult to exploit more than 20GB of real storage
 - Up to 32GB on z800, 64GB on z900, 256GB on z990
 - CPU-Storage trade-off
- **However, thread storage is still below 2GB in V8**
 - Hence, maximum number of threads supported, such as CTHREAD (2000) and MAXDBAT (1999), is not increased.



Notes

- **Bigger DBM1 virtual storage constraint relief if V7 with bigger**
 - Buffer pool below 2GB
 - Buffer control blocks for virtual pool, hiperpool, dataspace buffer pool
 - Dataspace buffer pool lookaside buffer
 - Compression dictionary
 - Castout buffers
 - RID pool, sort pool (in RDS Op pool)
 - Data Manager/Buffer Manager trace table
 - Dynamic statement cache control block
 - IRLM PC=NO
- **Less relief from more**
 - User and system thread storage with associated stack
 - Local dynamic statement cache



Estimation of V8 Below 2GB DBM1

Use based on V7 Stats

- Average estimates
 - Thread storage: +40 to 90% (40% for system, 40 to 90% for user thread)
 - Stack storage: +100%
 - Dynamic statement cache: +60%
 - EDM pool: roughly the same (more if <40% DBD, less if >60%)
 - Trace table: -50%
 - DSC control block: -70%
 - RID pool: -90%
 - Others: -100%
- Most customers get some to good relief but a small% of customers may get small% increase in DBM1 below 2GB use



Notes

- **For a fair comparison, use the same pool size, # of threads, etc. instead of the defaults which may have changed.**
- **Bigger thread/stack storage in V8 for**
 - **Long names, keys, statements, other new functions**
 - **A portion of RDS op pool for dynamic SQL**
 - **More system agents**

➔ How much more room in DBM1 address space below 2GB depends on % of storage used for threads, stacks, and local DSC versus others



Customer 1 (Europe)	V7 measured	V8 estimated
Pool Virtual	0 MB	0 MB
Buffer control block	78	0
Dataspace lookaside buffer	5	0
Castout buffer	38	0
EDM pool	118	118
Compression dictionary	340	0
1170 system agents	73	103
25 user threads	18	35
RDS OP pool	29	0
RID pool	92	10
DSC control block	53	16
Trace table	15	7
Stack storage	49	98
TOTAL DBM1 below 2GB	908MB	387MB (-57%)



Notes

- **Negligible local dynamic statement cache**
- **No virtual buffer pool as dataspace buffer pool is used instead.**
- **Assumes**
 - **Same number of system agents in V8**
 - **90% increase in user thread storage**
- ➔ **Good DBM1 virtual storage constraint relief for this customer**
 - ➔ **Even though dataspace buffer pool was used exclusively**
 - ➔ **Because of large compression dictionary and small thread/stack storage**



Customer 2 (US)	V7 measured	V8 estimated
Pool Virtual	98 MB	0 MB
Buffer control block	13	0
Dataspace lookaside buffer	6	0
Castout buffer	17	0
EDM pool	71	71
Compression dictionary	51	0
837 system agents	64	90
493 user threads	291	553
RDS OP pool	420	0
Dynamic Statement Cache	41	66
DSC control block	42	13
Trace table	36	18
Stack storage	143	286
TOTAL DBM1 below 2GB	1293MB	1097MB(-18%)



Notes

- Negligible RID pool
 - Both virtual buffer pool and dataspace buffer pool used here
 - Large RDS OP pool due to lots of concurrent sort
 - Assumes
 - Same number of system agents in V8
 - 90% increase in user thread storage
- ➔ DBM1 virtual storage constraint relief not as good as the Customer 1 because of large thread/stack storage



Virtual Storage-Related Tuning

- **Bind option release commit vs deallocate**
 - **Commit** releases pageset/partition locks, RDS sections, and storages sooner, resulting in better concurrency and less DBM1 virtual storage usage
 - Recommended default
 - **Deallocate** holds on to these resources longer, resulting in possibly less CPU time (0 to 20%)
 - Recommended only for frequently-executed, high-volume, and performance-sensitive packages or plans
 - DB2PM/PE Acctg Report Short shows a list of pkgs/plans executed along with #occurrences and avg# SQL statements, elapsed time, and cpu time.



Notes

- **If BP size increase is considered,**
 - **Make sure there is sufficient real storage to back it up 100%.**
 - **For BP with 100,000 buffers as an example, if 99,999 real storage frames are available, then every I/O could result in paging because of Least Recently Used buffer steal algorithm.**
 - **Deferred write threshold (VDWT) may need to be reduced in order to avoid “hiccup effect” at checkpoint.**
 - **Eg 5% VDWT of much bigger BP can have many more updated pages to be written out at checkpoint.**
- **V8 deferred write threshold default change**
 - **Buffer pool threshold (DWT): 50% to 30%**
 - **Dataset threshold (VDWT): 10% to 5%**
- **If migrating from VP+HP configuration with new BP size = VP+HP size, adjust BP thresholds, which are based on VP but not HP.**



Virtual Storage-Related Tuning ...

- ➔ **If necessary, reduce MAXKEEPD to reduce local DSC**
 - **Rely more on global DSC which is above 2GB**
- **Dynamic statement cache control blocks above 2GB**
- **CONSTOR and/or MINSTOR to reduce thread storage, especially for >1MB per thread storage**



Real Storage (RS) Usage

- From V1 R1 in 1985 to the present, real storage usage growing at about 20 to 30% yearly to support performance scalability
 - More and bigger buffer pools, other pools, threads, ...
- V8 continues a similar trend
 - By removing bottlenecks which would have prevented the exploitation of bigger real storage
 - ⇒ If everything under user control is kept constant, 1 to 20% increase in real storage typically observed
 - Less %increase for larger DB2 subsystem
 - Bigger %increase for smaller DB2 subsystem



Notes

- **Without 64bit support, difficult to exploit more than 20GB of RS because of DBM1 virtual storage constraint**
 - RS of up to 32GB on z800, 64GB on z900, and 256GB on z990
- **Example of more real storage usage**
 - Higher default and maximum buffer pool size, RID pool size, sort pool size, EDM pool size, ...
 - Bigger and possibly more threads
 - Bigger modules, control blocks, internal working storage
 - More parallelism enabled
 - Parallel sort for multiple tables in composite
 - Parallel multi-column merge join



Z9 Integrated Information Processor (zIIP)

- ZIIP intended to reduce the total cost of ownership
- Prerequisites: DB2 for z/OS V8, z/OS 1.6, z9 processor
- **SYS1.PARMLIB(IEAOPTxx) PROJECTCPU=YES** for projection without zIIP
- Off-loadable enclave SRBs in 3 areas
 - DRDA over TCP/IP
 - Parallel query
 - Load, Reorg, Rebuild Utility



Reference

- V8 manuals, especially Performance Monitoring and Tuning section of Administration Guide
- Redbooks at www.redbooks.ibm.com
 - DB2 UDB for z/OS V8 Technical Review SG24-6871
 - DB2 UDB for z/OS V8 Everything you ever wanted to know... SG24-6079
 - DB2 UDB for z/OS V8 Performance Topics SG24-6465
 - A Deep Blue View of DB2 Performance: IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS SG24-7224
- More DB2 for z/OS information at www.ibm.com/software/db2zos
 - E-support (presentations and papers) at www.ibm.com/software/db2zos/support.html

