IBM Software Group

Virtual Storage Constraint Relief in DB2 for z/OS V8 ... What to Expect

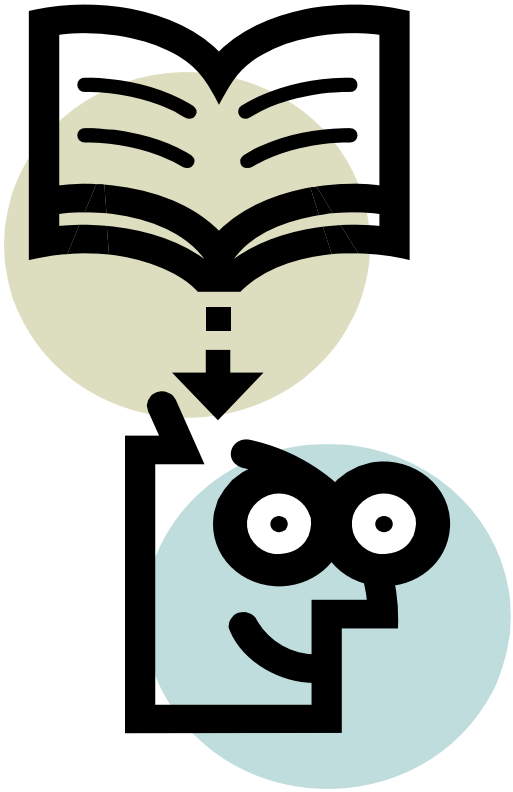
IBM Information Management software

John J. Campbell
Distinguished Engineer
DB2 for z/OS Development
Email: CampbelJ@uk.ibm.com

ON DEMAND BUSINESS

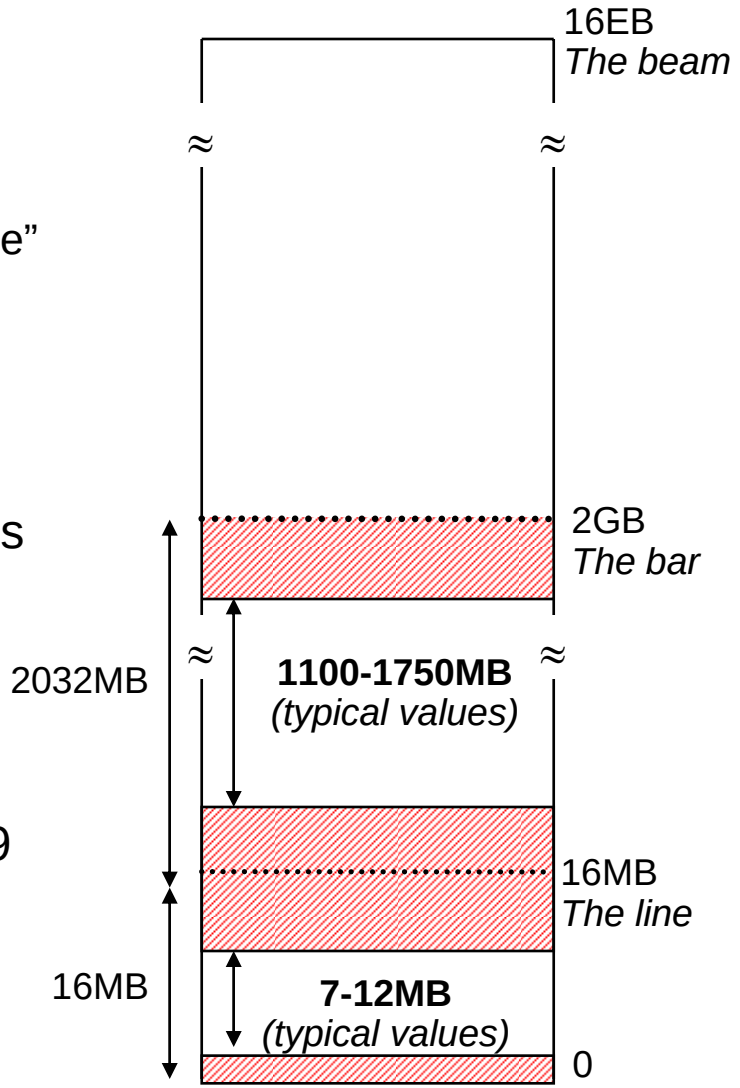
Topics

- DBM1 64-bit Virtual Memory Map
- DBM1 64-bit Virtual and Thread Storage
- Projecting V8 Use from V7 Statistics Trace
- Key Messages
- Problem Recap and Driving Factors
- Analysing Virtual Storage Used
- Tuning Options
- Protecting The System
- Real Storage Use
- Summary

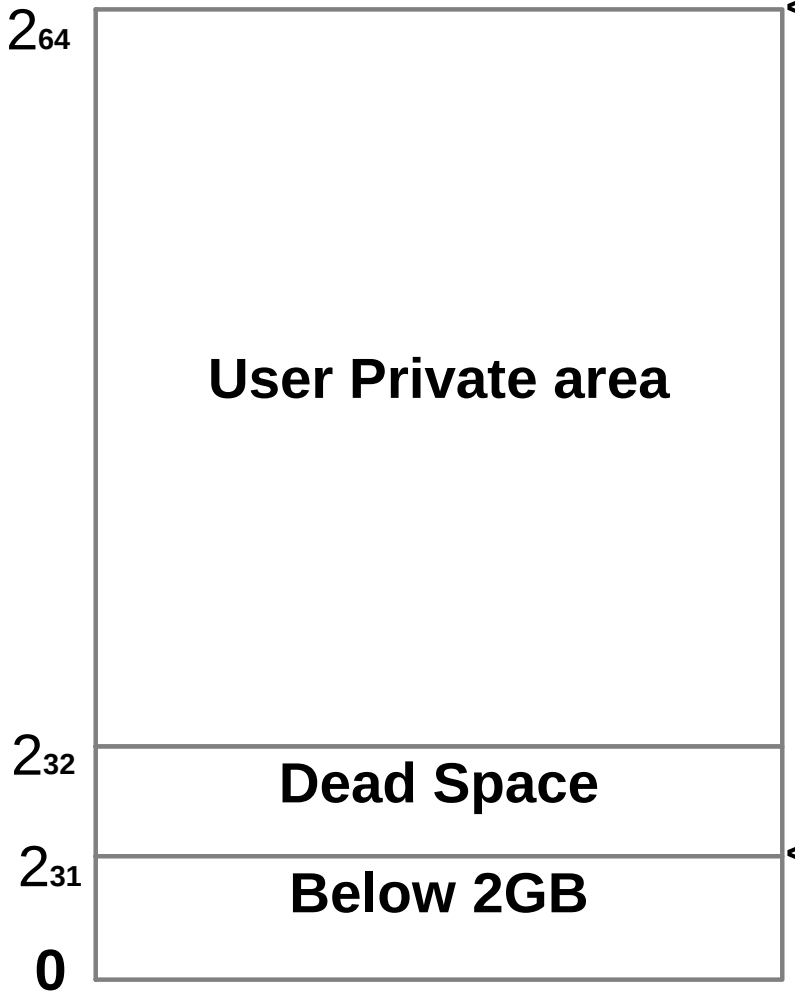


DBM1 64 bit Virtual Memory Map

- Each address space now has an addressing range of 16EB (“the beam”) based on 64 bit addressing but
  - Maximum of 16MB available “below the 16MB line”
  - Maximum of 2032MB available “above the 16MB line” and “below the 2GB bar”
- Practical maximum available to DB2 and specifically DBM1 Address Space is much less
  - Typical 7-12MB available “below the line”
  - Typical 1100-1750MB available “above the line”
- Storage is allocated into different subpools which have unique characteristics e.g., SP229
  - Storage acquired via GETMAIN
  - Storage released by FREEMAIN



DBM1 64 bit Virtual Memory Map …



<- 16EB "beam"(?)

- Buffer pools, BM control blocks
- Castout buffers
- RID pool
- EDM DBD cache (OBDs)
- Global dynamic stmt cache
- Sort pool
- Trace tables (Global, Lock, BB)
- Accounting blocks
- Compression dictionaries
- IRLM locks

<- 2GB "bar"

- Most thread storage. Local DSC
- Other EDM and RID component

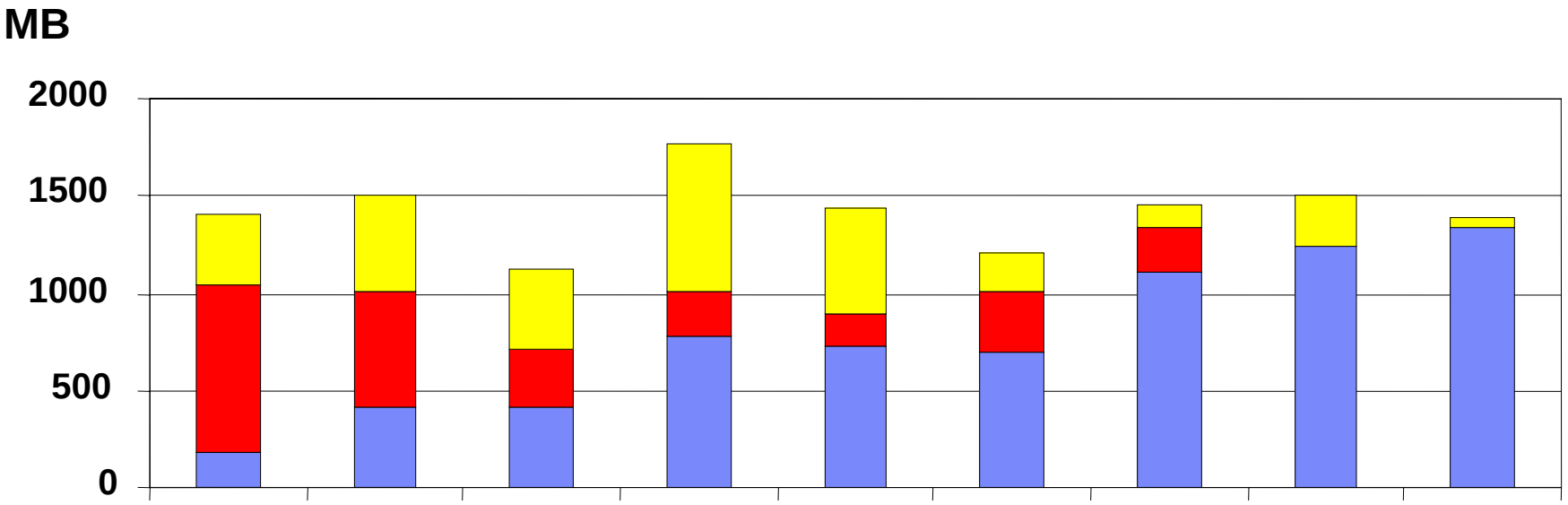
DBM1 64-bit Virtual and Thread Storage

- Most of the thread storage stayed below the 2GB with regression
  - Agent Local
  - Getmained Stack Storage
  - Local Dynamic Statement Cache
- Regression estimates
  - Agent Local Storage:
    - System: +40% for system threads
    - Non-system: +30 to 40% for static, +50 to 100% for dynamic
  - Getmained Stack storage: +100%
  - Local Dynamic Stmt Cache Cntl Blks: -75%
  - Thread Copies of Cached SQL Stmts: +100%
  - EDM pool: +30 to 50% (*)
  - RID pool: -75%
  - Others: -100%

Projecting V8 Use from V7 Statistics Trace

	V7 measured (MB)	V8 estimated (MB)		Notes:
Virtual buffer pool	15	0		0
Buffer pool control blocks	95	0		0
Dataspace lookaside buffe	48	0		0
EDM pool	88	124		+30 to 50%
Compression dictionary	54	0		0
Castout buffers	28	0		0
System thread storage	89	125		+40%
User thread storage	114	217		+30 to 40% static SQL, +50 to 100% dynamic SQL
RDS OP pool	8	0		0
RID pool	78	20		-75%
Pipe Manager subpool	1	1		Same
Local Dynamic Control Blk	70	18		-75%
Local Dynamic Stmts	8	16		+100%
BM/DM trace table	18	9		-50%
Fixed storage	3	3		Same
Stack storage	58	116		+100%
Total	775	649		
% VSCR		16		

Potential Value of V8 64-bit Virtual below 2GB bar



Top of ***Yellow*** = Extended Region Size (Max)
Top of ***Red*** = V7 use
Top of ***Blue*** =V8 use

Key Messages

- DBM1 64-bit will provide valuable VSCR for most installations
- Actual value of DBM1 will vary by installation
- But DBM1 64-bit support does not eliminate the problem …
- Installations must continue to capacity plan for, monitor, tune and optimise use of virtual storage below 2GB bar
- Must have sufficient real storage to fully back increased total virtual storage usage: below 2GB bar and above the 2GB bar
- May be able to support some additional active threads?
- May be able to set zparms CONTSTOR=NO and MINSTOR=NO

What Is The DBM1 Problem?

GETMAIN processing by DB2

- Requests may be conditional or unconditional
- "Short on Storage" condition can occur for both
- DB2 recovery routines may be able to clean up
- Individual DB2 threads (allied, DBAT) may abend with 04E/RC=00E200xx when insufficient storage available
  - e.g., 00E20003 & 00E20016
- Eventually DB2 subsystem may abend with abend S878 or S80A when critical task and no toleration of error

What Are The Drivers?

- Workload growth, both organic and through mergers & acquisitions
- Shrinking maximum region size (EPVT) available to DB2
  - Conflict between using ECSA (IMS) and EPVT (DB2)
  - Extensive use of ECSA by IMS across dependent regions
    - Mostly buffer pools, control blocks, data are in ECSA
    - Sizes are at user choice
    - For best performance they tend to be large
    - Not exploiting VSCR features of recent IMS releases
- LPAR consolidation
- Other use of extended common areas e.g., WebSphere
- Generous over allocation for safety of ECSA and other extended common areas
- Common LPAR image for Sysplex (best practice)

What Are The Drivers? ...

- Increase in average thread footprint across successive DB2 releases
- Long running persistent threads (IMS WFI, CICS Protected Entry Threads, WebSphere Connection Pool, DDF Connection Pool)
- Plans/packages with RELEASE(DEALLOCATE)
- Use of Local (thread) Dynamic Statement Caching (BIND option KEEPDYNAMIC YES and zparm MAXKEEPD > 0) to reduce CPU consumption
  - WebSphere with JDBC to avoid short prepares
  - ERP & CRM applications e.g., SAP
- CTHREAD and MAXDBAT throttles set to high values that cannot be supported when system slowdown occurs, workload keeps arriving and more threads come into play
- Degradation in DB2 Activity Time (Acctg Class 2 Elapsed)
  - Degraded IO response and CF service times
  - Application lockouts

Deadly Combination

- Maximum use of Dataspace Bufferpool under V7
- Many concurrent persistent threads
- KEEPDYNAMIC(YES)
- RELEASE(DEALLOCATE)
- CTHREAD and/or MAXDBAT throttles wide open
- V6->(V7)->V8

Thread Footprint

- Low end 200 to 400KB
  - Simple Static SQL
- Mid range 500 KB to 2MB
  - Most Static / Simple Dynamic SQL
- High end 3MB to 10MB plus
  - Complex Dynamic SQL, Heavy Sort, Parallelism
- Mileage will vary by installation

Analysing Virtual Storage Used

- RMF for very high level view
  - Virtual Storage (VSTOR) Private Area Report
    - Interval data collected in SMF Type 78-2
    - Collected by RMF Monitor I session option: VSTOR(D,xxxxDBM1)
    - Produced by RMF Post Processor option: REPORTS(VSTOR(D,xxxxDBM1))
  - Use to identify potential storage shortages and to get historical view of virtual storage consumption
  - Calculate amount of storage available above the line by subtracting MAX LSQA/SWA/229/230 PAGES ALLOCATED and MAX USER REGION PAGES ALLOCATED from REGION ASSIGNED
  - How much is enough?
    - Greater than 500MB spare is AOK (GREEN)
    - Between 200-500MB spare is boundary condition (AMBER)
    - Less than 200MB action is required (RED)

V I R T U A L S T O R A G E A C T I V I T Y

z/OS V1R5 SYSTEM ID X9 DATE 05/25/2005 INTERVAL 10.00.68
RPT VERSION V1R5 RMF TIME 15.00.00 CYCLE 0.250 SECONDS

PRIVATE AREA SUMMARY

JOB NAME - DBL1DBM1
STEP NAME - DBL1DBM1
PROGRAM NAME - DSNYASCP
NUMBER OF SAMPLES - 240

REGION REQUESTED 0K
REGION ASSIGNED (BELOW 16M) 11.0M
REGION ASSIGNED (ABOVE 16M) 1635M

PRIVATE STORAGE MAP

```
                    BELOW 16M                          EXTENDED (ABOVE 16M)
AFFFFF ____________________________                ____________________________ 7FFFFFF
       | LSQA/SWA                 |                | LSQA/SWA                 |
       | 229/230           328K   |    BOTTOM OF   | 229/230          1089M   |
AAE000 |_____15.00.00_____________| ALLOCATED AREA |_____15.05.32_____________| 3BEF700
       | UNUSED                0K |                | UNUSED                0K |
B00000 |____________________________|  GETMAIN LIMIT |____________________________| 7FFFFFF
       | UNUSED            10.4M   |                | UNUSED             521M   |
 3F000 |_____15.00.00_____________|     TOP OF     |_____15.00.00_____________| 1B58F00
       | USER                     | ALLOCATED AREA | USER                     |
       | REGION             228K  |                | REGION                   |
  6000 |____________________________|                |                   24.6M  |
       | SYSTEM REGION       16K  |                |                          |
  2000 ----------------------------                ---------------------------- 19D0000
```

	BELOW 16M MIN	BELOW 16M MAX	BELOW 16M AVG	ABOVE 16M MIN	**ABOVE 16M MAX**	ABOVE 16M AVG
LSQA/SWA/229/230						
FREE PAGES (BYTES)	40K 15.00.00	40K 15.00.00	40K	28K 15.02.37	724K 15.00.00	210K
LARGEST FREE BLOCK	36K 15.00.00	36K 15.00.00	36K	4K 15.02.37	320K 15.00.00	86K
PAGES ALLOCATED						
(IN BYTES)	288K 15.00.00	288K 15.00.00	288K	92.8M 15.00.00	**1089M** 15.05.32	828M
USER REGION						
FREE PAGES (BYTES)	10.4M 15.00.00	10.4M 15.00.00	10.4M	521M 15.05.32	1517M 15.00.00	783M
LARGEST FREE BLOCK IN GETMAIN LIMIT	10.4M 15.00.00	10.4M 15.00.00	10.4M	521M 15.05.32	1517M 15.00.00	783M
PAGES ALLOCATED						
(IN BYTES)	228K 15.00.00	228K 15.00.00	228K	24.6M 15.00.00	**24.6M** 15.00.00	24.6M

1635 – (1089 + 24.6) = 521.4MB available

Analysing Virtual Storage Used …

What consumes the virtual storage used by DBM1 address space?

- DB2 instrumentation for detail
  - IFCID 225
    - Summary Information
    - Snapshot as each DB2 Statistics interval becomes due
    - Available through DB2 Statistics Trace Class 1
    - Tiny overhead in terms of increased CPU resource consumption and increased SMF data volume
    - Start automatically via zparm SMFSTAT(1,…)
    - Recommend zparms STATIME=5 and SYNCVAL=0
  - IFCID 217
    - Detail Information at thread level
    - Available through Global Trace Class 10
  - For description of IFCIDs see DSN810.SDSNIVPD(DSNWMSGS)

Analysing Virtual Storage Used …

What consumes the virtual storage used by DBM1 address space? ...

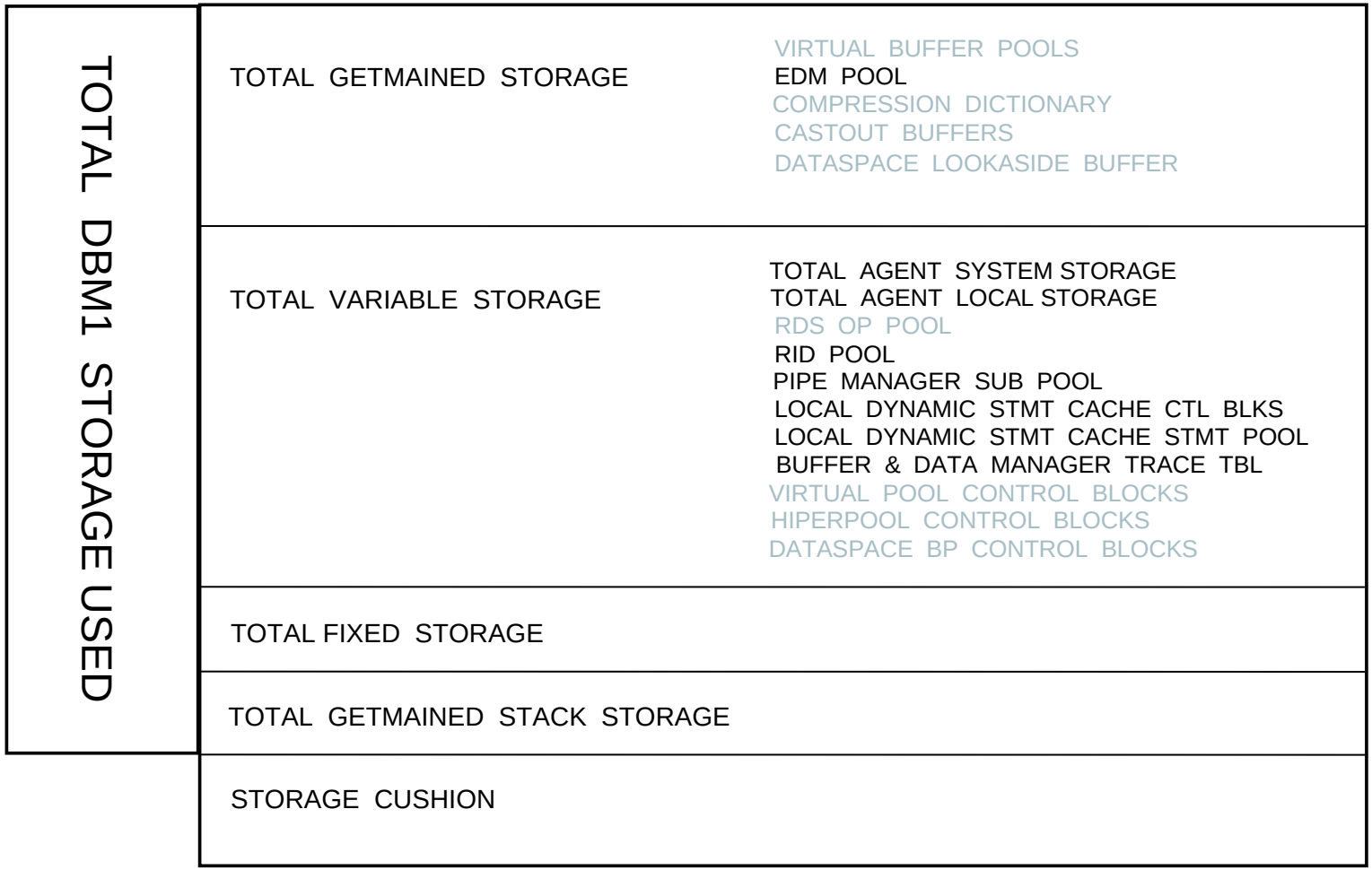
- First class support provided by OMEGAMON XE for DB2 PM/PE, DB2 PM and DB2 PE
  - Statistics Trace | Report
    - Includes FILE and LOAD data base table support as well as upgrade (ALTER TABLE ....) of already installed table DB2PM_STAT_GENERAL
  - Record Trace Report
    - IFCID 217 and 225 supported independent of DB2 release which created records
  - New SPREADSHEETDD subcommand option
    - Both DB2PE V2.1 & DB2PM V8.1 via APAR PK31073
    - OMEGAMON XE for DB2 PE V3 & V4 via APARs PK33395 & PK33406
- REXX Tools (MEMU2, MEMUSAGE) with User Guide
  - Available for download from DB2 Trading Post off DB2 for z/OS Home Page

Sample – Statistics Trace | Report

DBM1 AND MVS STORAGE BELOW 2 GB		QUANTITY
TOTAL DBM1 STORAGE BELOW 2 GB	(MB)	773.05
TOTAL GETMAINED STORAGE	(MB)	575.00
VIRTUAL BUFFER POOLS	(MB)	429.69
VIRTUAL POOL CONTROL BLOCKS	(MB)	13.43
EDM POOL	(MB)	117.19
COMPRESSION DICTIONARY	(MB)	2.35
CASTOUT BUFFERS	(MB)	9.13
DATA SPACE LOOKASIDE BUFFER	(MB)	0.00
HIPERPOOL CONTROL BLOCKS	(MB)	0.05
DATA SPACE BP CONTROL BLOCKS	(MB)	0.00
TOTAL VARIABLE STORAGE	(MB)	139.53
TOTAL AGENT LOCAL STORAGE	(MB)	53.94
TOTAL AGENT SYSTEM STORAGE	(MB)	32.35
NUMBER OF PREFETCH ENGINES		77.00
NUMBER OF DEFERRED WRITE ENGINES		300.00
NUMBER OF CASTOUT ENGINES		73.00
NUMBER OF GBP WRITE ENGINES		58.00
NUMBER OF P-LOCK/NOTIFY EXIT ENGINES		9.00
TOTAL AGENT NON-SYSTEM STORAGE	(MB)	21.60
TOTAL NUMBER OF ACTIVE USER THREADS		29.67
RDS OP POOL	(MB)	34.54
RID POOL	(MB)	16.97
PIPE MANAGER SUB POOL	(MB)	0.00
LOCAL DYNAMIC STMT CACHE CNTL BLKS	(MB)	0.99
THREAD COPIES OF CACHED SQL STMTS	(MB)	0.00
IN USE STORAGE	(MB)	N/A
STATEMENTS COUNT		N/A
HWM FOR ALLOCATED STATEMENTS	(MB)	N/A
STATEMENT COUNT AT HWM		N/A
DATE AT HWM		N/A
TIME AT HWM		N/A
BUFFER & DATA MANAGER TRACE TBL	(MB)	9.41
TOTAL FIXED STORAGE	(MB)	3.80
TOTAL GETMAINED STACK STORAGE	(MB)	54.71
STORAGE CUSHION	(MB)	112.04

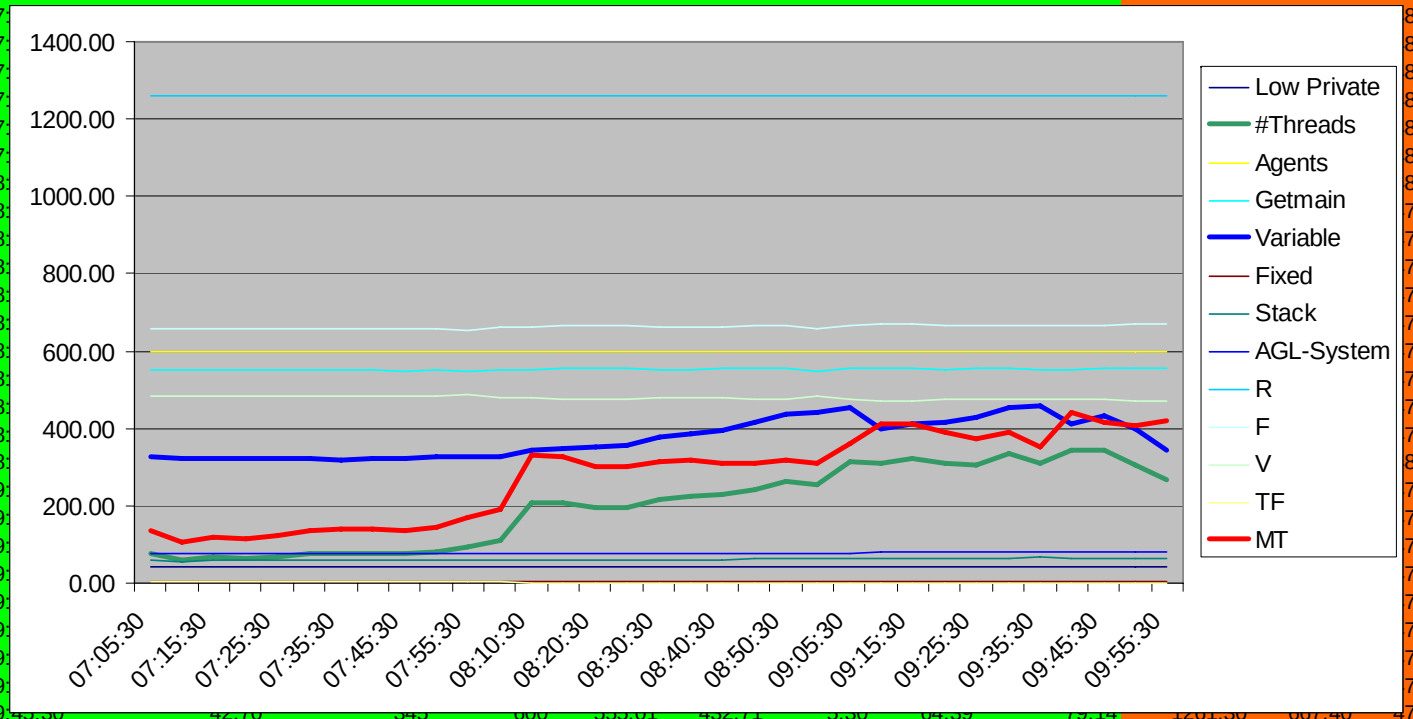
DBM1 AND MVS STORAGE BELOW 2 GB		QUANTITY
24 BIT LOW PRIVATE	(MB)	0.23
24 BIT HIGH PRIVATE	(MB)	2.25
31 BIT EXTENDED LOW PRIVATE	(MB)	27.38
31 BIT EXTENDED HIGH PRIVATE	(MB)	954.23
EXTENDED REGION SIZE (MAX)	(MB)	1714.00
EXTENDED CSA SIZE	(MB)	200.06
AVERAGE THREAD FOOTPRINT	(MB)	3.61
MAX NUMBER OF POSSIBLE THREADS		236.12

DBM1 Storage from Statistics Trace | Report



Study Historical Evolutionary Trend

Basic Cushion ©	163.00								Theoritical	Fixed	Upper		
EPrivate	1304.00	Allied Threads							Max Region	Areas	Limit	Thread	Max
	31Bit Extended	+ DBATs	# System	Total	Total	Total	Total		Size		Variable	Footprint	Threads
Time	Low Private	#Threads	Agents	Getmain	Variable	Fixed	Stack	AGL-System	R	F	V	TF	MT
07:05:30	42.70	78	600	551.70	326.34	5.07	59.69	78.16	1261.30	659.16	481.84	3.51	137
07:10:30	42.70	60	600	551.08	321.43	5.07	56.69	78.16	1261.30	655.54	485.46	4.50	108
07:15:30	42.70	66	600	549.52	322.02	5.07	59.79	78.24	1261.30	657.08	483.92	4.06	119
07:20:30	42.70	63	600	550.09	320.81	5.07	59.66	78.16	1261.30	657.52	483.48	4.24	114
07:25:30	42.70	67	600	550.97	320.89	5.07	59.68	78.16	1261.30	658.42	482.58	3.99	121
07:											483.41	3.52	137
07:											483.78	3.42	141
07:											483.57	3.49	139
07:											485.07	3.57	136
07:											484.08	3.37	144
07:											486.03	2.89	168
08:											480.53	2.52	191
08:											478.81	1.45	331
08:											476.76	1.46	326
08:											475.46	1.57	303
08:											476.51	1.58	301
08:											479.06	1.52	315
08:											478.66	1.51	317
08:											477.91	1.54	310
08:											476.75	1.55	308
08:											476.94	1.51	316
08:											481.59	1.56	308
09:											474.31	1.32	360
09:											472.55	1.14	413
09:											472.04	1.14	413
09:											474.56	1.21	392
09:											473.98	1.27	375
09:											473.81	1.22	388
09:											473.86	1.34	354
09:											476.19	1.08	439
09:											473.60	1.14	417
09:50:30	42.70	306	600	557.81	397.20	5.31	64.14	79.23	1261.30	669.96	471.04	1.16	406
09:55:30	42.70	266	600	557.37	344.21	5.33	64.00	79.14	1261.30	669.40	471.60	1.13	418



Tuning Options

- Full System Contraction (“Reserve Parachute”)
- Turn on Thread Storage Contraction (CONTSTOR=YES)
- Turn on “Best Fit” algorithm for Thread Storage (MINSTOR=YES)
- Reduce size of Local Dynamic Statement Cache (MAXKEEPD)
- Invalidate statements from the Local Statement Cache
- Super Size Bufferpools and More ESA Compression
- Reduce use of RELEASE(DEALLOCATE)
- Reduce size of Extended Common Areas (ECSA, EPLPA, etc)
- Reduce number of long running persistent threads
- Exploit Type 2 Inactive Connections for DDF work
- Switch SMF INTERVAL recording for STCs to NODETAIL
- Implement Data Sharing, increase width of existing Data Sharing Group, or distribute workload around existing Data Sharing Group

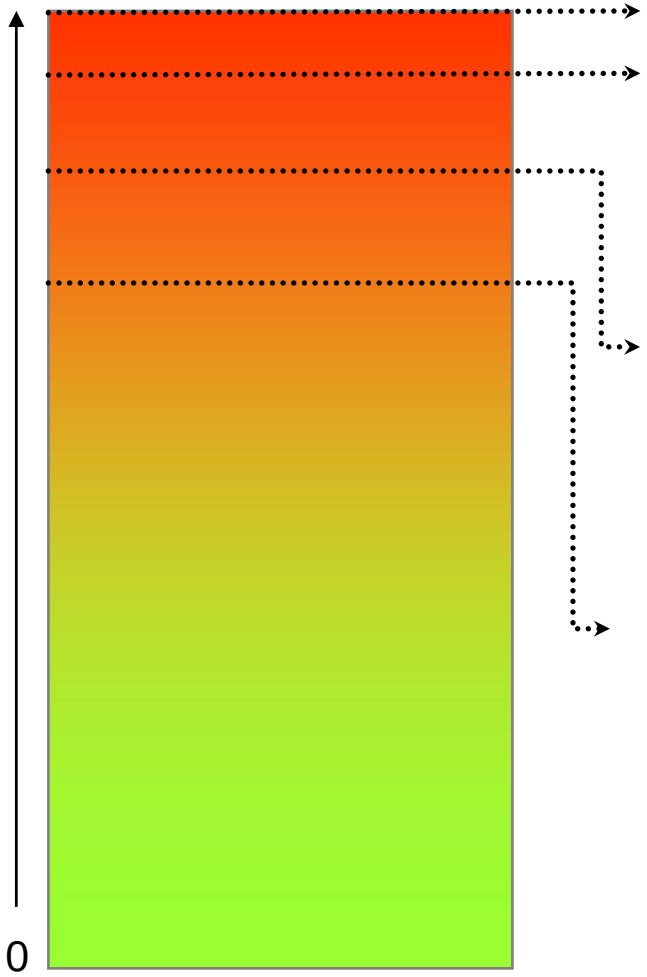
Full System Storage Contraction

Driven by

- ***PVTCRIT*** = MVS Cushion = Storage for Must Complete
  - Fixed, real value
  - Based on CTHREAD/MAXDBAT
  - (CTHREAD+MAXDBAT+1) * ***64K***
- ***PVTMVS*** = MVS Available = Storage for MVS
  - Amount set aside (reserved) for dataset opens
  - Based on DSMAX
  - "Reserve" – decrease on open and increase on close
  - (DSMAX*1300)+40K
- ***PVTSOS*** = MVS Warning To Contract = Cushion Warning
  - Max(5% of Extended Region Size, ***PVTCRIT***)

Do not oversize CTHREAD and MAXDBAT as it inflates Storage Cushion (*PVTSOS+PVTCRIT+PVTMVS*)

Full System Storage Contraction …



Extended Region Size (1650M)

Storage Critical (-12.8M)

- Less than *PVTCRIT+PVTMVS* remaining
- Thread abends start to occur

Storage Cushion (-95.3M)

- Less than *PVTSOS+PVTCRIT+PVTMVS* remaining
- Contraction starts to occur
  - See DBM1 TCB Time for CPU overhead

Basic Storage Cushion (-200M)

Note: Sample values based on V7 customer example

Thread Storage Contraction

- Turned on by zparm CONTSTOR = YES
- Associated CPU overhead (typical < 1-2%)
- Design point is long running persistent threads with RELEASE(COMMIT)
- Compresses out part of Agent Local Non-System storage
- Does not compress
  - Agent Local System
  - Getmained Stack Storage
  - Local Dynamic Statement Cache
- Controlled by two hidden zparms
  - SPRMSTH @ 1048576 (1MB)
  - SPRMCTH @ 10 (commits)
- Triggered at:
  - # Commits > SPRMCTH | (Agent Local Non-System > SPRMSTH & # Commits > 5)

Best Fit Algorithm for Thread Storage

- With zparm MINSTOR=NO (default), first fit algorithm is used
  - Fragmentation may happen with free space (leap frog effect)
- With zparm MINSTOR=YES, best fit algorithm is used instead
  - Will go through all the chains across all the segments
  - Makes the storage denser
  - Observed CPU overhead < 1% for 3-4MB storage pools
  - Danger is that it masks storage leaks (makes them appear to go away)
  - It makes debugging storage leaks more difficult
  - Also degraded slower performance as the storage leak progresses
- Use zparm MINSTOR=YES when
  - System is fully tuned and optimised for storage
  - You have determined there are no leaks
  - Out of other options and need the last ounce of storage

Reduce size of Local Dynamic Statement Cache

- Goal is to reduce storage requirement below the 2GB bar for Thread Copies of Cached SQL Stmts when using KEEPDYNAMIC YES
- But this increase in number “short prepares” and associated increase in CPU resource consumption
- Increase size of EDM Prepared Statement Cache above the 2GB bar to compensate by trying to reduce the number of “full prepares” and offset the increase in CPU resource consumption
- Reduce zparm MAXKEEPD incrementally, and
- Increase size of EDM Prepared Statement Cache above the 2GB bar

Invalidate statements from the Local Statement Cache

- Least Recently Prepared Statements are thrown away from the cache at commit based on MAXKEEPD with KEEPDYNAMIC YES
- APAR PK21861 introduced new zparm CACHEDYN_FREELOCAL
- Ahead of commit based on internal thresholds (subject to change) will invalidate statements from the Local Statement Cache and release the associated storage
- Statements will be purged at end of section
- CACHDYN_FREELOCAL settings

0 = off (default)

1 = If (LDSC >=500MB & DBM1 Used >=75%) then free >= 100KB statement | If DBM1 Used >=85% then free any statement

2 = If (LDSC >=500MB & DBM1 Used >=80%) then free >= 100KB statement | If DBM1 Used >=88% then free any statement

3 = If (LDSC >=350MB & DBM1 Used >=75%) then free >= 100KB statement | If DBM1 Used >=88% then free any statement

Reduce use of RELEASE(DEALLOCATE)

- Use RELEASE DEALLOCATE selectively based on benefit
  - Overuse with persistent threads can create a virtual storage issue
    - Accumulating ever more storage for statements that are not being re-used
    - Storage for unused statements can be left around until deallocation
    - Also drive up demand for EDM Pool resources
    - Ineffective thread and full system storage contraction
  - Best reserved for
    - High volume and/or performance sensitive at reasonable volume OLTP plans/packages
    - Long running batch programs that take frequent intermediate commits

Supersize Bufferpools and ESA Compression

- Super size bufferpools provided fully backed with large real storage
  - Potential for better bufferpool hit ratio
    - Reduce # sync IO waits
    - Reduce DB2 Activity Time
    - Fewer threads to maintain same throughput (VSCR)
- More use of ESA Compression provided fully backed by real storage
  - Less DASD space
  - Faster sequential scan
  - Potential for better bufferpool hit ratio
    - Reduce # sync IO waits
    - Reduce DB2 Activity Time
    - Fewer threads to maintain same throughput (VSCR)

Other Tuning Options

- Reduce appetite for Extended Common Areas (ECSA, …) and give back to get larger Extended Region Size (Max) for DB2
  - IMS users should exploit VSCR features of IMS V7 and later releases
  - Avoid excessive over allocation
- Reduce number of long running persistent DB2 threads
  - If over configured to meet throughput requirement
    - Reduce the number of such threads
  - Ruthlessly cut back on number of JDBC/SQLJ Data Sources
    - Collapse out redundant Data Sources
- Exploit Type 2 Inactive Connections for DDF work
  - Do not use KEEPDYNAMIC(YES) !!!
  - Close open held cursors ahead of commit
  - etc

Other Tuning Options …

- Switch SMF INTERVAL recording for STCs to NODETAIL
  - Problem
    - See Information APAR II07124
    - Symptom SP230 Key storage increasing x MB per day
    - Caused by the amount of SMF Record Type 30 Subtype 4 and Subtype 5 data filling SP230
  - Solution
    - Change from DETAIL to NODETAIL for STC in SMFPRMxx
    - May have to rewrite accounting programs if they are using Subtype 4 and Subtype 5
- Implement Data Sharing, increase width of existing Data Sharing Group, or redistribute work over existing members of Data Sharing Group
  - Redistribute and spread user workload over multiple members
  - Fewer active threads (allied, DBATs) per member

Preventative Maintenance

- Monitor DB2 Storage Information APAR on a weekly basis and apply as preventative service
  - See Info APAR II10817
- Some important storage related APARs to highlight
  - PK21237
    - Reset castout and notify exit engines. Also reduce the number of deferred write engines, GBP write engines, and castout engines.
  - PK21268
    - Move Current Path storage out of stack and above 2GB
  - PK21892
    - Reduce stack storage for ND type CICS threads
  - PK22442
    - DDF address space storage shortage while processing distributed threads with hundreds of output columns causing large SQLDA
  - PK21861
    - Local dynamic statement cache cleanup when infrequent commit and/or high concurrent full Prepare

Protecting The System

- Plan to keep 200MB spare
  - Avoid hitting short on storage and driving Full System Storage Contraction
  - Provide some headroom for:
    - Tuning, some growth, Fast Log Apply, abnormal operating conditions
    - Estimate Maximum Number of Threads that can be supported
      - Allied
      - DBAT
- Set zparms CTHREAD and MAXDBAT to protect the system
  - Theoretical maximum: CTHREAD+MAXDBAT = 2000
  - Practical maximum is much less (typical range 300-850)
  - Avoid over committing resources
  - Deny service and queue work outside the system to keep system alive

Estimating Maximum Number of Threads

"Basic" Formula for estimating Number of Active Threads

Working Max = Extended Region Size
minus 31bit Extended Low Private
minus 200MB (Basic Cushion)

Fixed Areas = Total Getmained Storage below the 2GB bar
plus Total Getmained Stack Storage
plus Total Fixed Storage

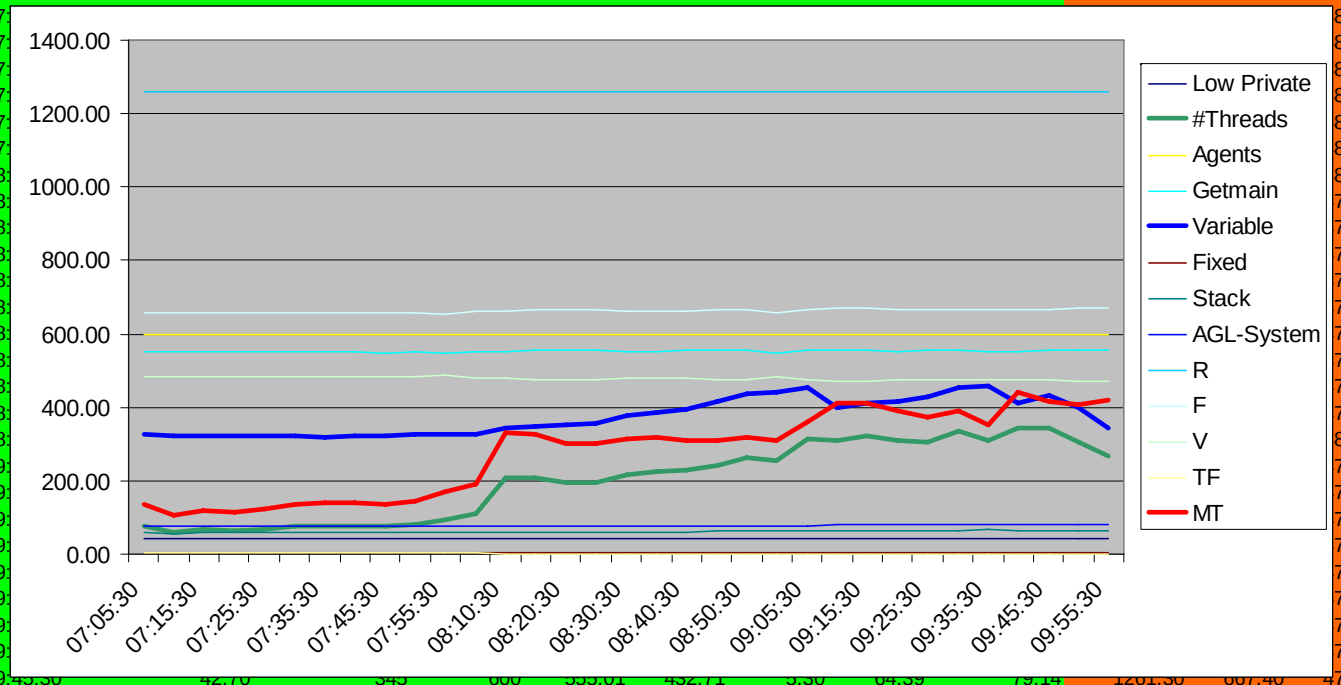
Upper Limit Variable = Working Max minus Fixed Areas

Thread Footprint = (Total Variable Storage minus Total Agent System Storage)
/ (Allied Threads plus *Current* Active DBATs)

Max. No. of Active Threads = Upper Limit Variable / Thread Footprint

Estimating Maximum Number of Threads …

Basic Cushion ©		163.00								Theoritical	Fixed	Upper		
EPrivate		1304.00	Allied Threads							Max Region	Areas	Limit	Thread	Max
	31Bit Extended		+ DBATs	# System	Total	Total	Total	Total		Size		Variable	Footprint	Threads
Time	Low Private		#Threads	Agents	Getmain	Variable	Fixed	Stack	AGL-System	R	F	V	TF	MT
07:05:30	42.70		78	600	551.70	326.34	5.07	59.69	78.16	1261.30	659.16	481.84	3.51	137
07:10:30	42.70		60	600	551.08	321.43	5.07	56.69	78.16	1261.30	655.54	485.46	4.50	108
07:15:30	42.70		66	600	549.52	322.02	5.07	59.79	78.24	1261.30	657.08	483.92	4.06	119
07:20:30	42.70		63	600	550.09	320.81	5.07	59.66	78.16	1261.30	657.52	483.48	4.24	114
07:25:30	42.70		67	600	550.97	320.89	5.07	59.68	78.16	1261.30	658.42	482.58	3.99	121
07:												483.41	3.52	137
07:												483.78	3.42	141
07:												483.57	3.49	139
07:												485.07	3.57	136
07:												484.08	3.37	144
07:												486.03	2.89	168
08:												480.53	2.52	191
08:												478.81	1.45	331
08:												476.76	1.46	326
08:												475.46	1.57	303
08:												476.51	1.58	301
08:												479.06	1.52	315
08:												478.66	1.51	317
08:												477.91	1.54	310
08:												476.75	1.55	308
08:												476.94	1.51	316
08:												481.59	1.56	308
09:												474.31	1.32	360
09:												472.55	1.14	413
09:												472.04	1.14	413
09:												474.56	1.21	392
09:												473.98	1.27	375
09:												473.81	1.22	388
09:												473.86	1.34	354
09:												476.19	1.08	439
09:												473.60	1.14	417
09:50:30	42.70		306	600	557.81	397.20	5.31	64.14	79.23	1261.30	669.96	471.04	1.16	406
09:55:30	42.70		266	600	557.37	344.21	5.33	64.00	79.14	1261.30	669.40	471.60	1.13	418



Real Storage Use

- Important subsystems such as DB2 should not be paging IN from auxiliary storage (DASD)
  - Recommendation to keep page in rates low (near zero)
  - Monitor using RMF Mon III
- Backing rate is dense for 31-bit storage (as before in V7)
- Common misconception!
  - Backing rate is low for 64-bit storage (<10%)
- Increase in real storage with V8 is to be expected as control blocks bigger for 64-bit storage
  - Stack +100%
  - System Threads +40%
  - User Threads: +30-40% static, +50-100% dynamic
  - …

  plus above the 2GB bar storage management!
- Have observed significant growth in real storage demand in some installations

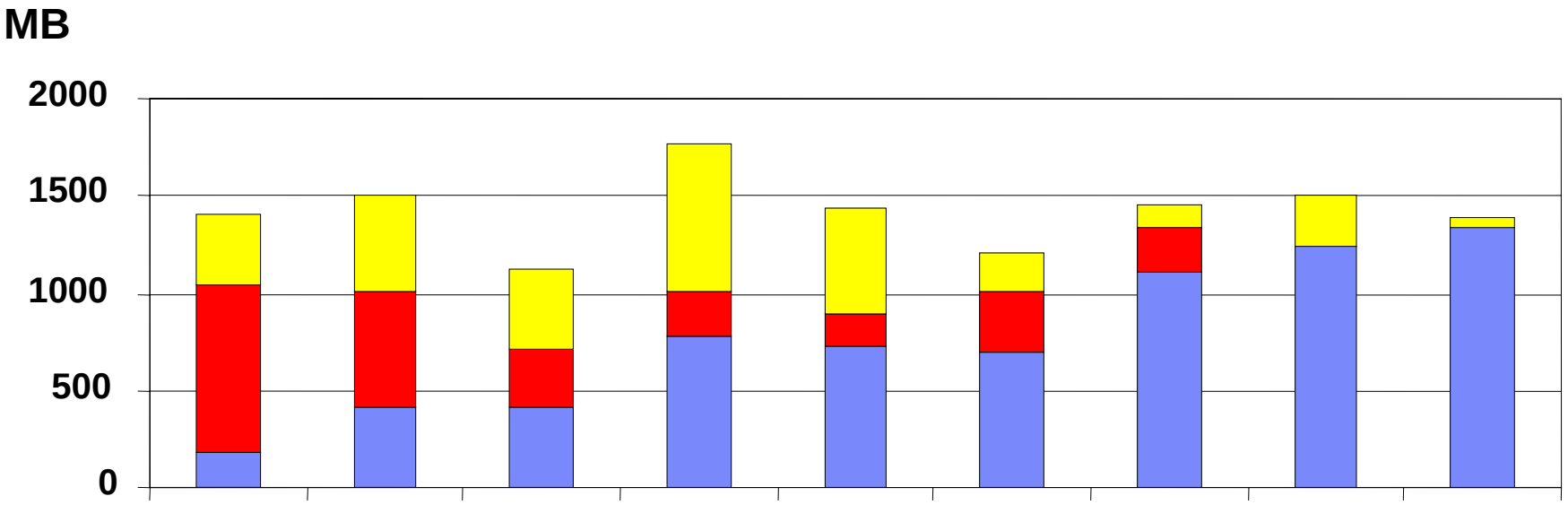
Real Storage Use …

- Recommendation to apply critical must have service
  - PK19769
    - Add MVS discard to pool reset
  - OA15666 and PK25427
    - Discard real frames without hitting AVQLOW condition
  - PK21237
    - Drop number of Write Engines back to pre V8 levels
    - Single 64-bit pool for all Buffer Manager engines
  - PK21892
    - Throw away cached stack on thread deallocation
  - PK25326
    - Contract PLOCK engines after use

Real Storage Use …

- How to estimate real storage requirement for DBM1
  - Method assumes critical maintenance has already been applied
  - Subtract out fixed areas
    - Bufferpools, EDM Pools, …
  - Below the 2GB bar
    - Assume V=R (Virtual = Real)
    - V7 -> V8
      - Stack +100%
      - User Threads +30-40% static, +50-100% dynamic
      - System Threads +40%
      - …
  - Above the 2GB bar
    - Assume 1MB per active thread (pessimistic)
  - Add back the fixed areas

Summary - Value of V8 64-bit Virtual below 2GB bar



Top of ***Yellow*** = Extended Region Size (Max)
Top of ***Red*** = V7 use
Top of ***Blue*** =V8 use

Summary …

- DBM1 64-bit Virtual Memory Map
- DBM1 64 Virtual and Thread Storage
- Projecting V8 from V7 Statistics Trace
- Key Messages
- Problem Recap and Driving Factors
- Analysing Virtual Storage Used
- Tuning Options
- Protecting The System
- Real Storage Use
- Summary