From VIRTUALly Constrained to REALly overcommitted

Friday, August 14, 2015: 10:00 AM - 11:00 A
Dolphin, Southern Hemisphere 5

Adrian Burke
DB2 SWAT team SVL
agburke@us.ibm.com

#SHAREorg

SHARE is an independent volunteer-run information technology association that provides **education**, professional **networking** and industry **influence**.

Agenda

- Storage terms and concepts
- Address spaces and DB2
- REAL and AUX impact on DB2
- Buffer Pools
- LFAREA
- Flash Express

Ma 1974 via Flickr

Virtual storage concepts

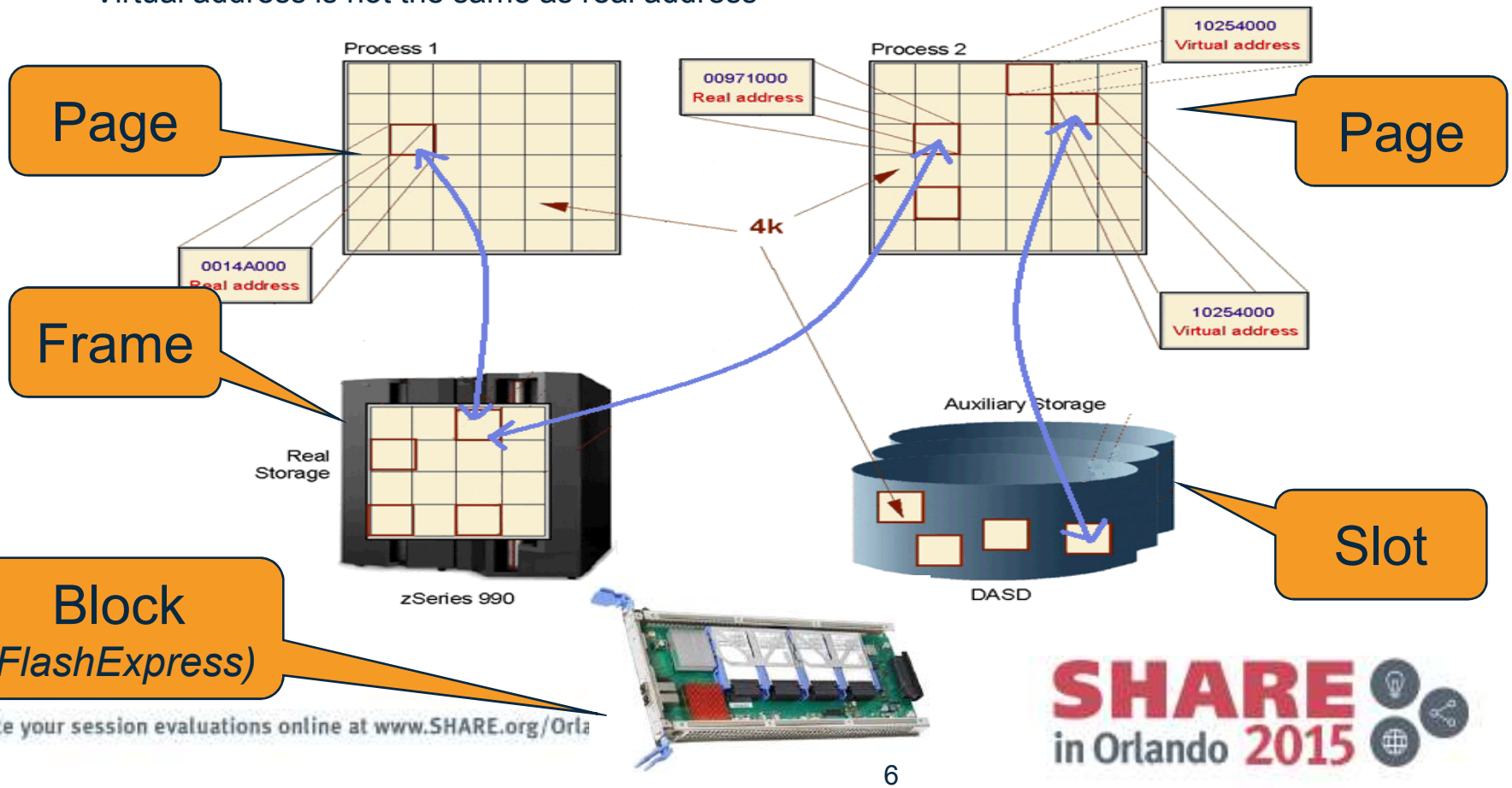
- *Virtual storage* is an “illusion” created through z/OS management of real storage and auxiliary storage through mapping tables
  - It allows each program access to more storage than exists on the box
  - REAL is what you paid for on the CEC
  - AUX is how many page data sets your DASD folks set aside to be used
- The executing portions of a program are kept in *real storage*; the rest is kept in *auxiliary storage*
- Range of addressable virtual storage available to a user or program or the operating system is an *address space*
- Each user or separately running program is represented by an *address space* (each user gets a limited amount of private storage)

How virtual storage works

- Virtual storage is divided into 2GB – 8GB *regions* composed of 1-megabyte *segments* composed of 4-kilobyte *pages*
  - z/OS uses region, segment and page tables to keep track of pages
- Transfer of pages between auxiliary storage and real storage is called *paging*
- When a requested page is not in real storage, an interruption (called a *page fault*) is signaled and the page is brought into real
- Addresses are translated dynamically, a process called *Dynamic Address Translation* (DAT)
- *Frames* and *slots* are repositories for a page of information
  - Page is a 4k,8k,16k,1MB grouping representing frames (Virtual Storage)
  - A frame is a 4K piece of real storage (Central storage)
  - A slot is a 4K record in a page data set (AUX)
  - A block is a 4k record in FlashExpress (SCM)

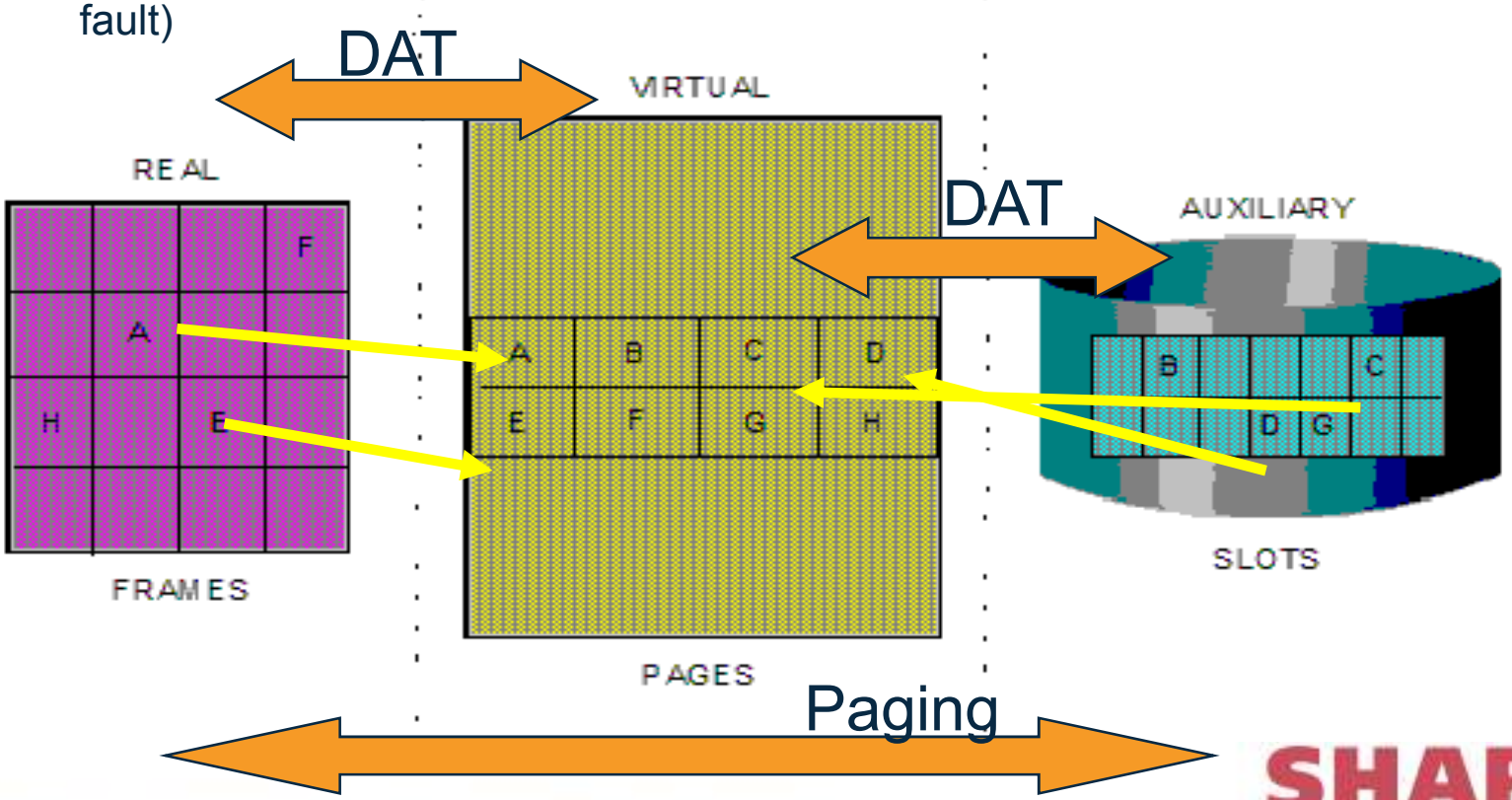
How virtual storage works (continued…)

- We can only read/write to virtual page or AUX slot if it is in REAL memory, so we need to find it or page it in
  - You can have more than 1 virtual address that maps back to a 1 frame of real storage – hence DBM1 and DIST have 64-bit shared in V9
    - Virtual address is not the same as real address



Pages, Frames, and Slots (continued)

- Dynamic Address translation (DAT)
  - How we determine where the block of storage we need is, could be in virtual or in AUX (DASD)
  - If it is out in AUX then we need a REAL 4k frame fixed in real first (page fixing buffers)
  - Then we need to page that 4k AUX slot into real memory so we can read/write it (page fault)



Page Stealing

- z/OS tries to keep an adequate supply of available real storage frames on hand
- When this supply becomes low, z/OS uses *page stealing* to replenish it (LRU mechanism, unreferenced interval count)
- Pages that have not been accessed for a relatively long time are good candidates for page stealing.
- z/OS also uses various storage managers to keep track of all pages, frames, and slots in the system:
  - Auxiliary Storage Manager (ASM)
  - Real Storage Manager (RSM)
  - Virtual Storage Manager (VSM)
- If you run out of AUX storage (page data sets full) z/OS will come down

Available Frame Queue Processing

RSM manages the queue and starts the stealing process when the number of frames falls below the threshold

There is a ‘low’ level where stealing begins, and ‘ok’ value above which stealing stops

- **MCCAFCTH=(*lowvalue*,*okvalue*) defaults in IEAOPTxx**
  - LOW will vary between MAX(MCCAFCTH lowvalue, 400, 0.2% of pageable storage)
  - OK will vary between MAX(MCCAFCTH okvalue, 600, 0.4% of the pageable storage)
    - RSM will automatically adjust the actual threshold values based on measurements of storage usage but doesn’t let values get lower than MCCAFCTH low threshold
  - Typically no need to specify this parameter

Messages to be aware of

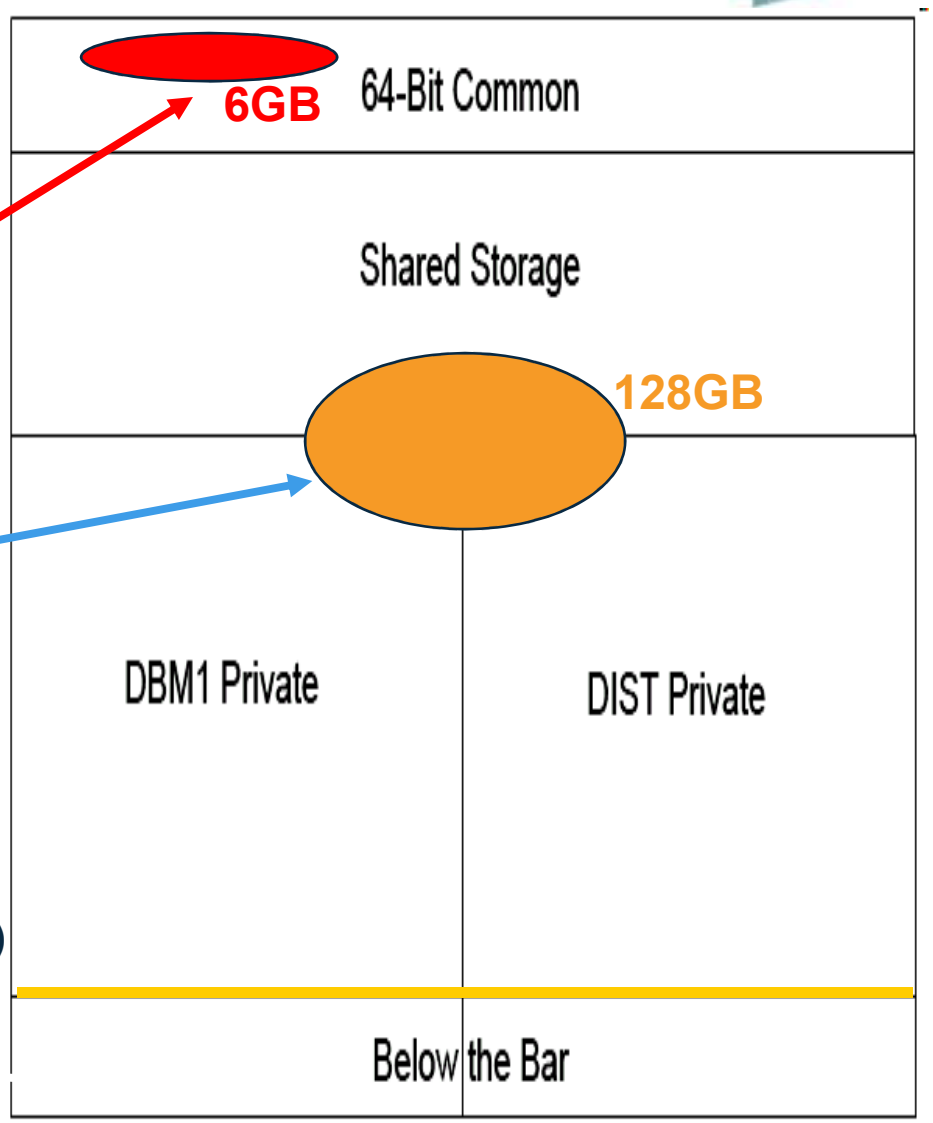
- **REAL Storage – (too much fixed, not enough pageable)**
  - Informational level **50%**
    - Issue ENF 55
    - Issue IRA405I: % of real is fixed
  - Shortage level **80%**
    - Issue ENF 55
    - Issue IRA400E: shortage of pageable frames and list of top 20 consumers
    - Issue IRA403E swapping culprit out (in-real, moving it higher)
    - Issue IRA410E set non-swappable AS as non-dispatchable (STORAGENSWDP=YES)
  - Critical Shortage level (everything seen in Shortage +) **90%**
    - Issue IRA401E Critical paging shortage
    - If this case continues for 15 seconds issue IRA420I and IRA421D to show largest consumers and allow operator to cancel them

Messages to be aware of…

- **AUX Storage**
  - Informational level
    - Issue ENF 55
    - Issue IRA205I: % of AUX is allocated
    - Also where DB2 goes into hard DISCARD mode
  - Warning level
    - Issue ENF 55
    - Issue IRA200E and IRA206I: shortage of AUX and list of top 20 consumers
    - Issue IRA203E swapping culprit out
    - Issue IRA210E set non-swappable AS as non-dispatchable (STORAGENSWDP=YES)
  - Critical Shortage level (everything seen in Shortage +)
    - Issue IRA201E Critical paging shortage
    - If this case continues for 15 seconds issue IRA220I and IRA221D to show largest consumers and allow operator to cancel them

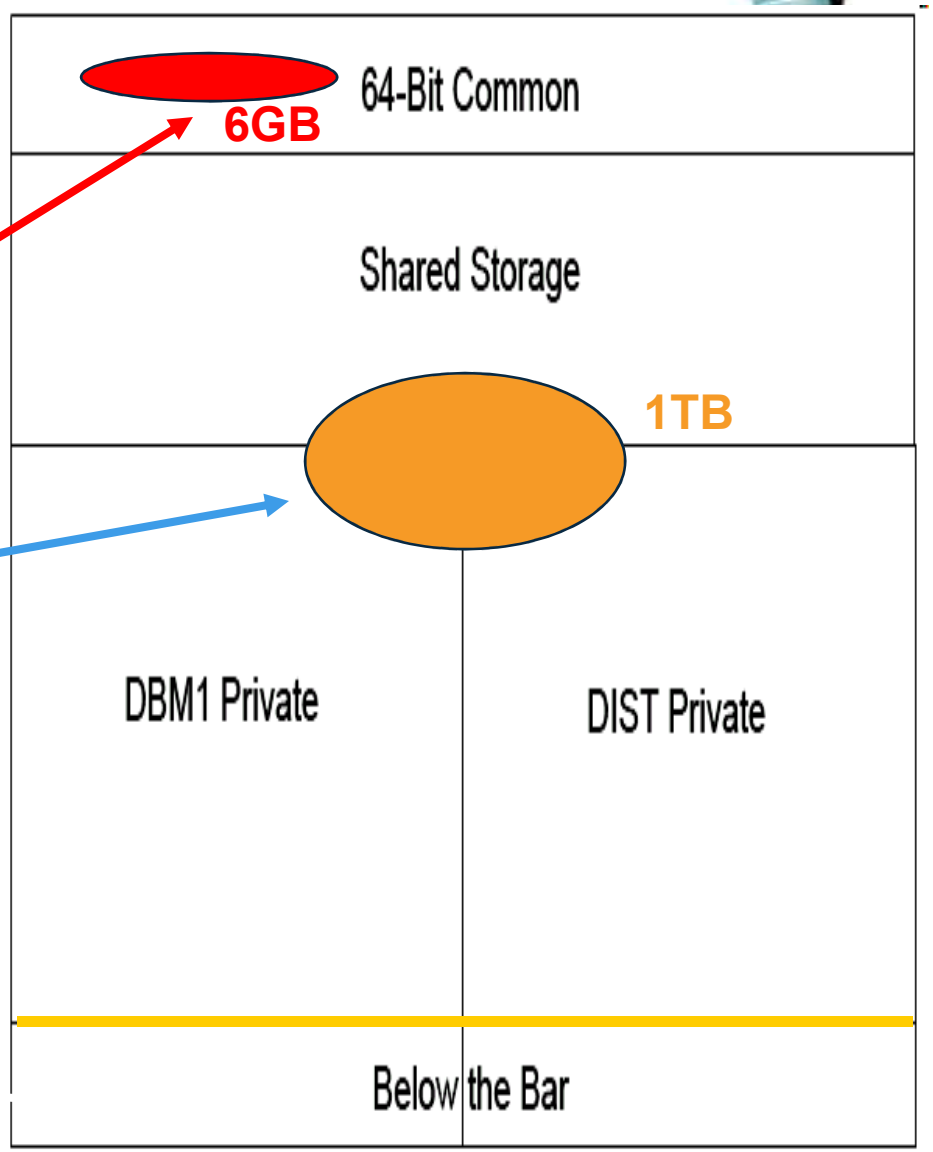
MVS Storage DB2 10

- Almost everything is above the bar:
  - –6GB Common per subsystem: z/OS 1.13 has 64GB as default
    - –If you have many subsystems on same LPAR this may not be enough
  - –128GB Shared per subsystem came in v9:: now in V10 this is where all the thread storage went
    - –In V9 only DRDA comm. area and trusted context ran here
    - –510 TB default in z/OS 1.13
- We still have 31bit stack for agents (threads)
  - –16K for system agents
  - –32k for attach
  - –xPROCS still here, and reported in IFCID 225



MVS Storage DB2 11

- Almost everything is above the bar:
  - –6GB Common per subsystem: z/OS 1.13 has 64GB as default
    - –If you have many subsystems on same LPAR this may not be enough
    - –OUTBUFF +15% goes here
  - –1TB Shared per subsystem:: xProc storage shared among threads now
    - –510 TB default in z/OS 1.13
    - –xPROCs moved above bar
- Still a small amount for threads
  - –31-bit low private is now eligible to be backed by 1 MB frames



What affects the storage?

- 64-bit private
  - Mostly buffer pools
  - Fixed-RID pool
  - EDM pool
- 64-bit shared thread and system
  - Used to execute SQL
  - Variable-dynamic statement caches, CT, PT, SKCT, SKPT
  - *Compare to number of system and user agents/threads, parallelism
- 64-bit stack
  - Just as in V9, used by thread to execute SQL
  - *same
- 64-bit common
  - Distributed agents, package accounting, rollup
- All are affected by REALSTORAGE_MANAGEMENT

REALSTORAGE_MANAGEMENT

- **OFF** Do not enter contraction mode unless the REALSTORAGE_MAX boundary is approached OR z/OS has notified us that there is a *shortage of AUX storage*

- **ON** Always operate in contraction mode. This may be desirable for LPAR with many DB2s or dev/test systems

- **AUTO (the default)** When significant paging is detected, contraction mode will be entered, done based on deallocated threads and number of commits, private pools contracted on similar boundaries

- Important notes:
  - Contraction mode is not exited immediately upon relief to avoid constant toggling in and out of this mode
  - Contraction mode shows <1% CPU degradation with no detectable impact to running workloads

ZPARM REALSTORAGE_MANAGEMENT= ??

- Affects when DB2 releases (discards) 64-bit storage (default is AUTO)
  - KEEPREAL(YES) means it still has DB2's name on it but is released to z/OS, while KEEPREAL(NO) means the storage is completely freed
- RSM=OFF means No DISCARD
- "Soft discard" is DISCARD with KEEPREAL(YES)
  - RSM=AUTO **with no** paging means DISCARD with KEEPREAL=YES at Thread Deallocation or 120 commits
  - RSM=AUTO **with** paging means DISCARD with *KEEPREAL=YES* at Deallocation or 30 commits. STACK also DISCARDED
  - RSM=ON means DISCARD with *KEEPREAL=YES* at Deallocation or 30 commits. STACK also DISCARDED

Storage Contraction

- "Hard discard" is DISCARD with KEEPREAL(NO)
  - ENF 55 signal when 50% of AUX storage is in use means DISCARD KEEPREAL=NO
  - Hitting REALSTORAGE_MAX means DISCARD KEEPREAL=NO at 100%
  - We now report number of contractions in MEMU2 as well IFCID 001 (DISCARDMODE64 or RSMAX_WARN)
- Important notes:
  - Contraction mode is not exited immediately upon relief to avoid constant toggling in and out of this mode
  - Contraction mode shows <1% CPU degradation with no detectable impact to running workloads
- Contraction mode start is indicated by a DSNV516I message
  - **DSNVMON – BEGINNING STORAGE CONTRACTION MODE**
- Ending with a DSNV517I message
  - **DSNVMON – ENDING STORAGE CONTRACTION MODE**

Paging Public Service Announcement… Why?

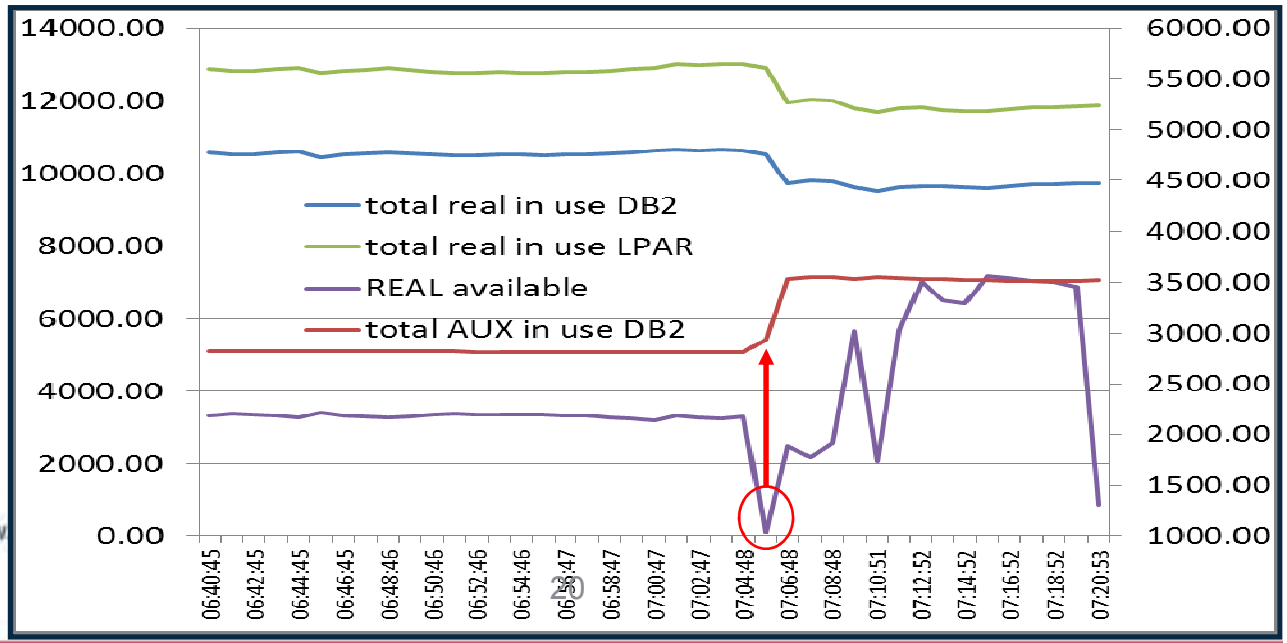
- What is a tolerable paging rate?
  - Only during dramatic workload shifts, hopefully storage isn't needed anymore??
    - i.e. batch to online and vice versa?
  - This inherently means you have no 'cushion' for DUMPs, unexpected SORTing or Utility/Batch job which ran at an inopportune time….
- ….DB2's answer is never
- z/OS lab measured cost of I/O at 20us-70us of CPU
- Don't forget the elapsed time wasted as well (2ms avg.)
- Most customers have undersized dev/test environment….
  - **But at ~5us synch I/O (+1us no page fixing) and ~20us for prefetch (+5us for no page fixing) you are trading CPU for REAL memory
    - DB2 path length measurement… much bigger if measured end-to-end
  - So why not right-size it to save CPU, and actually mirror the performance of what it will be in production??
- Storage $$$$ → $$ so trade your CPU cycles for Memory

Potential DB2 Benefit from Larger Memory

- DB2 local and group buffer pools
  - Reduction of elapsed time and CPU time by avoiding I/Os
  - PGSTEAL(NONE) in DB2 10 = In memory data base
  - CPU reduction from PGFIX=YES and large page frames
- Thread reuse with IMS or CICS applications
  - Reduction of CPU time by avoiding thread allocation and deallocation
- Thread reuse and RELEASE(DEALLOCATE)
  - Reduction of CPU time by avoiding package allocation and parent locks
  - DDF High performance DBATs support with DB2 10
  - Ability to break-in persistent thread with DB2 11
- Global dynamic statement cache
  - EDMSTMTC up to 4G with DB2 11, default 110MB
  - Reduction of CPU time by avoiding full prepare
- Local statement cache
  - MAXKEEPD up to 200K statements with DB2 11, default 5000
  - Reduction of CPU time by avoiding short prepare
- In-memory data cache for sparse index
  - MXDTCACH up to 512MB per thread, default 20MB
  - Reduction of CPU and elapsed time with potentially better access path selection with DB2 11
- RID Pool (MAXRBLK)
  - Lower CPU time if RID lists don't spill into work files

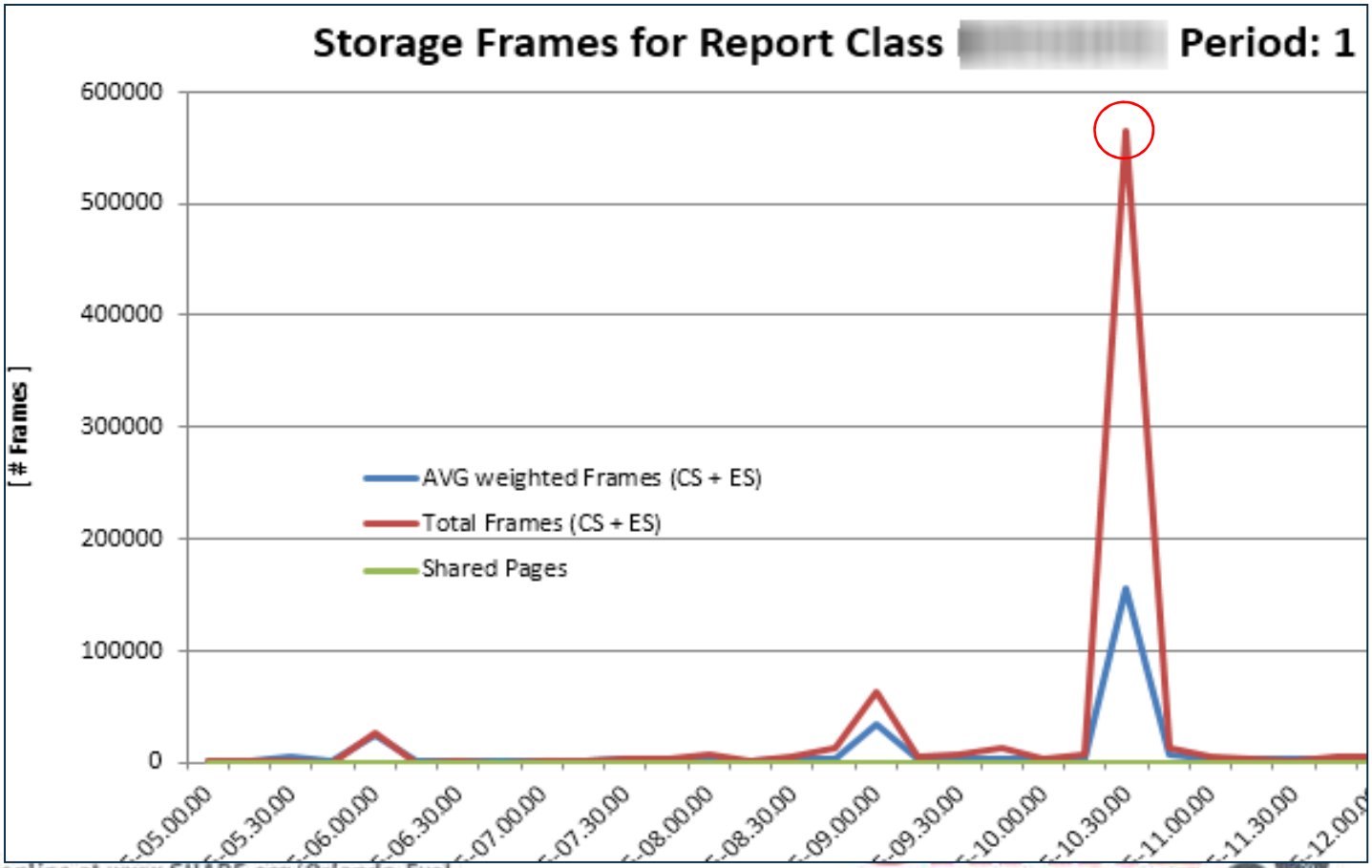
DB2 Storage

- In the graphic we can see DB2 storage goes out from REAL to AUX when the real available drops to '0' on the LPAR
- Worst case in this example to get those pages back in:
  - 700 MB – sync I/O time ~3ms = 0.003*179,200 = 537 seconds
  - If those pages were taken out of our buffer pools then we need to spend the I/Os to get the pages back in central storage – no prefetch from AUX
- Imagine a 16GB SVCDUMP occurring here!!
  - MAXSPACE = 16GB to allow for dump, should you reserve this space during peak processing hours?
  - MAXSNDSP = 15 seconds default (amount of time system non-dispatchable)
  - AUXMGMT=ON – no new dumps when AUX=50%, MAXSPACE always honored



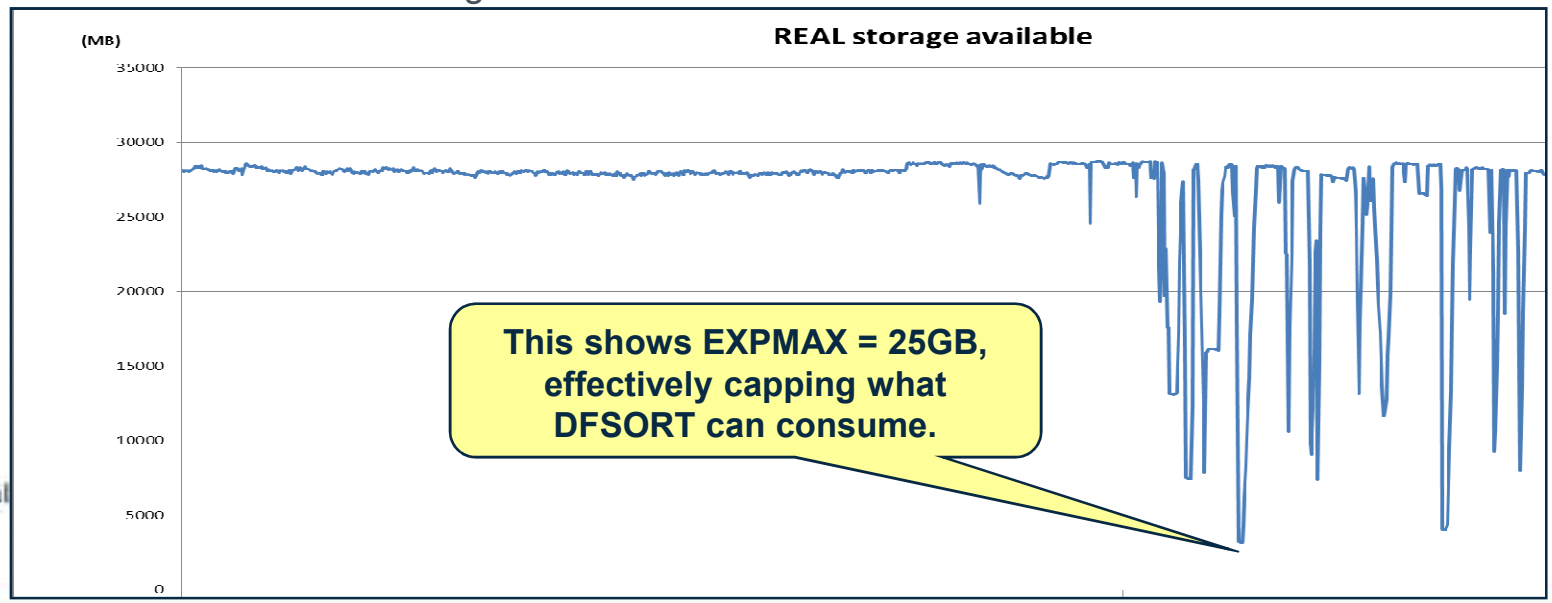
DB2 Storage

- So who caused me to get paged out??
  - If you run a WLM activity report and look at the Storage Trend graph in the reporter you can see the actual frames used by a service or report class
  - The Page In Rates would also be high for that report class as data is brought in



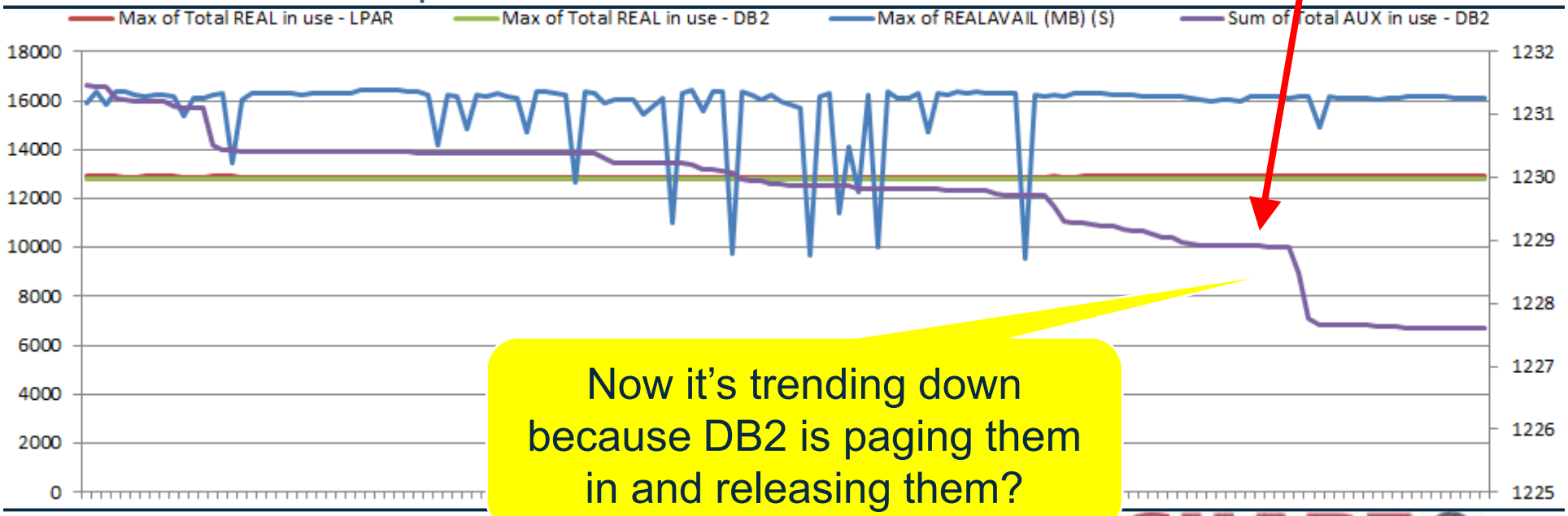
Real storage and Sort products

- By default DFSORT and other sort products usually take as much storage as they can get, to help performance… but what about everyone else?
- DFSORT parameters affecting storage use (II13495) → means to protect DB2
  - These can be dynamically changed for workloads using ICEPRMxx member
  - EXPOLD = % of storage allowed to be pushed to AUX → 0
  - EXPRES= % of storage to preserve, maybe in case of DUMPSPACE/ MAXSPACE → 16GB min in V10
  - EXPMAX=% of storage for memory object and hyperspace sorting, somewhat depends on EXPOLD and EXPRES → how much can you spare
- DB2Sort has:
  - PAGEMON=ON/OFF to limit central storage usage based on paging
  - PAGEMONL= total # of MB's of storage DB2SORT is allowed to consume



More on AUX storage

- AUX storage has been referred to as ‘double accounting’
- Once the page data sets (AUX slots) are utilized by an address space they remain assigned to that address space
- UNTIL ‘DB2’ is bounced, ***or we page them in and release the AUX slot***… it looks like we are using it..
  - We have a requirement out to z/OS to address this



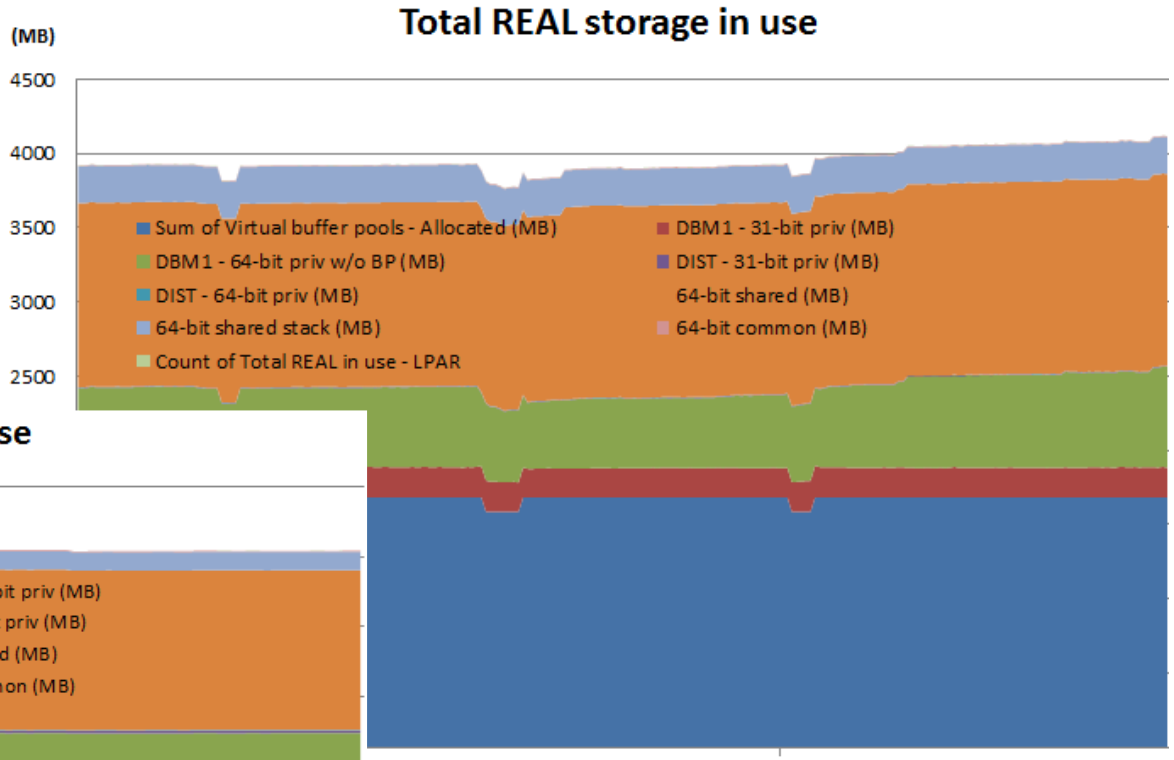
z/OS XCF Critical Paging

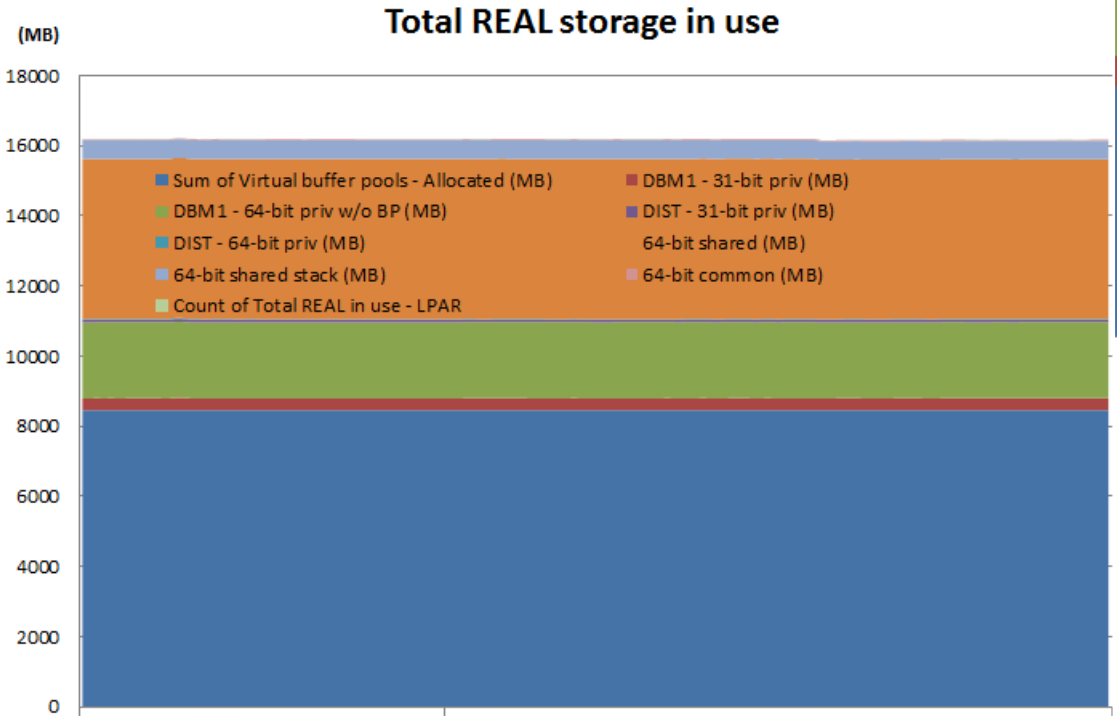
- Critical paging is a z/OS service designed to avoid page faults occurring for critical address spaces during a Hiperswap event
  - The downside of this is a massive amount of fixed storage to include the following:
    - 31- bit common storage (both above and below 16M)
    - Address spaces that are defined as critical for paging
    - All data spaces associated with those address spaces that are critical for paging (unless CRITICALPAGING=NO was specified on the DSPSERV CREATE)
    - Pageable link pack area (PLPA)
    - Shared pages
    - All HVCOMMON objects
    - All HVSHARED objects
- Apply z/OS APAR OA44913
  - Allows z/OS to reclaim DB2 64-bit SHARED KEEPREAL=YES frames
- If Flash Express (SCM) is installed HVSHARE and HVCOMMON can page to it, otherwise they are protected until real storage critical threshold

- How do I know if I have it? - D XCF,COUPLE

What is in REAL now?

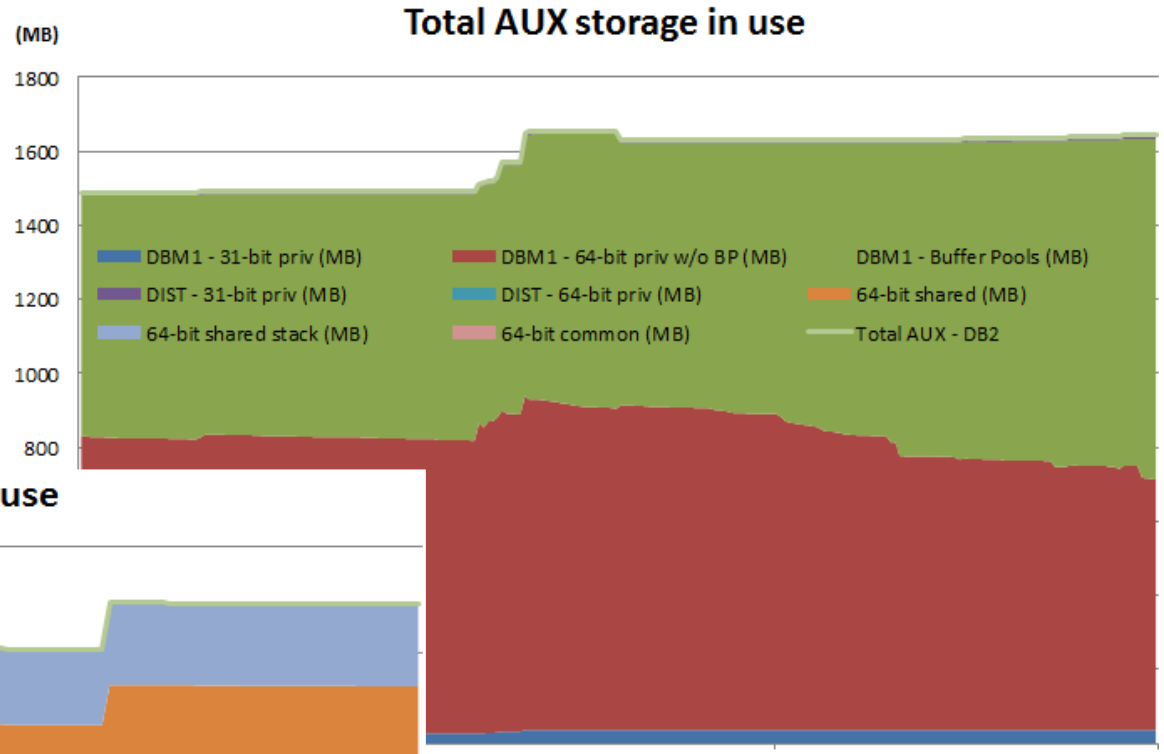
- 2 different customer's REAL storage usage in DB2, proportionally similar for address space usage

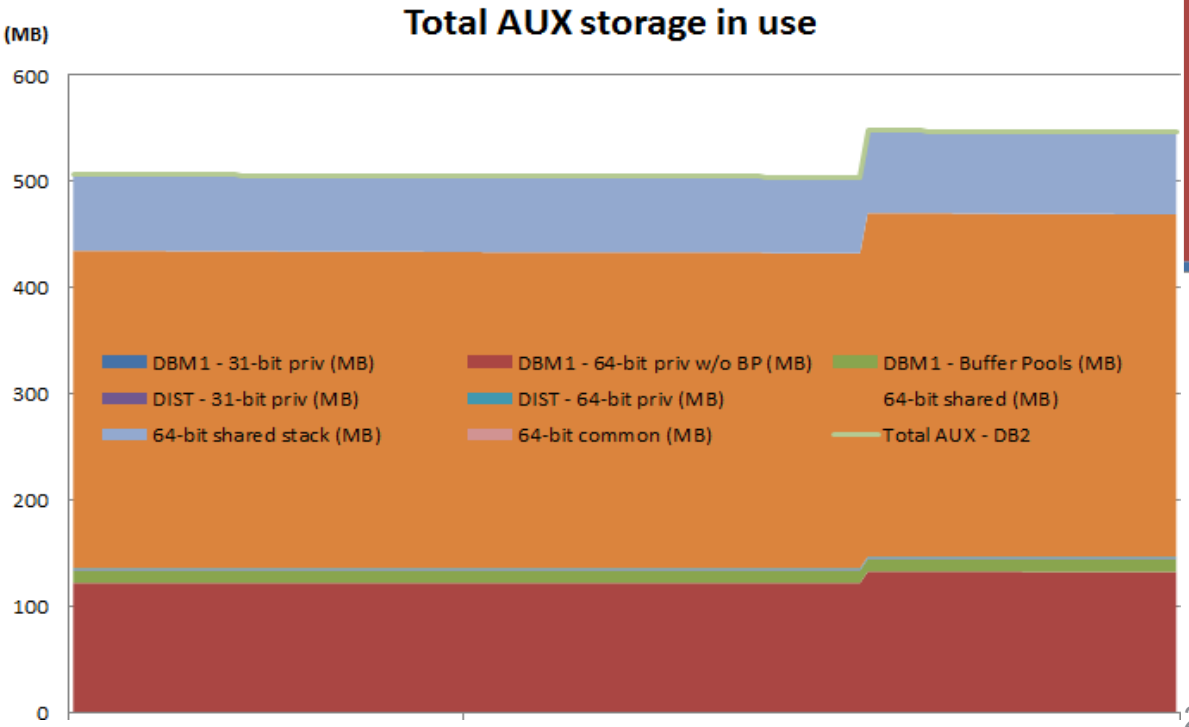




What is in AUX now?

- With CRITICALPAGING =YES HVSHARE becomes non-pageable so that leaves buffer pools and PRIVATE to be sacrificed





- Buffer pools are not paged out in customer #1's environment, but they are in #2 causing more I/Os (no prefetch)

Buffer Pools

Residency time

- Residency Time = average time that a page is resident in the buffer pool
  - System RT (seconds) = VPSIZE / Total pages read per second
  - Total pages read = synchronous reads for random getpages + synchronous reads for sequential getpages + pages read via sequential prefetch + pages read via list prefetch + pages read via dynamic prefetch
- ** Hit Ratios although popular will vary significantly between DB2 10 and DB2 11 due to classification/reclassification of buffers
  - In DB2 9 and 10 getpages that resulted in dynamic prefetch were classified as random, but the buffer was classified as sequential (inflates random hit ratio)
  - Also when a sequential buffer was touched by a random getpage it remained on the sequential LRU chain (might be stolen sooner)

Residency time…

- General Rule of Thumb of 60 seconds
  - The longer the page is resident in the BP the higher the likelihood of the page being re-referenced and avoiding an I/O in the future
    - Behavioural differences in DB2 10 and DB2 11 will cause this number to vary
  - If the BP shows a 30 second residency time, and we double the size to get to 60 seconds we have only seen linear improvement
  - Ideally we want the absolute number of I/Os to decrease with a larger pool, thus show savings from re-reference of pages
    - Thus if we aim for a 60 second residency, but doubling the pool ends up with an average residency of >60 seconds, we have achieved incremental performance gains
  - There is no magic number for page residency, but it is useful in determining the ratio of size increase, to I/O decrease

BP Simulation

- Ability to simulate changing VPSIZE and VPSEQT
  - Assumes LRU mechanism in place
- Storage cost for a simulated buffer pool is less than 2% for 4K pages similar to Hiperpool except not allocating actual buffers
  - Simulation is done with a separate buffer search hash table with buffer control blocks
  - For 4K page size BP, the storage overhead is 2% of SPSIZE * 4K (e.g. 20MB for 1GB size buffer pool)
  - CPU overhead is between 1-3% when running lab. Workloads
  - Shows up in DSNB431I-432I, and OMPE (IFCID 002)

```
DSNB432I  -CEA1 SIMULATED BUFFER POOL ACTIVITY -
          AVOIDABLE READ I/O -
           SYNC  READ I/O (R)   =365071
           SYNC  READ I/O (S)   =5983
           ASYNC READ I/O       =21911
           SYNC  GBP READS (R)  =89742
           SYNC  GBP READS (S)  =184
           ASYNC GBP READS      =279
       PAGES MOVED INTO SIMULATED BUFFER POOL
=13610872
           TOTAL AVOIDABLE SYNC I/O DELAY =158014 MS
```

BP Thresholds to be aware of

- **Immediate Write Threshold** — 97.5% dirty pages
  - Checked at page update
  - After update, synchronous write
- **Data Management Threshold** — 95% dirty pages
  - Checked at page read or update
  - Getpage can be issued for each row sequentially scanned on the same page – potential large CPU time increase
- **Prefetch Threshold** — 90% dirty pages
  - Checked before and during prefetch
  - 90% of buffers not available for steal, or running out of sequential buffers (VPSEQT with 80% default)
    - Disables Prefetch (PREF DISABLED – NOBUFFER)

Long-Term Page Fix for BPs with Frequent I/Os

- DB2 BPs have always been strongly recommended to be backed up 100% by real storage
  - Shoot for *PAGE-IN for READ / WRITE* in statistics to be **'0'**
  - Even if PAGE-IN = 0, if your buffers were paged out to AUX you would be incurring I/Os to the page data sets which won't show up in the stats
- Given 100% real storage, might as well page fix each buffer just once to avoid the repetitive cost of page fix and free for each and every I/O
  - New option: ALTER BPOOL(name) PGFIX(YES|NO)
    - Requires the BP to go through reallocation before it becomes operative
      - Means a DB2 restart for BP0
  - Up to 8% reduction in overall IRWW transaction CPU time (30% DBM1 CPU time)
    - About 1,000 instructions for fix/free

Long-Term Page Fix for BPs with Frequent I/Os

- Recommended for BPs with high I/O intensity
  - I/O intensity = [pages read + pages written] / [number of buffers]
  - Relative values across all BPs

BPID	VPSIZE	Read Sync	Read SPF	Read LPF	Read DPF	Read - Total	Written	I/O Intensity
BP0	40000	2960	0	0	6174	9134	107	0.2
BP1	110000	12411	5185	0	1855	19451	6719	0.2
BP2	110000	40482	19833	11256	9380	80951	5763	0.8
BP3	75000	23972	6065	0	14828	44865	7136	0.7
BP4	80000	22873	45933	3926	50261	122993	3713	1.6
BP5	200000	0	0	0	0	0	0	0.0
BP8K0	32000	9	11	0	11	31	169	0.0
BP32K	2000	693	873	0	6415	7981	38	4.0

In this example: Best candidates would be BP32K, BP4, BP2, BP3
No benefit for BP5 (data in-memory)

Buffer Pool enhancements…

Frame size	Page fix	Supported DB2	H/W Requirement	Benefit
4K	NO	All	N/A	Most flexible configuration
4K	YES	All	N/A	CPU reduction during I/O
1M	NO	DB2 10 with APAR, or DB2 11	zEC12 and Flash Express compatible Backed by real or LFAREA	CPU reduction from TLB hit Control Blocks only
1M	YES	DB2 10 above	z10 above LFAREA 1M=xx	CPU reduction during I/O, CPU reduction from TLB hit
2G	YES	DB2 11	zEC12 LFAREA 2G=xx	CPU reduction during I/O, CPU reduction from TLB hit

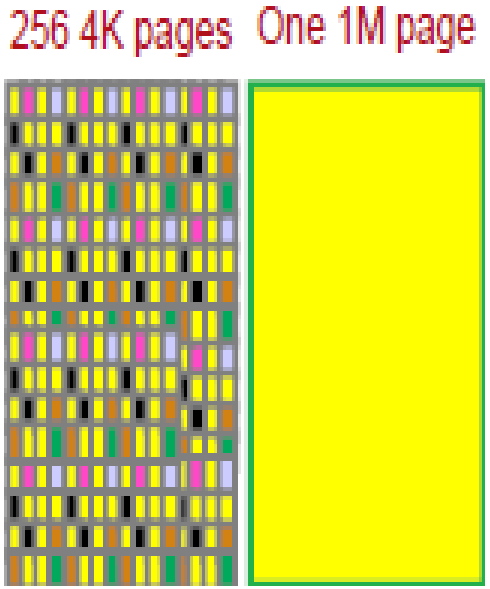
BP Summary

- **Page fixing has highest benefit for pools with high I/O intensity**
- **Backing with larger frames has highest benefit for pools with high get page counts**
  - Potential for reduction of CPU through less TLB misses
  - CPU reduction based on customer experience 0 to 6%
  - Buffer pools must be defined as PGFIX=YES to use 1MB size page frames until V11
    - **With PM85944 PMB control blocks can be backed by pageable 1MB frames**
  - Must have sufficient total real storage to fully back the total DB2 requirement
- Involves partitioning real storage into 4KB and 1MB size page frames
  - Specified by LFAREA xx% or n in IEASYSnn parmlib member
  - Only changeable by IPL so pad it out!!
- If 1MB size page frames are overcommitted, DB2 will use 4KB size page frames
  - Recommendation to add 5-10% to the size to allow for some growth and tuning
  - Must have both enough 4KB and enough 1MB size page frames

- **Apply PI12512 (V10) – need minimum of 6656 buffers for pool to be backed with 1 MB frames

1MB Frames

- TLB – translation lookaside buffer is a 'fast-path' through translation between virtual and REAL
  - coverage today represents a much smaller fraction of an application's working set size leading to a larger number of TLB misses
  - More TLB hits equates to CPU savings
- Applications can suffer a significant performance penalty resulting from an increased number of TLB misses
- Solution:
  - Increase TLB coverage without proportionally enlarging the TLB size by using large pages
- Large Pages allow for a single TLB entry to fulfill many more address translations
- Large Pages will provide exploiters with better TLB coverage



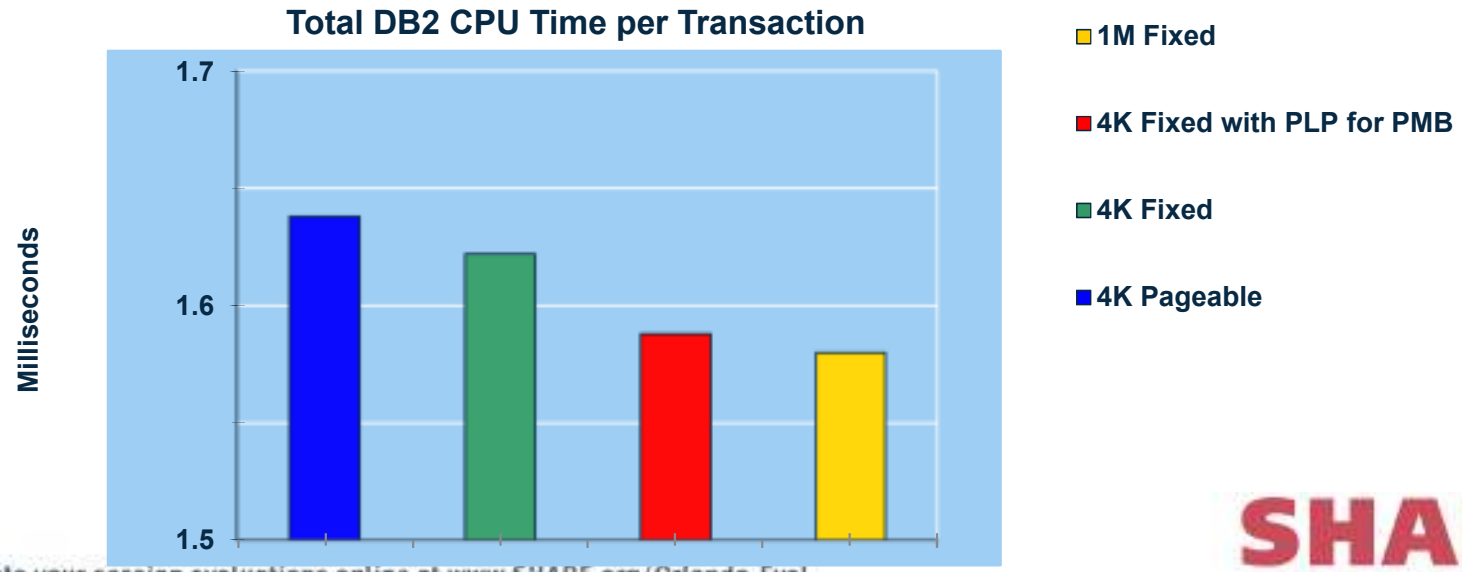
1MB Large Frames for BPs with highest getpages

- First exploited by DB2
  - Page fixed buffer pools in DB2 10
    - As opposed to I/O intensity, use large frames for pools with highest getpage counts
  - DB2 11 brings
    - 31-bit low private system storage
    - non-page fixed BP control blocks (PMBs) if it is FlashExpress compatible HW (also retrofit to DB2 10)
    - Ability to specify 1MB frame BPs without page fixing
    - OUTBUFF backed with 1MB frames
  - JAVA heap storage (controlled explicitly and monitored through traces)
- To allocate 1MB frames IEASYSxx in PARMLIB governs z/OS storage allocations
  - LFAREA=(1M=(*target*[%],*minimum*[%])
  - 2G=(*target*[%],*minimum*[%]))
    - We try to meet the targets, but if minimums cannot be met a console message is issued

Large Frames for BPs

All of buffer pools are backed by real storage

- zEC12 16 CPs, 5000-6000 tps (mid to complex transactions) with 70 GB local buffer pools
  - 120GB real storage with 75GB LFAREA configured for 1MB measurements
- 1MB frames with PGFIX=YES (long term page fix) is the best performer
- 4KB frames using PGFIX=YES and zEC12 Flash Express exploitation (1MB Pageable PMBs) is good alternative
  - Note : 70GB buffer pools are used, 8-10 sync I/O, 370 getpages per transaction



What about LFAREA?

- Useful commands
  - -DISPLAY BUFFERPOOL(BP1) SERVICE(4)
    - Useful command to find out how many 1MB size page frames are being used → *DSNB999I =D2V1 4K PAGES 0 DSNB999I =D2V1 1M PAGES 1476*
  - -DISPLAY VIRTSTOR,LFAREA

IAR019I 14.37.22 DISPLAY VIRTSTOR 735

SOURCE = WE

TOTAL LFAREA = 64M

LFAREA AVAILABLE = 32M

LFAREA ALLOCATED (1M) = 16M

LFAREA ALLOCATED (4K) = 4M

LFAREA ALLOCATED (PAGEABLE1M) = 12M

MAX LFAREA ALLOCATED (1M) = 62M

MAX LFAREA ALLOCATED (4K) = 5M

MAX LFAREA ALLOCATED (PAGEABLE1M) = 14M

NOT good, this means we broke down some of the 1MB frames

We reserve 1/8th of real on LPAR for pageable frames

- Show total LFAREA, allocation split across 4KB and 1MB size frames, what is available

How do I size LFAREA for DB2?

- LFAREA = 1.04* (sum of VPSIZE from candidate buffer pools)) + 20MB (+ OUTBUFF + 31-bit low private for DB2 11)

- **Do not oversize LFAREA**
  - LFAREA used ~ DB2 usage + JAVA heap (verbosegc traces)
  - Can't do anything about it until an IPL, if too small just means there is potential savings you could be missing out on
    - **- IRA127I 100% OF THE LARGE FRAME AREA IS ALLOCATED** = using it all
  - If for any reason RSM denies DB2 request for 1 MB frame, uses 4k instead
    - Specify INCLUDE1MAFC on LFAREA specification (OA42510)
- **Decomposing/coalescing 1MB frames into 4k frames is CPU intensive trying to maintain the LFAREA setting**

  - LFAREA ALLOCATED (4K) = 5M → Not a good sign
  - This indicates you do not have enough REAL storage needed by 4k frames, so add more real or make LFAREA smaller

Pageable 1MB frames

- By default z/OS 1.13 and up set aside 1/8th of real storage on the LPAR for pageable large frames, as long as the RSM support for it is applied
  - See RMF paging report
- PMB buffer pool control blocks can use this
- Once the 1/8th is gone we need LFAREA to allocate the rest

01 MB Frames	Fixed			Pageable		
	Total	Available	In-Use	Total	Available	In-Use
Min	0	.	0	6,064	0	6,064
Max	0	.	0	6,064	0	6,064
Avg	0	.	0	6,064	0	6,064

FlashExpress

- FlashExpress is simply flash memory (storage class memory) for z196 and up machines
- Generally speaking for performance…
  - Access to real memory ~ 1,000 machine cycles
  - Access to FlashExpress ~ 100,000..
  - Access to AUX storage ~ 1,000,000..
- So is it a substitute for REAL?? - No
  - Never intended for buffer pools or working storage, it still pages, just in 1MB chunks
  - Better than AUX for backing MAXSPACE (DUMPSERV)
    - 16GB recommended for V10 and up
- Highly recommended if CRITICAL PAGING is on
- Do not need it for pageable 1MB frames – but if those frames are paged out they become 4k
- Performance numbers and a better description:
  - http://public.dhe.ibm.com/common/ssi/ecm/en/zss03073usen/ZSS03073USEN.PDF
  - 5x reduction in dump time, 3x reduction in non-dispatchable time compared to going out to AUX

Summary – being short on REAL

- CPU cost of paging to AUX (DASD or SCM)
  - z/OS measured 20-70us
- Elapsed time if sync I/O - Roughly 2ms with current DASD
  - Buffer pools, and other in-memory pools now moved to AUX
  - DB2 thread working storage pushed out
- DUMP capture can grind LPAR to a halt
- Other complications..
  - Lose positive impact if pageable 1MB frames, broken down or moved to SCM
  - Waste CPU breaking down LFAREA to 4k frames, and rebuilding them
  - If too many fixed frames (PGFIX or z/OS fixed frames) end up with storage critical situation and address spaces marked non-dispatchable

References

- Techdoc for V10 and V11 MEMU2 with spreadsheet sample
  - http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS5279
- Subsystem and Transaction Monitoring and Tuning with DB2 11 for z/OS
  - http://www.redbooks.ibm.com/redpieces/abstracts/sg248182.html?Open
- RSM Enablement Offering (z/OS 1.13)
  - http://www-03.ibm.com/systems/z/os/zos/tools/downloads/
- Flash Express whitepaper
  - http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=ZSS03073USEN&appname=wwwsearch