The Present and Future of Large Memory in DB2 for z/OS

Dave Levish

IBM DB2 for z/OS development

Session 15768   August 7, 2014

#SHAREorg

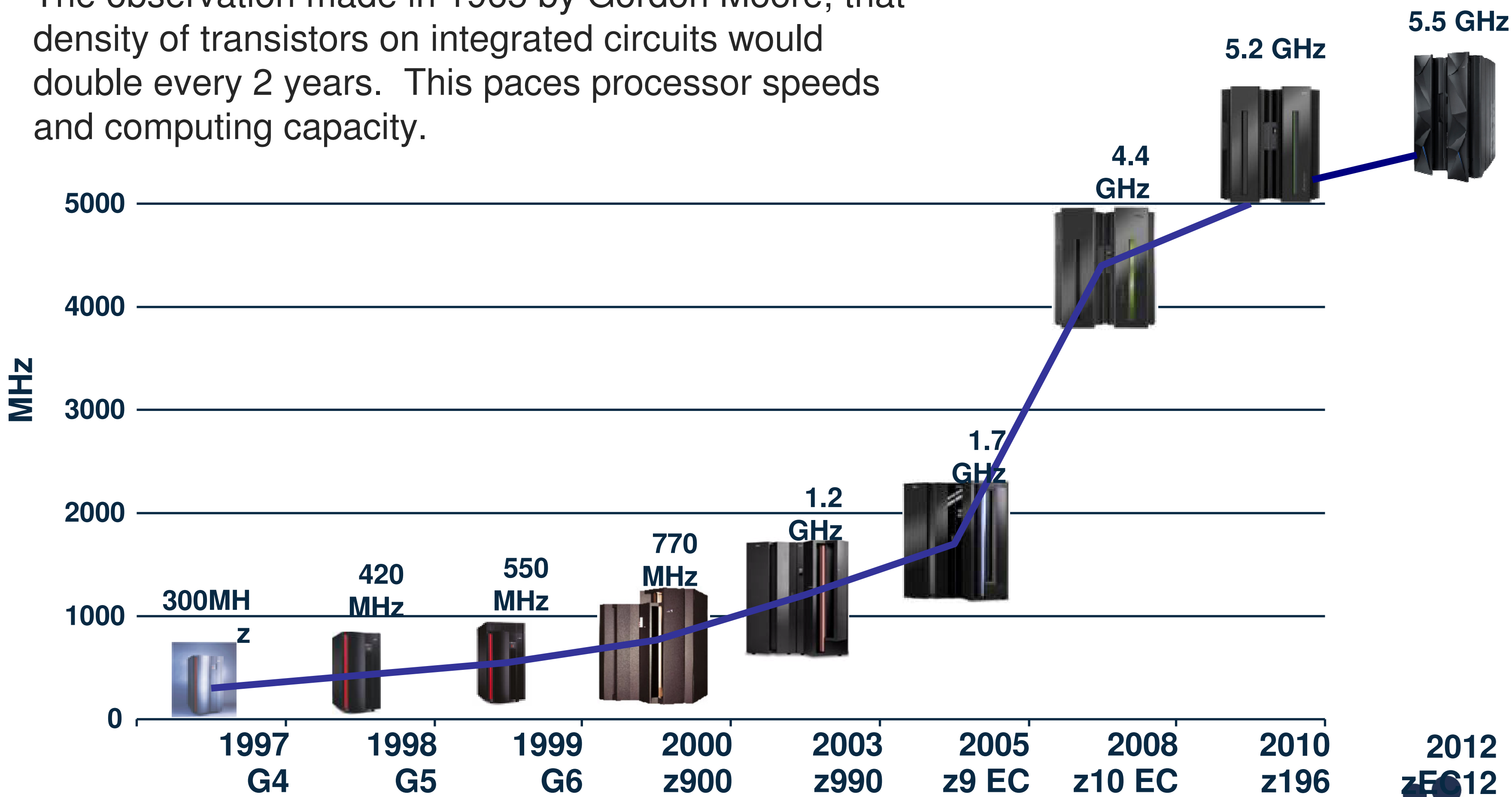
SHARE in Pittsburgh 2014

Agenda

- Moore's Law and Recent Mainframe Evolution
- Real storage management
- Storage contraction
- DB2 current exploitation of large memory
  – Buffer pools
  – High Performance DBAT and RELEASE(DEALLOCATE)
  – EDM
  – Other Exploiters
- Large Memory Futures

Moore's Law and Mainframe Evolution

The observation made in 1965 by Gordon Moore, that density of transistors on integrated circuits would double every 2 years. This paces processor speeds and computing capacity.



Moore's Law and Mainframe Evolution

- IBM set world microprocessor records with the with the zEC12 clocked at 5.5GHZ and 2.75 billion transistors in 597mm^2. Unfortunately this won't last forever.

"It can't continue forever. The nature of exponentials is that you push them out and eventually disaster happens."

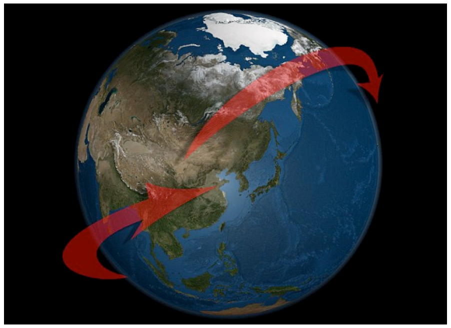
– Gordon Moore, April 2005

- The 'raw speed' trend is flattening so we must turn our attention to other means to get performance.
- Emphasis will turn to software to more intelligently use hardware resources for significant performance gains.

Where to Gain Performance?

- Cache and memory latency on a hypothetical modern server
  - L1 Cache - 1 machine cycle
  - L2 Cache - 4 machine cycles
  - L3 Cache - variable 10's of machine cycles
  - L4 Cache - variable 100's of machine cycles
  - Real Memory - 1000 machine cycles
  - Disk - depends on device type 1,000,000 - 10,000,000+ cycles

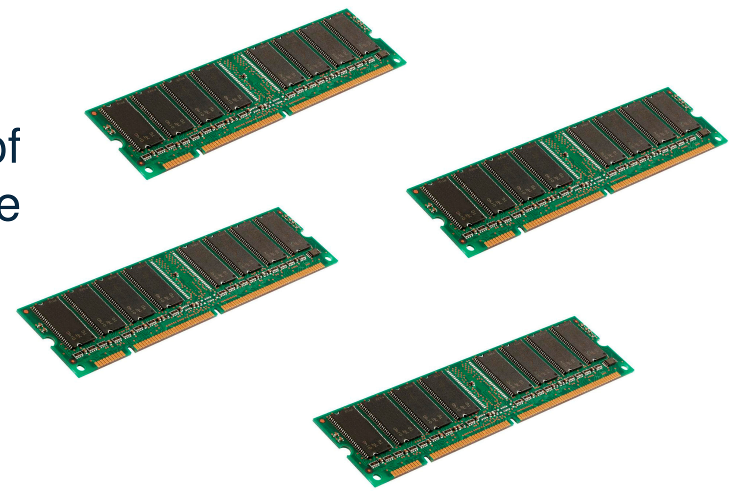
A baby step
vs.
a walk
around the
equator



Enterprise Machines with Large Main Memories

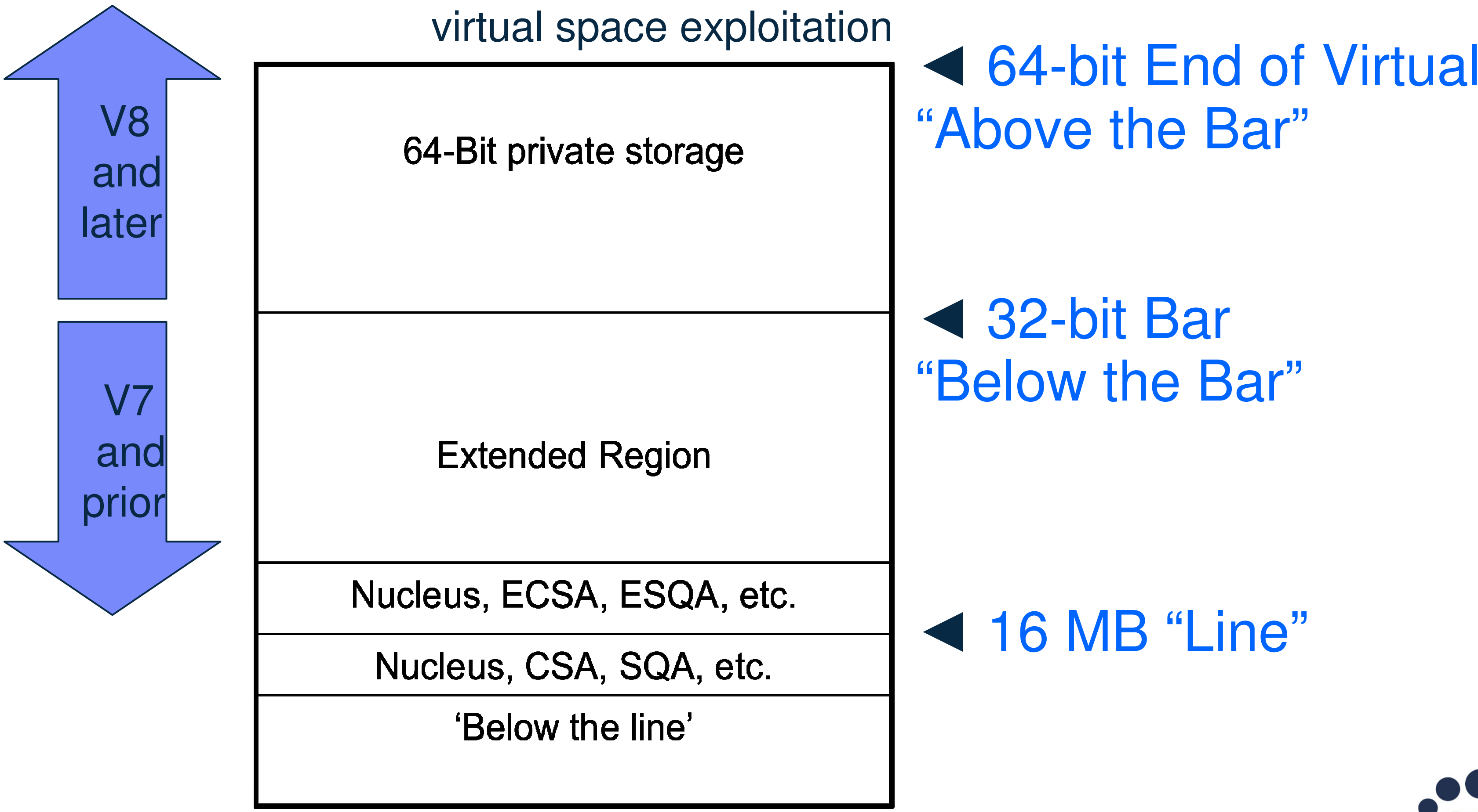
- zEC12 and z196 support up to 3T of real storage with a limit of 1T per LPAR (z10 1.5T and 1T per LPAR)
- This is expected to increase significantly in future generation machines
- Memory is a one-time charge item that can provide cost savings across the stack

Terabytes of available storage



DB2 Can Exploit Large Memory (Virtual and Real)

DB2 10 sets the stage for vast virtual space exploitation



Are Vast Virtual Spaces Safe?

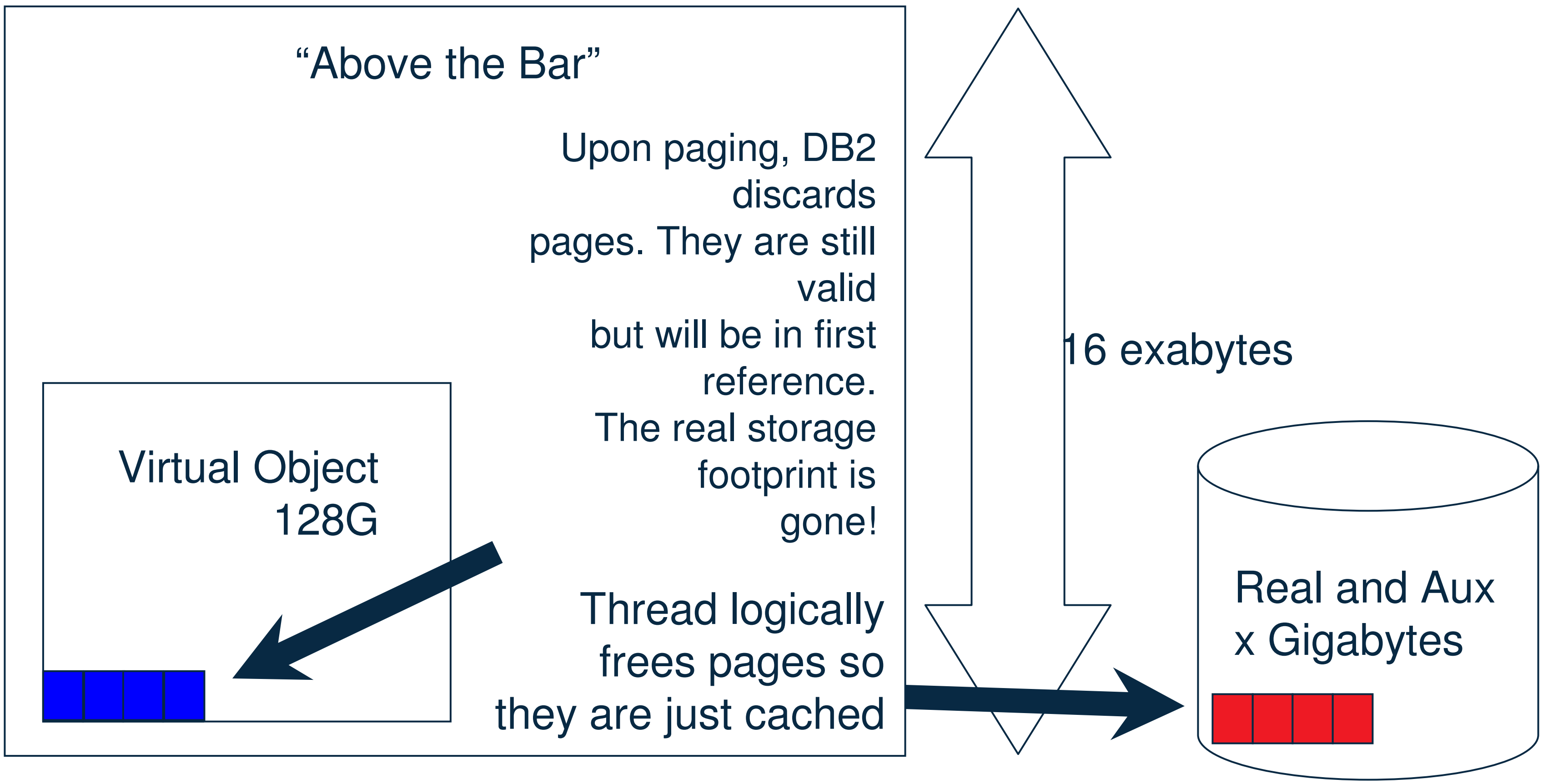
Performance Degrades as Real Storage is Consumed

Once all aux is consumed, the LPAR goes into a wait state

Nothing to Fear - Real Storage Management

- New system configuration parameters with apar PM24723/PM49816 for DB2 10 to define real storage boundaries and how aggressively DB2 should strive to remain in the defined boundary (Note: z/OS apar OA35885/OA37821 are required to enable the full functionality).
- ZPARM REALSTORAGE_MAX defines the sandbox
  - Amount of real and aux in GB a given DB2 subsystem is allowed to consume
- The frequency of contraction mode can also be controlled by system parameter REALSTORAGE_MANAGEMENT. ZPARM REALSTORAGE_MANAGEMENT defines the behavior within the sandbox.
- REALSTORAGE_MANAGEMENT options include:
  - **OFF** - only regulate when critical condition is detected
  - **ON** - operate with the smallest real footprint possible
  - **AUTO (the default)** - contract real footprint when paging (Note: apar PM88804 reduces discards in AUTO or OFF mode, recommended apply with PM86952. APAR PM99575 (PE) further refines the logic, PI14696 fixes the PE.)

What Does Real Storage Contraction Mean?



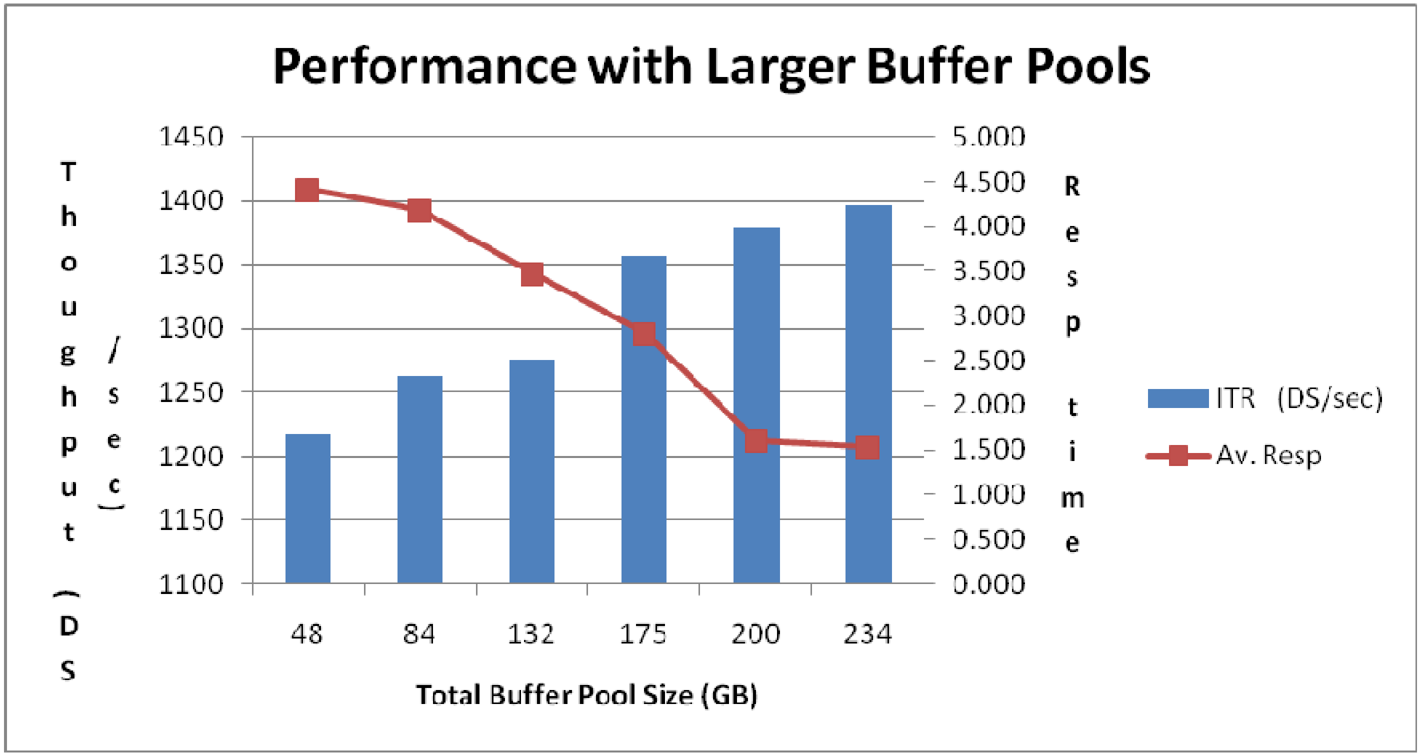
DB2 Exploitation

- **Buffer Pools**
- High Performance DBAT and RELEASE(DEALLOCATE)
- EDM
- Other Exploiters

Buffer Pools

- DB2 buffer pools serve as a data cache for DB2 objects
- Each buffer represents a page from a given object. Buffers come in 4K, 8K, 16K, and 32K page sizes (not 1M or 2G! explanation is forthcoming)
- The key to buffer pool performance is to retain buffers that are re-referenced. If a workload does not re-reference data, buffering will be of limited or no benefit.
- Buffer pool tuning is beyond the scope of this talk, but a basic tuning strategy is to increase the buffer pool size to buffer more data.
  - Large buffer pools today are in the 10's of G range
  - All in-service releases of DB2 support a total of 1T of buffer pool space (for smaller installations it is limited to 2x amount of real available to the LPAR)

What if We Had Huge Buffer Pools?



- DB2 control structures and algorithms need analysis for scalability
- CF considerations -- how about the GBPs?
- May reduce the need for micro-management tuning

Exploiting Memory with Large Buffer Pools

- **In Memory Objects**
  - DB2 10 supports PGSTEAL(NONE)
  - All objects on first reference are read completely into the buffer pool
  - Object needs to fit else performance problems may ensue
  - As buffer pools get bigger with large main memories, this can be more easily exploited
- **Page Fixed Buffer Pools**
  - Page fixing is not new, but is best suited for a large memory environment
  - When a buffer is read or written to disk, it must be fixed in real memory
  - Long term page fixing the buffer pool requires sufficient memory to fix all frames. This is a good opportunity for performance gains by eliminating the need to fix and unfix frames.
  - Future thoughts are that it will be standard to page fix buffer pools (currently required for large frame exploitation)

What About Large Page Sizes and Buffer Pools?

- z10+ support 1M pages and zEC12 introduces 2G pages
- ***What is this all about?***

Answers 1 and 2 are:

"This has nothing to do with data page sizes!"

Answer 3 is:

"This relates solely to the computing environment on these machines. It is an optimization for translating a virtual address into a real address"

What About Large Page Sizes and Buffer Pools?

Program refers to virtual address 0x5F001800

Look in processor cache TLB for real frame address. In this case we look for page entry 0x5F001000

Miss

Resolve via page table structure

Hit

Get 4K page in memory

TLB entries will represent a full 1M range. We will be looking for 0x5F000000

Will get 1M page in memory. DB2 exploits only fixed pages so it will be in memory already!

More efficient page table lookup with entries representing a full 1M. We will be looking for 0x5F000000

What About Large Page Sizes and Buffer Pools?

- **Exploitation Summary**
  - DB2 10 supports fixed 1M frames
  - DB2 11 supports fixed 2G frames
  - Enabled via LFAREA in IEASYSxx
  - Observed 1-4% with DB2 10 and 1M buffer pools
  - Smaller additional improvement with DB2 11 and 2G buffer pools

- **Futures?**
  - Possible exploitation of 1M pageable frames
  - Page fixed large frame buffer pools will be common?
  - Larger frame sizes?

DB2 Exploitation

- Buffer pools
- **High Performance DBAT and RELEASE(DEALLOCATE)**
- EDM
- Other Exploiters

RELEASE(DEALLOCATE)

- RELEASE is a BIND option that tells DB2 how to handle the caching of package locks and structures. The default is COMMIT and DB2 will free package structures upon commit. The alternative is DEALLOCATE where package structures will persist until full thread deallocation.
- This can be an opportunity for performance for:
  - Long running batch jobs that COMMIT frequently
  - Thread reuse (e.g., CICS protected threads, JCC T2)
- Why isn't RELEASE(DEALLOCATE) used pervasively?
  - Virtual storage constraints (mainly v9 and earlier)
  - DDF workload
  - The need to break in for BIND/DDL activity

High Performance DBAT

- Re-introducing RELEASE(DEALLOCATE) in distributed packages
  - Could not break in to do DDL, BIND
  - V6 PQ63185 to disable RELEASE(DEALLOCATE) on DRDA DBATs
- High Performance DBATs reduce CPU consumption by
  - RELEASE(DEALLOCATE) to avoid repeated package allocation/deallocation
  - Avoids processing to go inactive and then back to active
  - Bigger CPU reduction for short transactions
- Using High Performance DBATs
  - Stay active if there is at least one RELEASE(DEALLOCATE) package exists
  - Connections will turn inactive after 200 times (not changeable) to free up DBAT
  - Normal idle thread time-out detection will be applied to these DBATs
  - Good match with JCC packages
  - Not for KEEPDYNAMIC YES users

Can I Break In?

V10 DDF Threads:

- To enable
  - Rebind packages with RELEASE(DEALLOCATE)
  - -MODIFY DDF PKGREL(BINDOPT)
- To disable
  - Wait 200 commits
  - -MODIFY DDF PKGREL(COMMIT) and this only effects newly allocated DBATs!

V10 Other Threads:

- No solution

Can I Break In? ...Continued

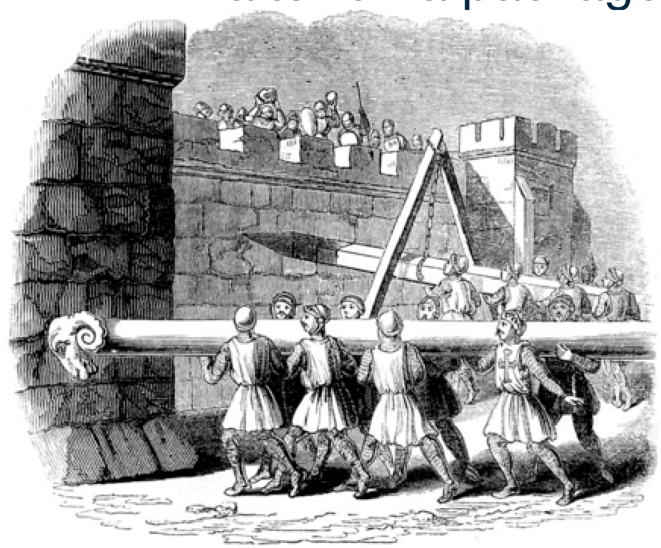
V11 DDF Threads:

- To enable same as V11
- To disable
  - Automatically done on next COMMIT if waiter on a package lock

V11 Other Threads:

- Automatically done on next COMMIT if waiter on a package lock
- Idle threads? PM96004!

  (March 2014)



DB2 Exploitation

- Buffer pools
- High Performance DBAT and RELEASE(DEALLOCATE)
- **EDM**
- Other Exploiters

EDM

- The EDM pool is another cache repository for DB2 storage. It caches dynamic statements, DBDs, and skeleton packages/plans.
- Controlled by a number of ZPARMs EDMDBDC, EDMPOOL, EDMSTMTC, EDM_SKELETON_POOL
- Logically the bigger the pool, the greater the chance for a cache hit
  - Less I/O to load packages
  - Fewer full prepares
  - Less chance of EDM full
- The max size for these parms in v10 is 2G. In v11 these were increased to 4G.
- **Is the future simply bigger is better?**
  - We may need to restructure some of the internals to properly scale higher
  - Page fixing to keep critical cache in memory?
  - Large page exploitation for TLB access benefits?

DB2 Exploitation

- Buffer pools
- High Performance DBAT and RELEASE(DEALLOCATE)
- EDM
- **Other Exploiters**

Other Memory Exploiters

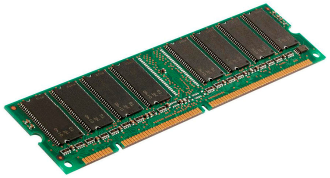
- **v11 Storage Awareness**
  - ***Parallelism***
    - When spawning child tasks, runtime will evaluate the storage environment. If we are paging or cannot allocate many new agents due to ECSA constraints, we will cut the amount of parallelism.
  - ***Sort***
    - Will evaluate the storage environment to determine if an in-memory sort can be conducted.
  - ***Sparse Index***
    - When attempting to allocate MXDTCACH storage, reduce the request if real storage is limited.

Going forward …

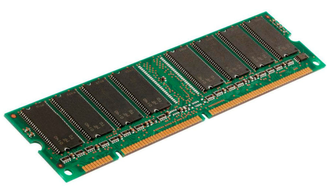
- **Storage Awareness Futures**
  - Current strategies are more green/red in nature -- “Everything looks OK, go for it!” Probably fall apart with high degree if parallel inquiries.
  - Developing strategies to determine the size of available “slush” storage and give it to threads in controlled fashion
  - Reduction in ZPARM settings to control storage usage (e.g., MXDTCACH)

- **Continued movement of storage ATB**
  - Storage moving to 64-bit shared and common to avoid cross-memory moves
  - Long term goal: 24-bit and 31-bit virtual will *never* limit scaling

Futures TBD?



- Large page support for all of DB2
- RELEASE(DEALLOCATE) only limited by appropriateness of the option and not by real or virtual storage constraints
- Huge buffer pools with many objects completely in the pool
- EDM micro-managing disappears
- Other ZPARMs controlling storage sizes disappear (maybe not REALSTORAGE_MAX or REALSTORAGE_MANAGEMENT)
- In memory everything? DB2 will likely not be an “in-memory” database any time soon due to the size of the data we handle but hybrid solutions may be available for exploitation.
- Performance improvements continue through exploitation of large real memory



Dave Levish

IBM Silicon Valley Laboratory

levish@us.ibm.com

Session 15768

The Present and Future of
Large Memory in DB2 for z/OS

#SHAREorg

SHARE in Pittsburgh 2014