z/OS Virtual Memory

Session Number: 24662
March 13th, 2019

Elpida Tzortzatos
Distinguished Engineer
z/OS Core Design and Analytics Lead for IBM Z
elpida@us.ibm.com
Agenda

• Memory Management Basics
• VSM Overview
• VSM DIAGxx Options
• VSM Health Checks
• Large Pages and Their Value
Z/OS BASIC MEMORY MANAGEMENT CONCEPTS
z/OS Memory Types

• There are three z/OS memory types used to process system and user/application storage requests:
  – Real frames: the physical main memory.
  – Auxiliary: paging dataset slots and storage-class memory (SCM) blocks.
  – Virtual pages: created through dynamic address translation (DAT) for multiple address spaces.

(This presentation uses the traditional term “storage” and more common term “memory” interchangeably.)
Processor Storage Overview

- Processor Cache
- Main Memory
- Storage-Class Memory
- Paging Datasets

On the box

Real Storage

External

Auxiliary Storage

Increasing Speed

Copyright© 2017 by SHARE Inc. Except where otherwise noted, this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 license. http://creativecommons.org/licenses/by-nc-nd/3.0/
Virtual Memory

- Pages of data in virtual is backed by real or auxiliary storage.
- Contiguous in virtual are typically not contiguous when backed.
- In some cases a page can be backed both in real and aux.
- DAT tables translate virtual addresses to real addresses.
The Memory Managers

• The Virtual Storage Manager (VSM)
  – Receives the requests to obtain and release virtual storage.
  – Keeps track of the allocated and free virtual areas for the different types of storage below the 2G bar.

• The Real Storage Manager (RSM)
  – Backs the virtual storage pages with real storage frames.
  – Keeps track of the various frames needed for the different areas of virtual storage.
  – Manages 64-bit “above the bar” virtual.

• The Auxiliary Storage Manager (ASM)
  – Reads and writes pages to/from auxiliary storage.
Virtual/Real Sizes

Theoretical Max 16 EB (2**64)

- **64 bit addressable**
- **Current maximum real storage supported by z/OS**
- **31 bit addressable**
- **24 bit addressable**

- **4 TB**
- **“The Bar” 2 GB**
- **“The Line” 16 MB**
- **Low memory address: 0**

(Not to scale)
Full Virtual Memory Map

- Three addressing ranges:
  - 64-bit: 2G-16EB “Above the Bar”
  - 31-bit: 16MB-2G “Above the Line”
  - 24-bit: 0-16MB “Below the Line”

- Each has both private and common areas.
Common, Private, Shared

- Private storage is unique to each address space.
- Common storage is global to every address space.
- Share storage access can be granted to multiple address spaces.
- The Prefix Save Area (PSA) is a special area unique to each processor.
Memory Attributes

• **Viewability:** Private, Common, Shared.
• **Type:**
  – Pageable – Default for most cases. Can be backed in real or auxiliary.
  – Fixed – Use if doing I/O, running in a FLIH, or obtaining real address.
  – DREF – Use if disabled but don’t need fixed.
• **Residency:**
  – Virtual: Controlled by first part of LOC option on STORAGE OBTAIN for 24 and 31 bit virtual. E.g. LOC=(31,xx).
    • IARV64/IARST64/IARCP64 are always 64-bit virtual.
  – Real: Controlled by second part to LOC option, e.g. LOC=(xx,64).
    • Only enforced if fixed.
    • LOC=(31,64) should be used by most applications.
    • 64-bit services (IARV64 et al) use 64-bit real backing.
• **Ownership:** Task, Address Space, System.
• **Others:** Key, Fetch Protection, Executable, and more.
Virtual Above and Below “The Bar”

• Below 2G:
  – Storage requests by using the following services:
    • Getmain/Freemain, Storage Obtain/Release, CPOOL (cell pool).
    • Allocate in 8 byte increments.

• Above 2G:
  – Storage requests by using the following services:
    • IARV64 (GETSTOR/GETCOMMON/GETSHARED), IARCP64 (cell pool), IARST64.
    • IARV64 allocates in 1MB increments. Use IARST64 and IARCP64 to obtain smaller increments.
VSM: VIRTUAL STORAGE MANAGER
z/OS Memory Managers: VSM

• Virtual Storage Manager
• Address Space-centric view of the system and processes.
• Objectives:
  – Control the allocation/deallocation of 31-bit virtual storage addresses.
  – Efficiency – minimum overhead per request.
• Associate a storage protection key with each virtual storage block requested.
• Maintain storage use information by generating SMF records.
VSM Services

VSM:

• GETMAIN – Allocate 31-bit virtual.
• FREEMAIN – Free 31-bit virtual.
• STORAGE – Newer service to allocate/free 31-bit virtual.
• CPOOL – 31-bit cell pool service.
31-Bit Address Space Memory Map

- 31-bit (below the bar) virtual is managed by VSM.
- Sizes of CSA, ECSA, SQA, ESQA are specified via IEASYSxx parmlib member.
VSM Storage Management Rules

- z/OS manages 31-bit virtual storage through the use of subpools designed to accommodate a variety of storage needs.
- Storage is allocated or assigned to a subpool in one page (4K) multiples.
- Storage belonging to different subpools cannot occupy the same page.
- Storage with different storage keys cannot occupy the same page.
- Storage belonging to different TCBs cannot occupy the same page.
Private Subpool Attributes

- **Subpool numbers**: 0 – 255
- **Storage protection**: Keys 0 – 15
- **User Region (AKA Low Private)**:
  - Subpools 0 – 132, 250 – 252
  - TCB-related
  - Keyed storage
  - Unauthorized
  - General purpose subpools
- **High Private**:
  - Subpools 229, 230, 249
  - TCB-related
  - Keyed storage
  - Authorized
  - Special authorization application storage needs
- **ELSQA/LSQA**:
  - Subpool 255 (mainly)
  - Fixed, key 0 storage
  - Address space-related, not TCB-related

See MVS Diagnosis: Reference, Chapter 9, for additional subpool information
Virtual Storage Areas: Common

- **Common (Global) Storage**
  - Shared by all address spaces
    - Contents of a particular virtual address is the same for all address spaces.
    - Accessible using the DAT tables for any address space.

- **Different (separate) areas of common storage.**
  - **Prefixed Save Area (PSA)** – Maps fixed hardware and software locations for the related processor
  - **Common Service Area (CSA)**
    - Pageable and fixed data areas
    - Some load to global modules
  - **Link Pack Area (LPA)**
    - Pageable Link Pack Area (PLPA)
      - Built at IPL time from libraries specified in LPALSTxx or PROGxx.
      - Contains SVC routines, access methods, and other read-only system programs, some select read-only re-enterable user programs that can be shared among users of the system, some frequently used refreshable SYS1.LINKLIB and SYS1.CMDLIB modules.
    - **Modified Link Pack Area (MLPA)**
      - Built at IPL time as specified in IEALPAxx.
  - **System Queue Area (SQA)**
    - Contains tables and queues relating to the entire system
    - When not enough SQA storage available, storage may be taken from CSA
  - **Nucleus (NUC)**
    - Built at IPL time
    - Read-only nuc, Read-write nuc
Virtual Storage Areas: Private

- Private (Local) Storage
- Not shared across address spaces (each address space has its own)
  - Content of a particular virtual address not same in another address space
- Different (separate) areas within the private area
  - System Region
    - GETMAINs for tasks running under RCT
  - ‘Low-end’ of private area
    - User Region
  - ‘High-end’ of private area
    - Local System Queue Area (LSQA)
    - Area for system tables and queues
      - associated with the users address space
    - Scheduler Work Area (SWA) – Contains control blocks for Initiator/Scheduler
    - Subpools 229 and 230
    - Storage obtained in requestor’s storage protect key
    - Used for control blocks only obtained by auth programs with appropriate key
Storage Key Protection

• Storage keys ensure only programs with the right permissions have access to storage.

• If a program attempts to access a page in the wrong key an ABEND results.
Storage Key Details

• Pages have a storage key and fetch protect status.
• Programs run in a PSW key.
• Programs can only read and write to pages that have a storage key matching their PSW key, with the following exceptions:
  – Programs running with PSW key 0 can read and write to any key.
  – All programs can read and write key 9 storage.
  – All programs can read any storage that is not fetch protected.
• System keys are 0-7.
• User keys 8-15.
Changing Keys

- In addition to PSW key, each program has a PSW key mask (PKM) in control register 3.
  - This is initially set to match the program's PSW-key but can be changed on attach or PC routines.
- Programs can only change to a key defined by their PKM.
- Additionally authorized programs (key 0-7, supervisor state, or APF authorized) can change to any key.
- A program must be Supervisor State and Key 0 to change the storage key of a page.
EXECUTABLE=NO

• Exploiting of z14’s Instruction Execution Protection.
• New keyword EXECUTABLE.
• Specify on STORAGE OBTAIN & RELEASE (and IARV64).
• EXECUTABLE=YES is default and matches old behavior.
• EXECUTABLE=NO indicates it will not be executed.
  – ABEND0C4-4 will result if execution is attempted.
• Helps prevent security exposures that rely on injecting executable code into data buffers.
  – Best to use EXECUTABLE=NO if you know it's not executable.
• Supported for non-LSQA private subpools.
VSM DIAGXX AND HEALTH CHECKS
VSM DIAGxx Statements

- **VSM TRACE** … – Enable GETMAIN, FREEMAIN, STORAGE (GFS) trace.
  - Many options for filtering.
- **VSM TRACK** … – Enable the Common Storage Tracker.
- **VSM CHECKREGIONLOSS(256K,10M)** – Provides way to recycle initiators when their storage becomes fragmented.
- **ALLOWUSERKEYCSA(NO|YES)** – YES allows user-key CSA. Default and recommendation is NO.
  - v2.3 is last release to support YES.
- **VSM BESTFITCSA(NO|YES)** – YES may avoid CSA fragmentation. NO is default.
- **TRAPS** … -- Various other diagnostic functions (mostly undocumented).
Common Storage Tracker

• Tracks owners of currently obtained SQA/ESQA and CSA/ECSA storage.
  – Address and length of storage.
  – ASID, jobname, and PSW address of owner.
  – Time and date of GETMAIN.

• Activated via DIAGxx parmlib member
  – Can be activated/deactivated at any time.
  – DIAGxx: VSM TRACK CSA(ON) SQA(ON)
  – Set DIAG=XX
### Viewing Tracker Data

- **VERBX VSMDATA 'OWNCOMM DETAIL'**
- Formats a detailed report of common storage usage.

<table>
<thead>
<tr>
<th>ASID</th>
<th>Job Name Id</th>
<th>St T Address</th>
<th>Length</th>
<th>Ret Addr MM/DD/YYYY HH:MM:SS</th>
<th>CAUB</th>
<th>GQE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0028</td>
<td>IBMUSER</td>
<td>TSU00016 Ac C</td>
<td>00B44400</td>
<td>00000088 23FCC9D6</td>
<td>09/12/2005 16:45:45</td>
<td>0241FEB0</td>
</tr>
<tr>
<td>Data</td>
<td>----&gt; 23FAF2D8 23DB5D80 E3606000 00000088</td>
<td>00000088</td>
<td>0241FEB0</td>
<td>01DDD6A0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0028</td>
<td>IBMUSER</td>
<td>TSU00016 Ac S</td>
<td>00FC5018</td>
<td>00000030 00CA5024</td>
<td>09/12/2005 16:45:45</td>
<td>0241FEB0</td>
</tr>
<tr>
<td>Data</td>
<td>----&gt; 00000000 00000000 00F97B80 00000028</td>
<td>00000000</td>
<td>0241FEB0</td>
<td>01E35AF0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0028</td>
<td>IBMUSER</td>
<td>TSU00016 Ac S</td>
<td>022E7A28</td>
<td>00000018 039E5364</td>
<td>09/12/2005 16:45:57</td>
<td>0241FEB0</td>
</tr>
<tr>
<td>Data</td>
<td>----&gt; E2E8E2F1 40404040</td>
<td>00000000</td>
<td>0241FEB0</td>
<td>01E43C88</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0028</td>
<td>IBMUSER</td>
<td>TSU00016 Ac S</td>
<td>02546000</td>
<td>00000060 00D3D1AE</td>
<td>09/12/2005 16:45:58</td>
<td>0241FEB0</td>
</tr>
<tr>
<td>Data</td>
<td>----&gt; D1E2C1C2</td>
<td>00000000</td>
<td>0241FEB0</td>
<td>01E35C40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0028</td>
<td>IBMUSER</td>
<td>TSU00016 Ac S</td>
<td>02546000</td>
<td>00000060 00D3D1AE</td>
<td>09/12/2005 16:45:58</td>
<td>0241FEB0</td>
</tr>
<tr>
<td>Data</td>
<td>----&gt; E2E3D8C5 F500005C</td>
<td>00000000</td>
<td>0241FEB0</td>
<td>01E35C40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0028</td>
<td>IBMUSER</td>
<td>TSU00016 Ac S</td>
<td>025C1578</td>
<td>00000048 2466F036</td>
<td>09/12/2005 16:45:57</td>
<td>0241FEB0</td>
</tr>
<tr>
<td>Data</td>
<td>----&gt; D3D4C1C2</td>
<td>00000000 7FF4EF60 7FF4EF60</td>
<td>0241FEB0</td>
<td>01E35BC8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
VSM Health Checks

- **VSM_CSA_CHANGE** – Warn if CSA size is different this IPL.
- **VSM_CSA_THRESHOLD** – Warn if CSA is getting full.
- **VSM_SQA_LIMIT** – Warn if SQA size is too small.
- **VSM_PVT_LIMIT** – Warn if private area is too small.
- **VSM_CSA_LIMIT** – Warn if CSA size is too small.
- **VSM_SQA_THRESHOLD** – Warn if SQA is getting too full.
  - Note that some installation normally overflow SQA into CSA.
- **VSM_ALLOWUSERKEYCSA** – Warns if using insecure “YES” option.
- **VSM_CSA_LARGEST_FREE** – Warns if CSA is becoming fragmented.
RSM AND LARGE PAGES
Large Pages: What Are They?

4K Pages:
• Page table points to 4K frames.
• Uses up to 256 TLB entries per segment.

1M Pages:
• No page table.
• Segment table points to 1M frames.
• Uses only one TLB entry per segment.
Importance of Exploiting Large Pages

- Problem: Performance degradation due to increased TLB (Translation Lookaside Buffer) misses.
  - Over the past years application memory sizes have dramatically increased due to support for 64-bit addressing in both physical and virtual memory.
  - TLB sizes have remained relatively small due to low access time requirements and hardware space limitations.
  - Therefore TLB coverage today represents a much smaller fraction of an applications working set size leading to a larger number of TLB misses.
  - Applications can suffer a significant performance penalty resulting from an increased number of TLB misses as well as the increased cost of each TLB miss.
Importance of Exploiting Large Pages

• Benefits: Increase TLB coverage without proportionally enlarging the TLB size by using large pages:
  – Large pages will provide exploiters with better TLB coverage, and therefore better performance by decreasing the number of TLB misses that an application incurs.
  – Less time spent converting virtual addresses into physical addresses.
  – Less page faults as whole 1MB is backed at once instead of for each 4k page.
  – A large page is a memory page larger than an ordinary 4K base page.
  
  z/OS supports the following 2 large page sizes:
  – 1MB.
  – 2GB.
Key Exploiters of Large Pages

• Db2:
  – 1M (V10) and 2G (V11) page frame size for PGFIX(YES) pools.
  – FRAMESIZE parameter in Db2 11.
  – IBM evaluation shows:
    • 1-3% improvement from 4K frames to 1M frames (zEC12).
    • 1-3% improvement from 1M frames to 2G frames (z13).

• Java™:
  – Pageable 1MB is default as of Java 7.
  – Use option -Xlp to control the page size such as to use 2GB pages.
  – WAS Day Trader benchmarks showed up to an 8% performance improvement.

• z/OS itself uses large pages in many places when they are available.
Large Pages in 31-bit Virtual

• The most significant benefit of large pages comes from applications that use vast amounts of data (e.g. 64-bit).
  – 31-bit applications can gain benefits as well if they frequently touch many pages together.

• 31-bit private via STORAGE OBTAIN and CPOOL:
  – LOC=(31|EXPLICIT,PAGEFRAMESIZE1MB).
  – Indicates to back pages in 64-bit real and prefer 1MB frames.
  – Only 0-127, 129-132, 240, 244 or 250-252 (Private low, pageable).

• Data spaces via DSPSERV:
  – PAGEFRAMESIZE=1M on CREATE (changes default to BACK=64).
  – BLOCKS does not need to be multiple of 256.
Large Frame Areas (Pre-v2.3)

• 1MB LFAREA - Large Frame Area
  – Fixed storage each frame is 1MB
  – Defined by LFAREA (1M) (IEASYSxx)
  – Included in Available Frame Count when INCLUDE1MAFC=YES

• PLAREA – Pageable Large Area
  – Pageable storage each frame is 1MB.
  – System defined size approximately (online storage at IPL time)/8 if enough storage is left above 2G after LFAREA and Quad Area are defined.
  – Allocated on SCM capable machines.
  – Can overflow in the LFAREA.

• 2GB LFAREA Large Frame Area
  – Fixed storage each frame is 2GB
  – Defined by LFAREA (2G) (IEASYSxx)
  – Not included in Available Frame Count – used only for 2G requests
  – Reserved for specific 2GB memory objects
Large Frame Areas in z/OS v2.3

• 1MB LFAREA - Large Frame Area
  – No longer physical range.
  – Managed dynamically in non-reconfigurable memory above 2G bar.
  – Capped by LFAREA (1M= ) (IEASYSxx)
  – INCLUDE1MAFC=NO is ignored.

• PLAREA – Pageable Large Area
  – No longer physical range.
  – Managed dynamically in non-reconfigurable memory above 2G bar.
  – No cap.

• 2GB LFAREA Large Frame Area
  – Unchanged in v2.3.
Sizing LFAREA for Fixed 1MB

• Calculate how much you will need for Db2 buffer pools, JVM Heaps, etc.
• Best to add additional memory corresponding to the specified LFAREA size to existing system memory.
• Have enough 4K frames to handle your 4K workload needs (both pageable and fixed).
  – Have enough 4K frames above the bar to avoid RSM breaking down free 1M frames and paging or page movement for 4K page Fixes
    • Include RSM needs for memory mapping - 1/64 total online real at IPL (4g for 256G system)
    • System address space memory usage.
    • Include enough spare 4K frames for taking dumps quickly.
• Doc APAR OA34024 gives some guidance on how to size the LFAREA.
Sizing LFAREA for Pageable 1M (Pre-v2.3)

- Pageable Large Pages overflow into the LFAREA when PLAREA is depleted.
- If you want to ensure your system has enough Pageable Large Pages specify additional memory for the LFAREA to also accommodate Pageable Large Pages.
- The following z/OS System Console command D VS,LFAREA can be used to display LFAREA usage by 1MB fixed, 4K, and 1MB pageable large pages.

RESPONSE=PA1  
IAR019I 17.13.02 DISPLAY VIRTSTOR 175  
SOURCE = OM  
TOTAL LFAREA = 120422M, 0G  
LFAREA AVAILABLE = 0M, 0G  
LFAREA ALLOCATED (1M) = 0M  
LFAREA ALLOCATED (4K) = 0M  
MAX LFAREA ALLOCATED (1M) = 0M  
MAX LFAREA ALLOCATED (4K) = 0M  
LFAREA ALLOCATED (PAGEABLE1M) = 120422M  
MAX LFAREA ALLOCATED (PAGEABLE1M) = 120422M  
LFAREA ALLOCATED NUMBER OF 2G PAGES = 0  
MAX LFAREA ALLOCATED NUMBER OF 2G PAGES = 0
**Right-Sizing the LFAREA**

**RMF Monitor III - STORF**

<table>
<thead>
<tr>
<th>Command</th>
<th>RMF V2R1 Storage Memory Objects</th>
<th>Line 1 of 306</th>
<th>Scroll</th>
<th>2SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samples: 90</td>
<td>System: R7F Date: 03/02/16 Time: 14:36:30 Range: 90 Sec</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**System Summary**

<table>
<thead>
<tr>
<th>MemObj</th>
<th>Frames</th>
<th>1MB MemObj</th>
<th>1MB Fixed</th>
<th>1MB Pageable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared</td>
<td>1</td>
<td>Shared 677</td>
<td>Total 1</td>
<td>Total 104K Initial 23704</td>
</tr>
<tr>
<td>Common</td>
<td>101</td>
<td>Common 125K</td>
<td>Common 1</td>
<td>Common 5 Dynamic 0</td>
</tr>
<tr>
<td>%Used</td>
<td>5.6</td>
<td>%Used 0.0</td>
<td>%Used 0.2</td>
<td></td>
</tr>
</tbody>
</table>

---

**Service Memory Objects**

| Jobname | Class | ASID | Total Comm Shr 1MB Fixed Pgble Total Comm Shr |
|---------|------|------|------------------------------------------------|---------------------------------|
| XGLOGR  | S    | 0028 | 374 0 0 0 0 4 379M 0 0 |
| SMSMOSSC | S    | 0009 | 49 0 0 0 0 68.0M 0 0 |
| JES2AUX | S    | 0077 | 33 32 0 0 0 144M 136M 0 |
| TRACE   | S    | 0004 | 32 0 0 0 0 35.0M 0 0 |
| SMSMOSSC | S    | 0008 | 27 0 0 0 0 46.0M 0 0 |
| SDFSAUX | S    | 0066 | 15 0 0 0 0 416M 0 0 |
| *MASTER* | S    | 0001 | 12 9 0 1 5 70.0M 64.0M 0 |
| GRS     | S    | 0007 | 12 1 0 0 0 236G 1024K 0 |
| TCP342  | S    | 0057 | 10 2 0 0 0 2591M 2580M 0 |
| APPC    | S    | 0035 | 9 0 0 0 0 12.0M 0 0 |
| SMSVSAM | S    | 0010 | 8 1 0 0 0 33.0M 1024K 0 |
| DMVS    | S    | 0017 | 8 1 0 0 0 7 931M 1024K 0 |
| H2SPROC | S    | 0020 | 7 1 1 0 0 820M 1024K 1024K |

---

Monitor and adjust LFAREA parms based on workload.

Pre-v2.3, track Pagable large overflow into LFAREA. Average number of 1 MB frames in the LFAREA that were used to satisfy 1 MB pageable page request.

Note: all system address space large page usage.
### z/OS v2.3 Large Page Display

<table>
<thead>
<tr>
<th>Display</th>
<th>Total Size</th>
<th>Available</th>
<th>In-Use</th>
<th>Max In-Use</th>
<th>Fixed Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGEABLE 1M STATISTICS</td>
<td>4824.0MB</td>
<td>4585.0MB</td>
<td>3.0MB</td>
<td>3.0MB</td>
<td>1.0MB</td>
</tr>
<tr>
<td>LFAREA 1M STATISTICS - SOURCE = IEASYS23</td>
<td>64.0MB</td>
<td>62.0MB</td>
<td>2.0MB</td>
<td>2.0MB</td>
<td>0.0MB</td>
</tr>
<tr>
<td>LFAREA 2G STATISTICS - SOURCE = IEASYS23</td>
<td>0.0MB</td>
<td>0.0MB</td>
<td>0.0MB</td>
<td>0.0MB</td>
<td>0.0MB</td>
</tr>
</tbody>
</table>

- New display in v2.3 with large page statistics.
- Pageable 1MB stats including number of 1M still available. If often low, consider adding memory.
- LFAREA 1M stats including total size (cap).
- LFAREA 2G stats.
Other Performance Techniques

• Large pages have great benefits but there are many other ways to gain performance.
• Applications:
  – Avoid frequent obtaining and freeing storage.
    • Use cell pools or pre-allocated storage on common paths instead of GETMAIN each time.
  – Buffer data instead of a second I/O or long calculation.
  – Multi-threaded applications should consider cache alignment of frequently used fields.
    • Keep frequent updates in separate cache lines from unrelated frequent reads.
• Installations:
  – Ensure plenty of memory is available.
    • Avoid all paging if performance is a premium.
  – Use storage-class memory (SCM) such as Flash Express or Virtual Flash Memory (VFM).

And much more… performance and tuning is a vast topic barely touched on here.
Questions?
Virtual Storage Areas: Common

- **Common (Global) Storage**
  - Shared by all address spaces
    - Contents of a particular virtual address is the same for all address spaces.
    - Accessible using the DAT tables for any address space.

- **Different (separate) areas of common storage.**
  - **Prefixed Save Area (PSA)** – Maps fixed hardware and software locations for the related processor
  - **Common Service Area (CSA)**
    - Pageable and fixed data areas
    - Some load to global modules
  - **Link Pack Area (LPA)**
    - **Pageable Link Pack Area (PLPA)**
      - Built at IPL time from libraries specified in LPALSTxx or PROGxx.
      - Contains SVC routines, access methods, and other read-only system programs, some select read-only re-enterable user programs that an be shared among users of the system, some frequently used refreshable SYS1.LINKLIB and SYS1.CMDLIB modules.
    - **Fixed Link Pack Area (FLPA)**
      - Built at IPL time as specified in IEAFIXxx.
    - **Modified Link Pack Area (MLPA)**
      - Built at IPL time as specified in ILEAPAXxx.
  - **System Queue Area (SQA)**
    - Contains tables and queues relating to the entire system
    - When not enough SQA storage available, storage may be taken from CSA
  - **Nucleus (NUC)**
    - Built at IPL time
    - Read-only nuc, Read-write nuc
Virtual Storage Areas: Private

- Private (Local) Storage
- Not shared across address spaces (each address space has its own)
  - Content of a particular virtual address not same in another address space
- Accessible only using the DAT tables from that address space
- Different (separate) areas within the private area
  - System Region
    - GETMAINs for tasks running under RCT
  - ‘Low-end’ of private area
    - User Region
  - ‘High-end’ of private area
    - Local System Queue Area (LSQA)
    - Area for system tables and queues
    - associated with the users address space
    - Scheduler Work Area (SWA) – Contains control blocks for Initiator/Scheduler
    - Subpools 229 and 230
    - Storage obtained in requestor’s storage protect key
    - Used for control blocks only obtained by auth programs with appropriate key
Storage Key Protection

- Storage keys ensure only programs with the right permissions have access to storage.
- If a program attempts to access a page in the wrong key an ABEND results.
Storage Key Details

- Pages have a storage key and fetch protect status.
- Programs run in a PSW key.
- Programs can only read and write to pages that have a storage key matching their PSW key, with the following exceptions:
  - Programs running with PSW key 0 can read and write to any key.
  - All programs can read and write key 9 storage.
  - All programs can read any storage that is not fetch protected.
- System keys are 0-7.
- User keys 8-15.
Changing Keys

• In addition to PSW key, each program has a PSW key mask (PKM) in control register 3.
  – This is initially set to match the programs PSW-key but can be changed on attach or PC routines.
• Programs can only change to a key defined by their PKM.
• Additionally authorized programs (key 0-7, supervisor state, or APF authorized) can change to any key.
• A program must be Supervisor State and Key 0 to change the storage key of a page.
DIAGxx TRAPS for 64-bit

- iarCp64InitGet – Put non-zero into cells on get for test.
- iarCp64InitFree – Put non-zero into cells on free for test.
- iarCp64Trailer – Always use cell trailer to detect overflow.
- iarSt64InitGet – Put non-zero into storage on get for test.
- iarSt64InitFree – Put non-zero into storage on free for test.
- iarSt64Trailer – Always use storage trailer to detect overflow.