

Unit 5

IMS Locking, IMS Logger, Syncpoint Processing & IMS System Services



Program Isolation (1 of 2)

LOCK Mgmt

- **Program Isolation** (PI) facility is IMS's original Lock Manager
 - key function of any lock manager: preventing other units of work from accessing updated data prior to the changes being committed
 - PI allows multiple units of work (UOW) the ability to concurrently access the same database (but not the same segment occurrence!) for update while preserving integrity
- **Dynamic Backout facility:**
 - Works with the IMS Lock Managers to automatically remove the effects of any program that abnormally ends (ABEND) before committing its updates
 - Eliminates partial updates - uncommitted updates are backed-out prior to lock release
 - One of the more common reasons for Dynamic Backout being invoked is to manage ABENDs triggered due to *lock failures*



Program Isolation (2 of 2)

LOCK Mgmt

- **Deadlocks** are a variety of lock failure that is possible with any lock manager
 - PI handles deadlocks through Identification and avoidance:



- PI analyzes for the possibility of a deadlock as part of granting lock requests
 - If PI determines that granting a lock would produce a deadlock, it selects *loser* UOW for *pseudo-ABEND* and dynamic back-out
 - *Loser* UOW is re-scheduled for later (with IMS TM) if possible
 - The ability to automatically reschedule the UOW is one of the factors PI uses to select the *loser* – that is, the UOW easier to reschedule if it is more likely to be ABENDED if involved in a deadlock
- A lengthy wait for a lock is not necessarily a deadlock
 - PI will wait indefinitely for locks as long as they are not deadlocks

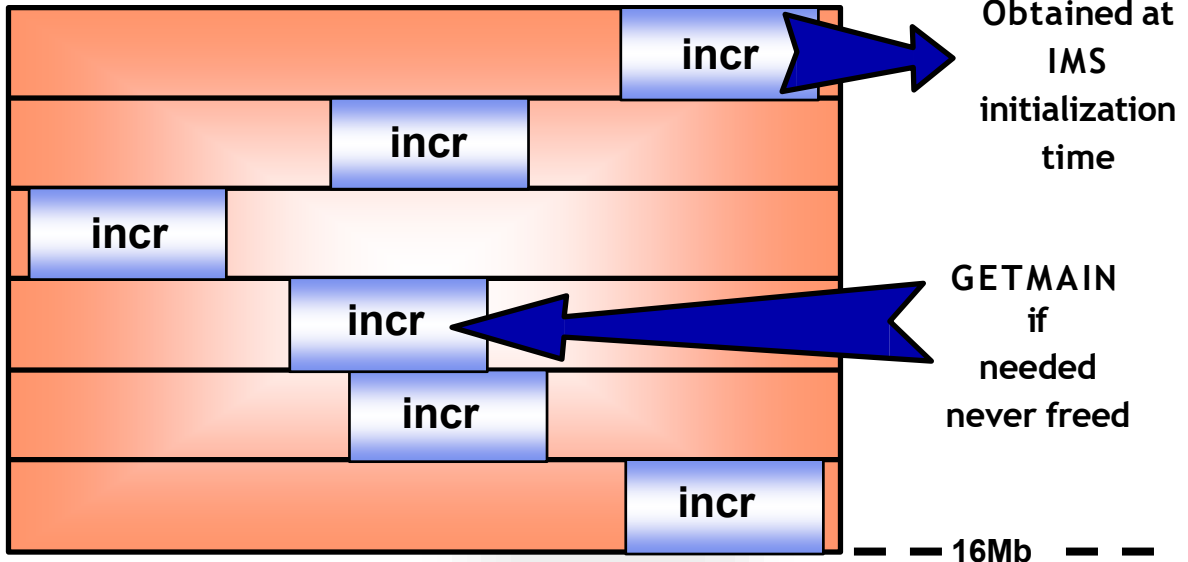
Program Isolation ENQ/DEQ Pool

LOCK Mgmt

Specified in IMS startup procedure

PIMAX = (max. bytes)

PIINCR = (increment)



IRLM

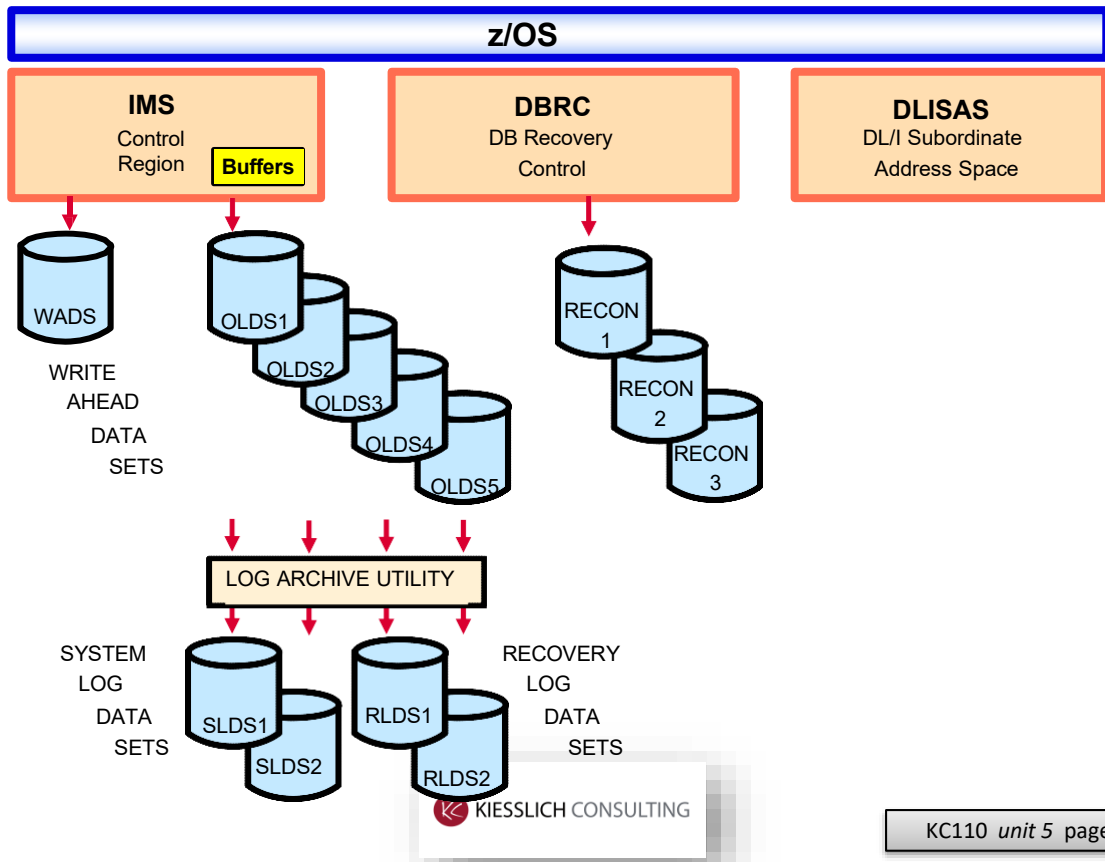
LOCK Mgmt

- IRLM was introduced to enable IMS *Data Sharing*
 - This lock manager was later adopted by DB2
- Data Sharing use is not a requirement for the use of IRLM instead of PI:
 - Lock granularity is different between PI and IRLM
 - PI locks at the Segment and Database record level and IRLM locks at the Block Level for updates
 - In most cases, PI uses less CPU than IRLM
- We will discuss IRLM further in the Data Sharing topic
- There is a very good doc about [IMS Locking](#)



Common Logging Facility overview

LOG Mgmt



Common Logging Facility (1 of 3)

LOG Mgmt

- Recording of IMS/DB and IMS/TM system activity:
 - *Before* and *After* images of changed DL/I segments
 - System status via periodic checkpoints
 - Input and Output messages are also logged
- Provides integrity and recoverability of DB/TM systems:
 - Used for Dynamic Backout of failing transactions
 - Used for system *Warm* and *Emergency* restarts
 - Used for database recovery
 - Used for performance and accounting statistics



Common Logging Facility (2 of 3)

LOG Mgmt

- IMS internally has two log processes:
 - Logical Logger moves data to log buffer
 - *WRITE* - requests
 - For example, database *before* and *after* records
 - Physical Logger writes/reads data between log buffer and disk
 - A *CHECK WRITE* is a synchronous process to verify certain records are on Disk
 - If not, a request to initiate I/O is issued and requesting ITASK will wait for completion
 - Check Write requests can result in partial buffers being written to the WADS
 - Check Write example: Purge changed DB buffers for buffer steal
 - A *WAIT WRITE* is also a synchronous request to externalize log records
 - Similar to *Check Write* except *Wait Write* does not initiate I/O request
 - Requesting ITASK will be notified when its log records have been written by :
 - > *Piggybacking* on Check Writes (or the log buffer filling) of other ITASKS
 - > A *timer pop* that periodically writes log records during periods of low activity
 - Wait Write example: Commit related records at application syncpoint
 - *READ* requests read data from disk to buffer
 - Used mainly for dynamic backout processing



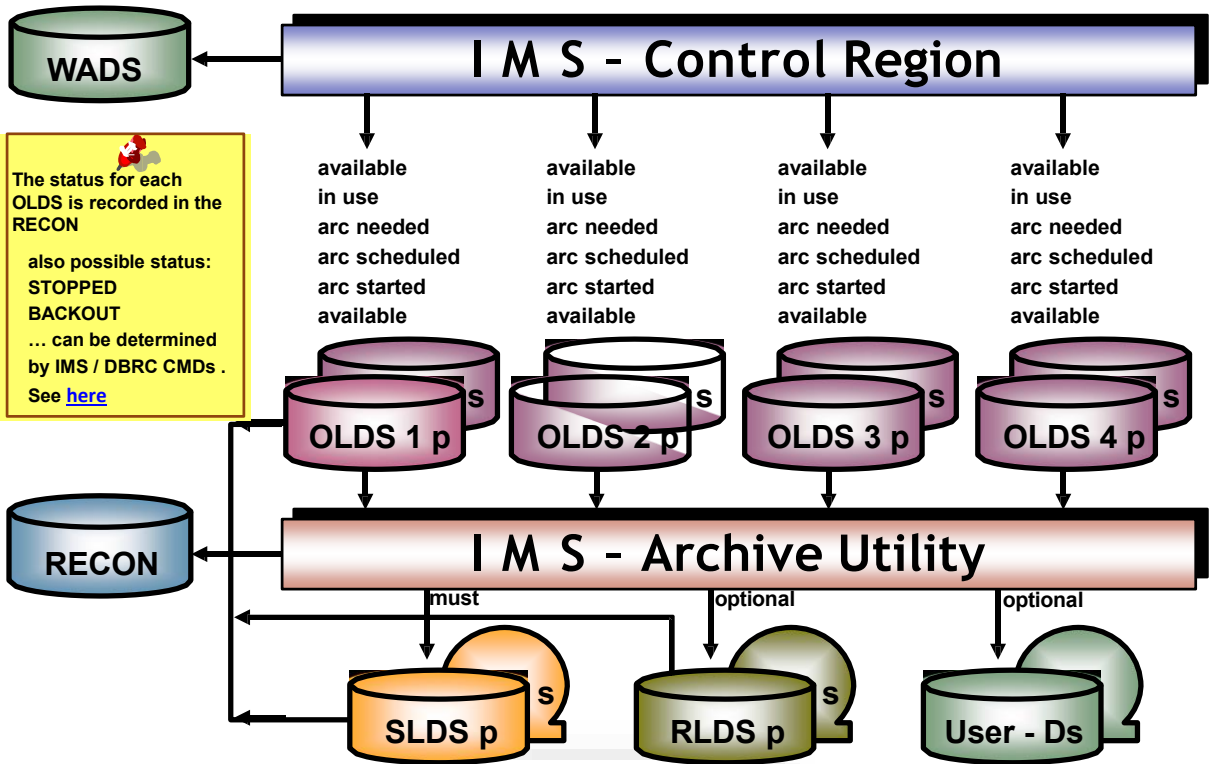
Common Logging Facility (3 of 3)

LOG Mgmt

- Log Write-Ahead (LWA) - WADS:
 - To guarantee database integrity, log records are physically written before database CI's/Blocks are physically written
 - Only selected *events* qualify for this technique
 - The DCLWA= parm (on TRANSACT macro) also controls when log records associated with messages must be written for the transaction
 - Implemented through the use of Check Writes and Wait Writes



Principle of IMS DASD logging LOG Mgmt



The status for each OLDS is recorded in the RECON

also possible status:
STOPPED
BACKOUT
... can be determined by IMS / DBRC CMDs .
See [here](#)





and



LOG Mgmt

- **OLDS** and **WADS** are written from common IMS buffers
 - Contains log records needed for:
 - IMS System Restart
 - Database Recovery
 - Application Dynamic Backout
- Online Log Data Set (**OLDS**):
 - Are only written when the IMS log buffer is filled
 - Consists of multiple data sets - usually in pairs
 - Allows IMS to continue logging when an *OLDS switch* occurs because of the following events:
 - I/O error – especially OLDS is full
 - Operator commands /DBR or /DBD
 - Written in a *wrap-around* manner
 - Contents archived by Archive Job to the System Log Data Set (SLDS)
- Write Ahead Data Set (**WADS**):
 - Written in response to a Check Write or *timer pop*
 - Contains committed log records not yet written to OLDS
 - Only short (2048 or 4096 bytes of data), full blocks are written
 - Can be used to *close* the OLDS in an emergency restart



Definition of IMS log datasets (1)

LOG Mgmt

- **OLDS**: Online Log Data Set
 - Contains all log records required for IMS restart, dynamic backout and recovery
 - BLKSIZE min 6K and max 30K
 - 26K was previous recommended blocksize because it was half-track blocking
 - > Currently, if OLDS Blocksize is divisible by 4K, Log Buffers will be stored above 2 GB bar and WADS blocksize will be 4K instead of 2K
 - For 3390 BLKSIZE=24K is **recommended**
 - Dual OLDS **recommended**
- **WADS**: Write Ahead Data Set
 - Might contain committed log records not yet written to OLDS
 - Guarantees log records available for restart/recovery after system failure
 - Can be formatted at startup
 - Dual WADS **usually recommended**
 - Spare WADS **always recommended**
 - BLKSIZE=2080 (2048 + 32 Prefix) or 4128
 - Min TRKs = $[(\text{OLDS-BLKSIZE} / 4096) + 1] * 2$
 - WADS size of 4-5 cylinders is usually recommended
 - Max TRKs = $((\text{OLDS-BLKSIZE} / 2048) + 1) * \text{OLDS-BUFNO}$

Definition of IMS log datasets (2)

LOG Mgmt

- SLDS: System Log Data Set
 - Contains archived log records
 - Dual SLDS **recommended**
- RLDS: Recovery Log Data Set
 - Contains log records needed to recover databases (DB change and I/O-error records, and records that record the start and end of UOWs)
 - Dual RLDS **recommended**
- Note that all of these log data set types provide the option of IMS software-implemented *mirroring*
 - If you **do** select to have a dual copy of a data set, make sure that it is not allocated on the same VOL=SER as the primary copy
 - The major purpose of a second copy is to reduce or eliminate the impact of a hardware failure



Specifying IMS logging options

LOG Mgmt

Some **OLDS/WADS attributes** extracted from DFSVSMnn member of PROCLIB

```
OLDSDEF OLDS = (00,01,.....,99),  
MODE = DUAL / SINGLE,  
BUFNO = 3 → 9999,  
DEGRADE=YES / NO  
  
WADSDEF = (0,1,.....,9)  
ARCHDEF ALL MAXOLDS(1)
```

- If no DD statements in IMS JCL, then DFSMDA members required for:
 - DFSOLPnn, DFSOLSnn, DFSWADSnn
 - **The use of DFSMDA (Dynamic allocation) for OLDS and WADS data sets is strongly recommended**
- WADS duplexing via WADS = S/D parameter in IMS startup procedure **ONLY – not** in DFSVSMnn
- **Minimum of 3** OLDS subparameter required
- BUFNO, MODE extracted from DFSVSMnn (OLDSDEF)
- **Minimum of 1** WADS subparameter
- OLDS and WADS data sets must be preallocated and cataloged
 - Be careful of multiple data set extents - especially with SMS, only first extent will be used

WADS Writes and Definition

LOG Mgmt

- The concept of WADS track groups is not used any longer since IMS V12
 - WADS should be sized to provide enough space for any OLDS buffers not yet written at any time plus one track
 - WADS writes are changed from previous IMS versions
 - IMS 12 writes to WADS from previously written log record in the buffer to the last log record in the buffer
 - WADS written in wrap-around fashion
- Performance tipp : WADS could be kept in cache in storage subsystem

Maximum WADS size before IMS V12 with 200 24K buffers:

$(\text{OLDS block size}/\text{WADS segment size} + 1) \times (\text{no. of OLDS Buffers})$
 $((24\text{K}/4\text{K}) + 1) \times 200 = \underline{\mathbf{1400 \text{ tracks}}}$

Maximum WADS size with IMS 12 with 200 24K buffers:

3390 Model 9 allows 56664 bytes per track

$56664/24\text{K} = 2$ blocks per track

$200 \text{ buffers}/2 + 1 = \underline{\mathbf{101 \text{ tracks}}}$



DBRC functions (1 of 2)

RECOV Mgmt

- Two types of DBRC used
- *Log-Control*, is mandatory and its primary function is to track IMS log data set usage:
 - Maintains information about OLDS status, for example, INUSE ARCHIVE-NEEDED AVAILABLE, and so on
 - Maintains information on data set names, creation times, and contents of the various RLDS and SLDS data sets
- Optionally, on a case-by-case basis, databases can be registered to DBRC
 - Once registered, DBRC provides the same functions for databases that were called *Share-Control* prior to IMS 6.1
 - These functions include:
 - Assistance in database integrity and recovery
 - Preventing access to databases in a way that could impact recoverability
 - An example is when a job is prevented from updating a database not yet Image Copied

DBRC functions (2 of 2) *RECOV Mgmt*

- The functions provided for Registered Databases (continued)...:
 - Records the occurrence of key events that impact DB usability such as:
 - Database Image Copy (ICs), Change Accumulation (CA), Reorganization (Reorg) and Recovery (DBR) Job executions
 - Automatic JCL generation of needed job-streams
 - Database Sharing, with integrity, between IMS systems
 - Intra-processor (same z/OS) or inter-processor (different z/OS) block-level sharing with integrity
 - Changes (in addition to DBRC changes) are required when implementing Data Sharing



RECON maintained Information (1)

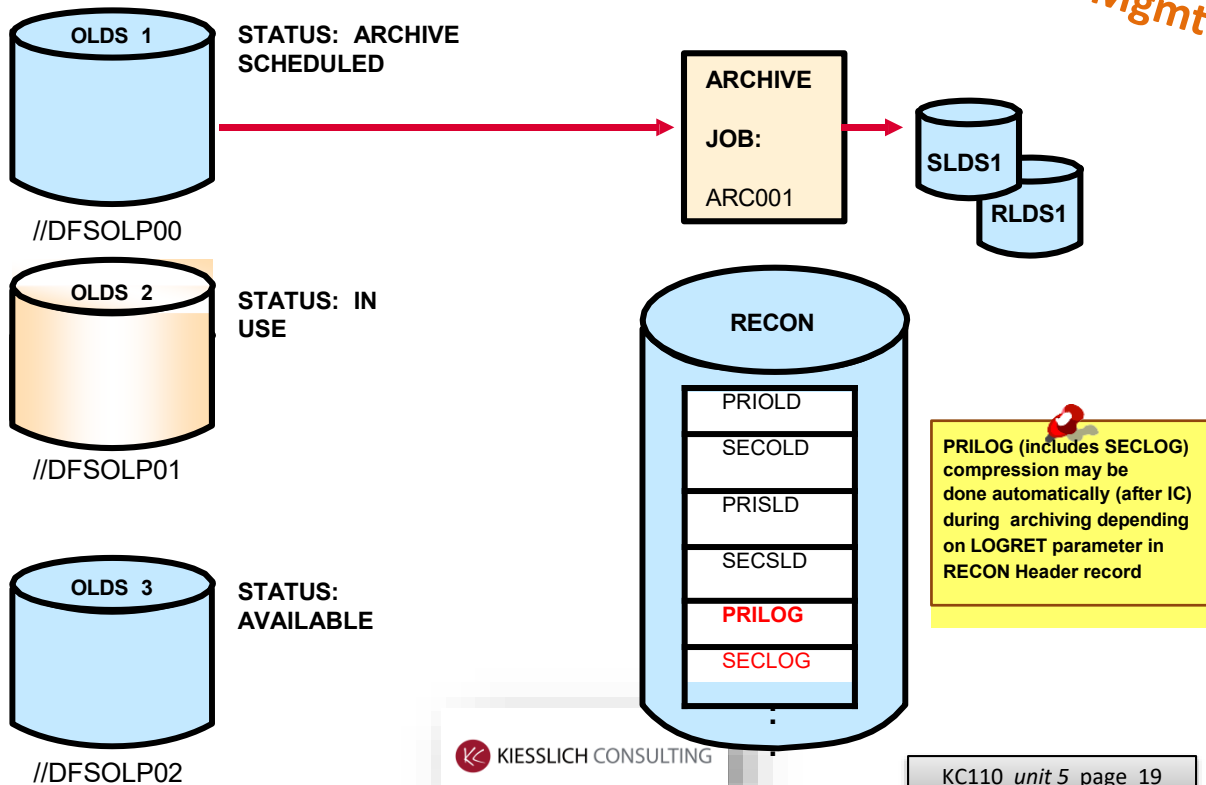
RECOV Mgmt

- Recovery Control Data Sets (RECONs)
- Automatic recording of information about:
 - Online Log Data Set (OLDS)
 - System Log Data Set (SLDS)
 - Recovery Log Data Set (RLDS)
- Generation of utility JCL via DBRC command for:
 - Log Archive
 - Log Recovery



RECON maintained Information (2)

RECOV Mgmt



DBRC Database Registration

RECOV Mgmt

- Advantages of Registering Databases:
 - Records recovery-related information in RECON
 - Example: Provides information on which RLDS has log data for each database data set (DBDS) registered with DBRC
 - Provides databases with enhanced integrity by identifying valid recovery points for databases
 - DBRC GENJCL feature Generates JCL for recovery-related utilities:
 - Used to select correct data sets to be used by Recovery and Image-Copy
 - Consolidates changes of log records by creating the correct input for *Change Accumulation* Utility Jobs
 - Reduces or Eliminates errors
 - Verifies that JCL used with IMS Utilities has valid input specified
- Registration of databases is performed through DBRC commands
 - DBRC is covered in detail in the 32 hrs DBRC class CM20xx



Checkpoint ID Table – RDS (1 of 2)

Restart
(RECOV)
Mgmt

Assume System Failure at 12:45 – restart must examine logs starting from 9:00 in support of PST 2 backout

Checkpoint ID Table

PST	prev Sys-CHKPT	Appl - Commitpoint	VOL=SER
1	9:00	9:05 end	=
2	→ 9:00	9:10 begin	
3	→ 10:00	10:05 begin	
4	11:00	11:05 end	
5	→ 12:00	12:05 begin	
.			
.			

Most recent System (simple)
Checkpoint prior to PST UOW start

where:

end = End UOR (Synchpoint)
begin = Begin UOR



KISSLICH CONSULTING

Checkpoint ID Table – RDS (2 of 2)

Restart
(RECOV)
Mgmt

IPCS OUTPUT STREAM -----

CHECKPOINT ID TABLE

18F4A000	00000000	00001000	1A010040	00000000	*.....*
18F4A010	00000000	00000000	00000000	00000001	*.....*
18F4A020	00000000	00000000	00000000	C2C3D7E3	*.....BCPT*
18F4A030	00000000	00000000	CABC11DE	5E31B0CB	*.....;...*
18F4A040	D9E5C5E3	02040200	05660000	00000000	*RVET.....*
18F4A050	C2C3D7E3	18F4A000	00000001	00000366	*BCPT.4.....*
18F4A060	E2C9C4E7	18F49000	00000002	00000000	*SIDX.4.....*

Command ==>

SCROLL ==> CSR

F1=HELP F2=SPLIT F3=END F4=RETURN F5=RFIND F6=MORE F7=UP
F8=DOWN F9=SWAP F10=LEFT F11=RIGHT F12=CURSOR

Menu Utilities Compilers Help

BROWSE IMS12A.RDS

Line 00000000 Col 001 080

***** Top of Data *****

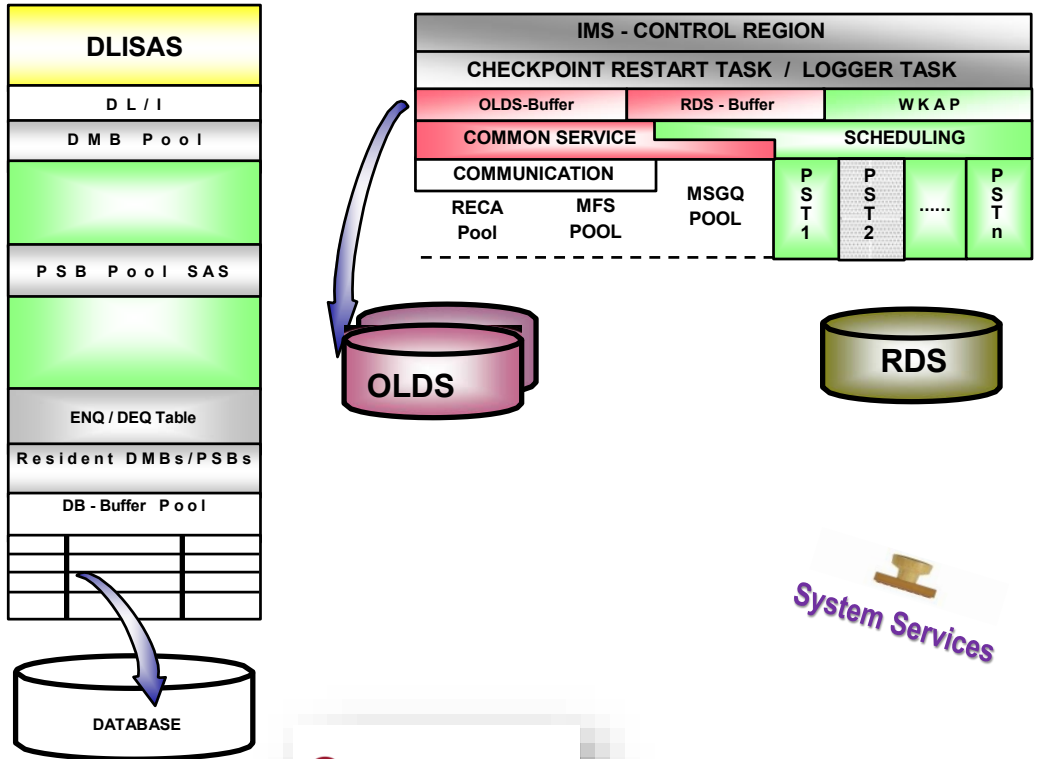
.....BCPT.....	-ü;.	¢ôRVET.....	Ã.....	
.....ö.....	&.....	SIDX.....	-ü;.	¾»...°... IMSA
.....ü.....	Q.....	LCRE.....	-ü;.	:.....ç

Command ==>

Scroll ==> PAGE



Syncpoint processing (1 of 2)



Syncpoint processing (2 of 2)

- Syncpoint processing common to IMS/TM, CICS and BMP:
 - Changed data committed:
 - Full function changes written to database
 - DEDB changes become eligible to be written (asynchronously)
 - Locks released - changed data available for other users
 - Pool space is retained until program **termination** (except for CICS, see below)
- Commit point processing specific to IMS/TM:
 - MSDBs updates and SPA changes made permanent
 - Output messages enqueued to final destination
- Commit point processing specific to CICS:
 - Pool space becomes eligible for other use
 - PSB and DMBs marked inactive and PSBW space freed
 - Done even if the next PSB scheduled is the one we are terminating here
 - CICS thread released at PSB termination
 - CICS thread - PST assignment will **not** be released
 - Dependent on workload, additional threads (above MINTHRD) and their associated PSTs are freed



IMS Storage Management POOLS (1 of 2)

- IMS has different storage manager modules, they are:
 - The old Pool Storage Manager - DFSISMNO
 - Also called CBT (Control Block Table) Storage manager
 - The new Pool Storage Manager - DFSPPOOL
 - Also called the Dynamic Storage Manager
- The old Pool Storage Manager manages pools that are fixed in size at IMS Initialization:
 - Space reassignment and reclamation are not sophisticated
 - IMS can run out of space for pools managed by this method
 - Unfortunately, all Scheduler-related pools (PSB, DMB, PSBW, others as DLMP, DPSB, DBWP, EPCB, MAIN and so on) use this old Manager


System Services



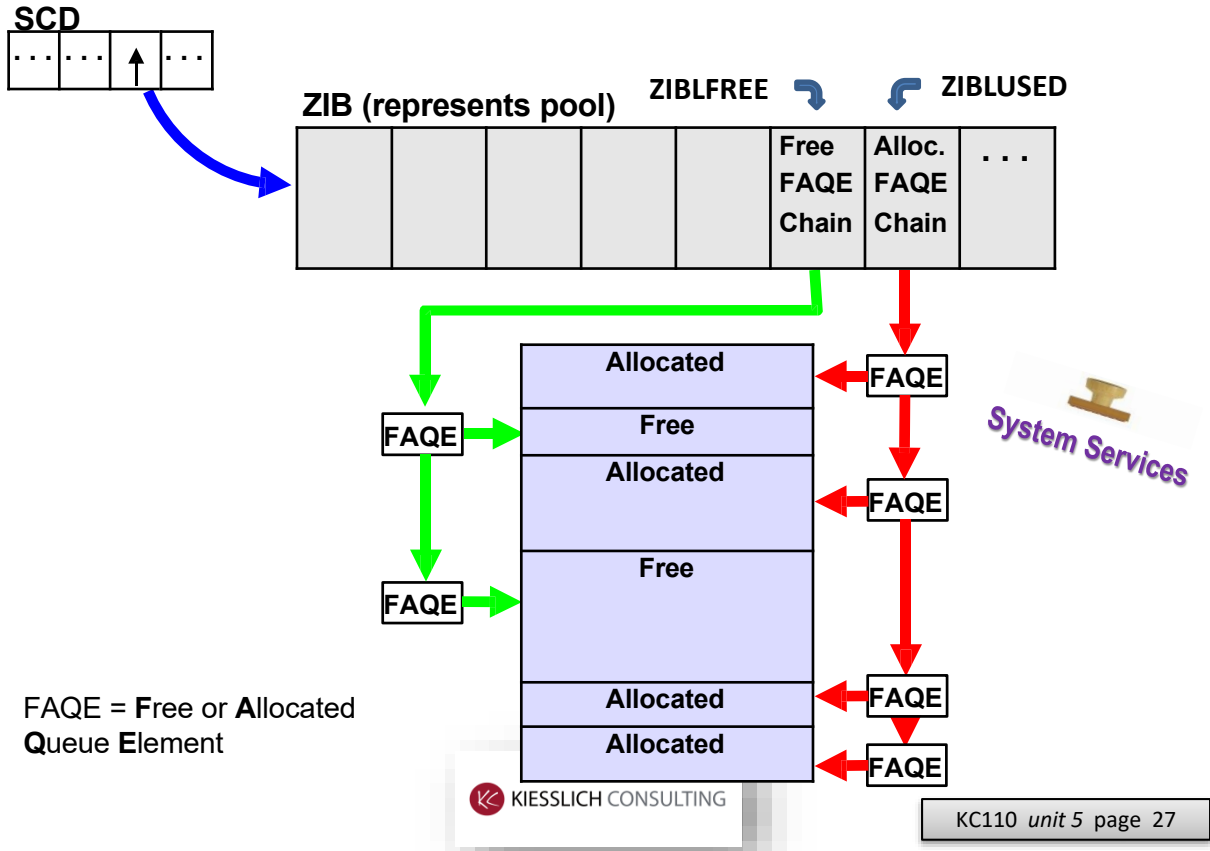
IMS Storage Management POOLS (2 of 2)

- The New Pool Storage Manager is better than the old one, at responding to varying demands for storage:
 - A user-specified amount of space is allocated at IMS initialization
 - Fixed length buffers / space chunks are assigned in a *best fit* manner for variable length requests
 - Pools managed by this Storage Manager can dynamically grow in size, if necessary
 - The CIOP and the HIOP are two pools managed this way (and SPAP, CESS, EMHB, FPWP, LUMC und LUMP)
- Third (newest) type of storage management is done by DYNAMIC CONTROL BLOCK STRUCTURE (CBS)
 - Allocating blocks on page boundaries (4k) – minimizes paging
 - Can release unused **IPAGES** : If no blocks are allocated out of the IPAGE , it is freed (timing interval scans)
 - CBS has header and entry portion (see DFSCBTS and CBTE)

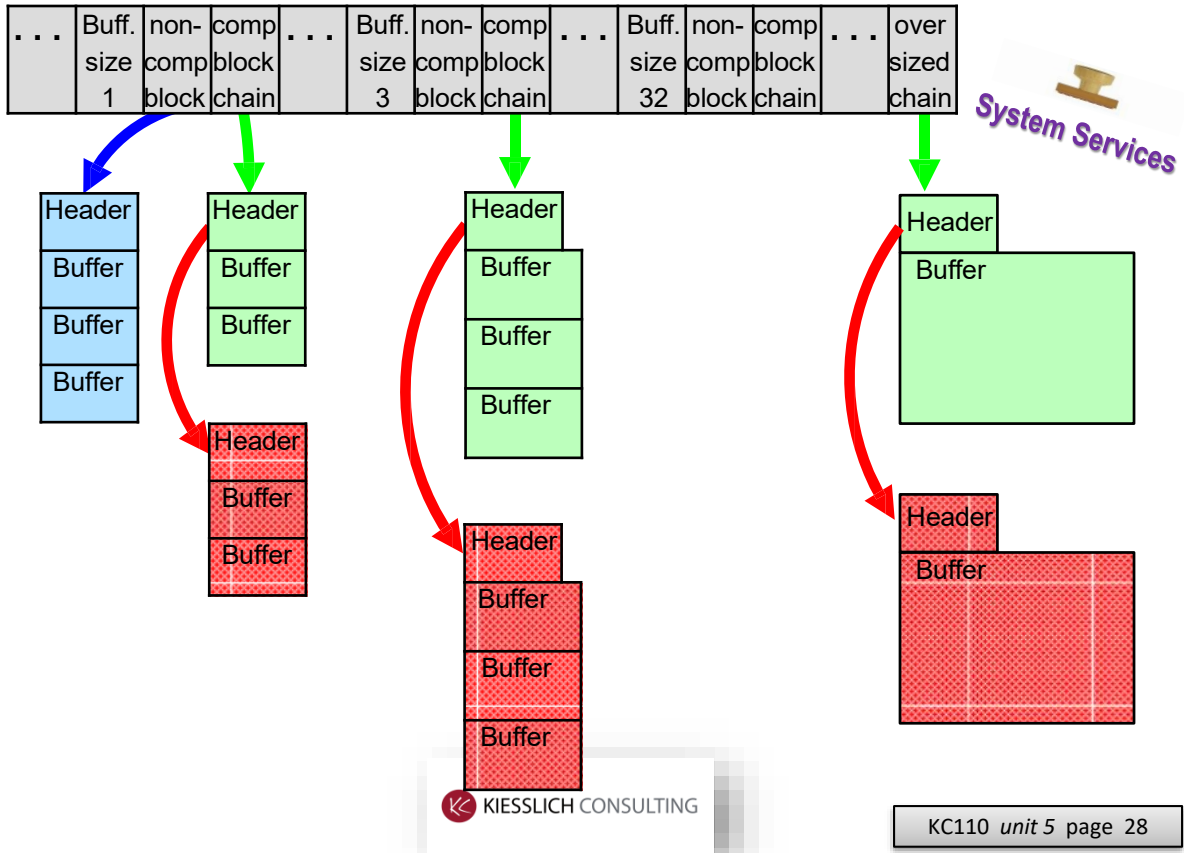

System Services



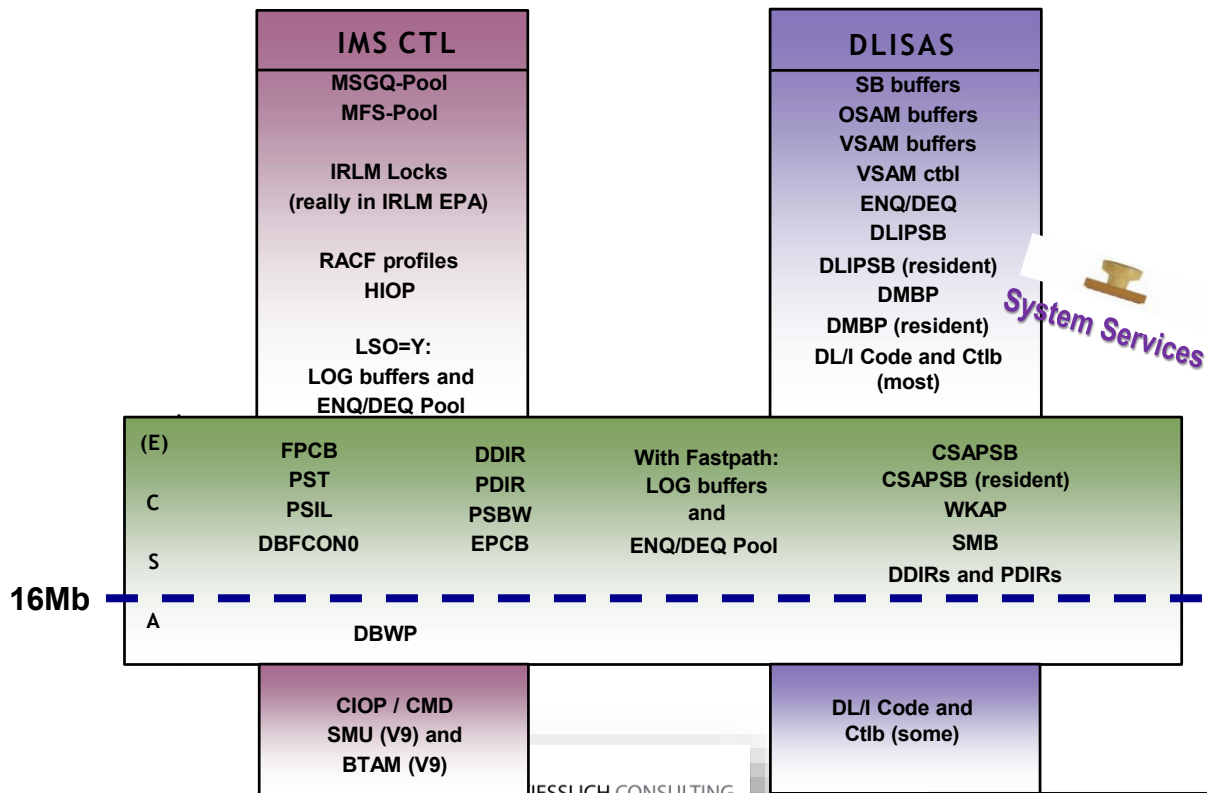
Old Pool Storage Manager



New Pool Storage Manager



z/OS: IMS CSA and Priv. Storage use



z/OS Cross-Memory Services with IMS

