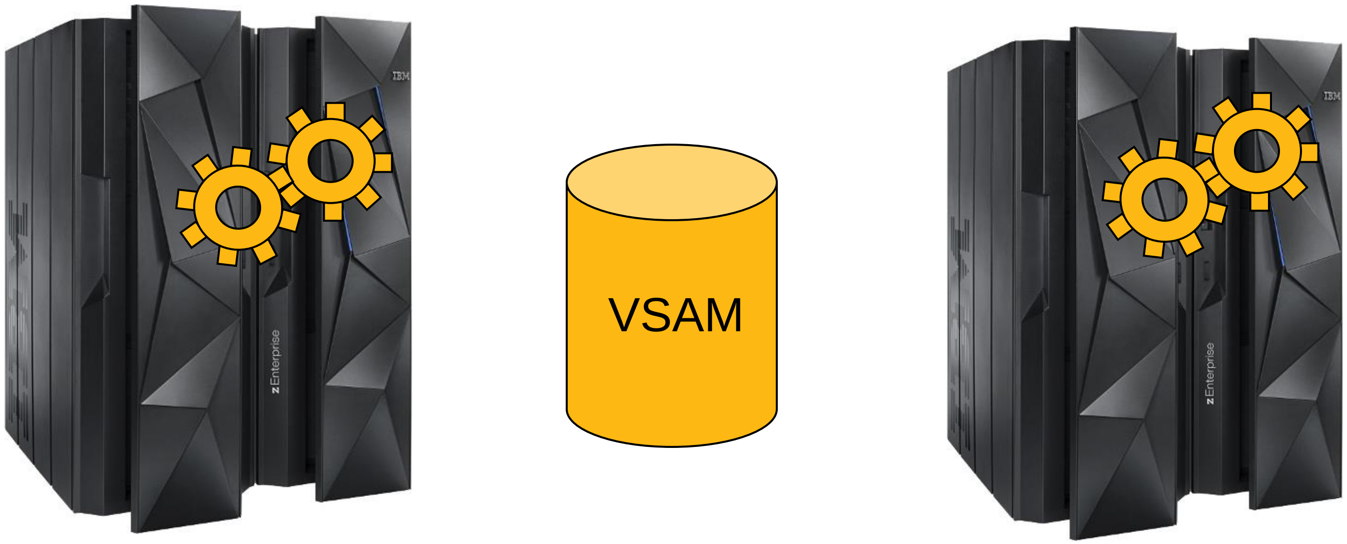
VSAM RLS/TVS Overview



Terri Menendez
STSM
z Systems Division
VSAM/RLS/Catalog Development
terriam@us.ibm.com

Notices & Disclaimers

No part of this document may be reproduced or transmitted in any form without written permission from IBM Corporation.

Product information and data has been reviewed for accuracy as of the date of initial publication. Product information and data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or programs(s) described herein at any time without notice.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such such products, programs or services available in all countries in which IBM operates or does business. Consult your local IBM representative or IBM Business Partner for information about the product and services available in your area.

Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectually property rights, may be used instead. It is the user's responsibility to evaluate and verify the operation of any non-IBM product, program or service.

THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR INFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. IBM is not responsible for the performance or interoperability of any non-IBM products discussed herein.

Notices & Disclaimers

The performance data contained herein was obtained in a controlled, isolated environment. Actual results that may be obtained in other operating environments may vary significantly. While IBM has reviewed each item for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere.

The responsibility for use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's or user's ability to evaluate and integrate them into their operating environment. Customers or users attempting to adapt these techniques to their own environments do so at their own risk. IN NO EVENT SHALL IBM BE LIABLE FOR ANY DAMAGE ARISING FROM THE USE OF THIS INFORMATION, INCLUDING BUT NOT LIMITED TO, LOSS OF DATA, BUSINESS INTERRUPTION, LOSS OF PROFIT OR LOSS OF OPPORTUNITY.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not necessarily tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Agenda

- Overview of RLS z/OS Release enhancements.
- IBM Products Exploiting RLS.
- Record Level Sharing - Design direction.
- Review of VSAM vs RLS:
  - Share Options
  - Buffering
  - Locking
  - RAS
  - Performance Measurements
- RLS/TVS Configuration Changes
  - Parmlib Changes
  - SYSPLEX with SMSVSAM
- SMSVSAM Initialization
- SMSVSAM Commands
- RLS/CICS Environment
  - CICS and base VSAM FOR configuration
  - CICS and RLS configuration
  - RLS/CICS data recovery
  - RLS/CICS automation enhancements
- Transactional VSAM (TVS)
  - Hardware/Software Requirements
  - Application Requirements
- Questions

RLS z/OS Release Enhancements

RLS z/OS Release Enhancements

- ❑ OS 2.1
  - ❑ RLS for Catalog (removes SYSIGGV2 enq)
  - ❑ RLS for AMS
  - ❑ Omegamon RLS Support
  - ❑ SHCDS LISTSTAT (sysplex wide stats while DS open - OA42435)
  - ❑ **Access Method Encryption (OA50752 / OA51065)**
  - ❑ **VSAM / VSAM RLS zHyperlink™ Exploitation (OA52941 / OA52876)**
  - ❑ **SMB Index Buffer Parameter SMBVSPI (OA49604)**
- ❑ z/OS 2.2
  - ❑ KSDS RLS Index Record Locking (removes CI split lock)
  - ❑ Primary / Secondary Extent Reduction
  - ❑ VERIFY RECOVER
- ❑ z/OS 2.3
  - ❑ TVS Auto-Commits **(OA55176)**
  - ❑ RLS Upgrade Locking (removes AIX upgrade lock)
- ❑ z/OS 2.4
  - ❑ **EzNoSQL NoSQL Key:Value Store (OA65536)** . Also in 2.3.

RLS z/OS Release Enhancements

- z/OS 2.5
  - Local Lock reduction: OA60377, OA61661, OAxxxxx
  - Change Data Capture (CDC) for RLS/TVS (OA59251/OA63195)
- z/OS 2.4/2.5 (futures)
  - REST / JAVA Support for EzNoSQL (OA64018)

IBM Products Exploiting VSAM RLS

IBM Products Exploiting RLS/TVS:

- CICS - User application files
- HSM   - HSM Control Data sets (MCDS, BCDS, etc.)
- INFOMAN
- SCLM
- IMS (RLS and TVS) – IMS Recon Data Sets
- Catalog – User catalogs

Record Level Sharing (RLS) – Design Direction

Record Level Sharing (RLS) - Design

- RLS is another method of access, to your existing VSAM files, which provides full read and write integrity at the record level, to any number of users in your parallel sysplex.

Review of VSAM

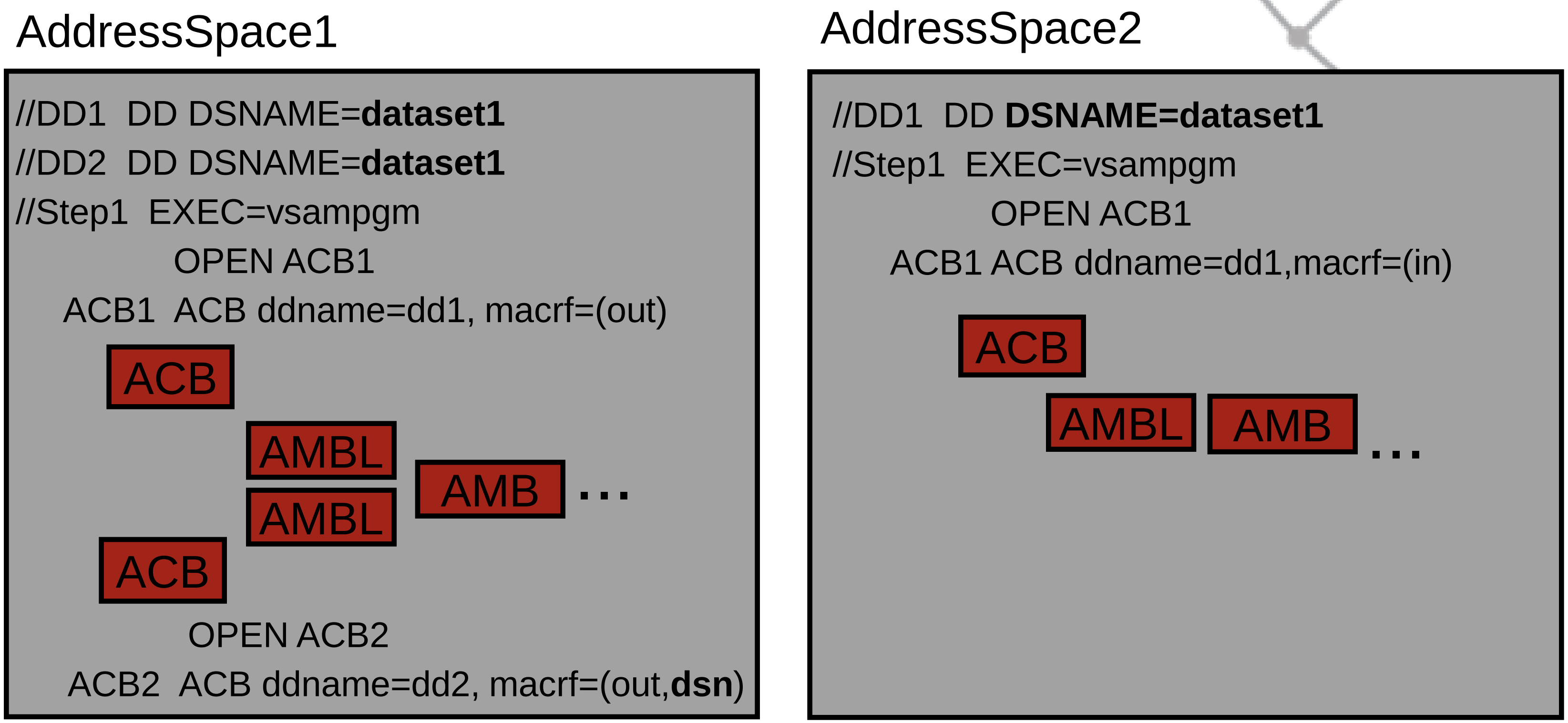
Review of VSAM

- ❑ Share options
- ❑ Buffering
- ❑ Locking
- ❑ RAS
- ❑ Performance Measurements

Review of VSAM

- ShareOptions(x,y).
  - attribute of the data set.
  - SHAREOPTIONS(crossregion,crosssystem)
    - **SHAREOPTIONS(1,x)** - Defined as one user opened to the data set for read/write or any number of users for input only.   VSAM provides full read/write integrity.
    - **SHAREOPTIONS(2,x)** - Defined as one user opened to the data set for read/write and any number of users for input VSAM provides full read/write integrity for the read/write user, however, the readers do not receive read integrity***.
    - **SHAREOPTIONS(3,x)** - Defined as any number of users opened to the data set for read/write. VSAM does not provide any read/write integrity.
    - **SHAREOPTIONS(4,x)** – VSAM will flush buffers after each request.
  - ACB MACRF=(DDN/DSN) is the only real mechanism for sharing VSAM files.
- *** **No read integrity means even “old” records may not be found.**

Example of ShareOptions (2,x)



AddressSpace1

```
//DD1  DD DSNAME=dataset1
//DD2  DD DSNAME=dataset1
//Step1  EXEC=vsampgm
               OPEN ACB1
    ACB1  ACB ddname=dd1, macrf=(out)
```

```
                OPEN ACB2
     ACB2  ACB ddname=dd2, macrf=(out,dsn)
```

AddressSpace2

```
//DD1  DD DSNAME=dataset1
//Step1  EXEC=vsampgm
               OPEN ACB1
     ACB1 ACB ddname=dd1,macrf=(in)
```

(read/write integrity)

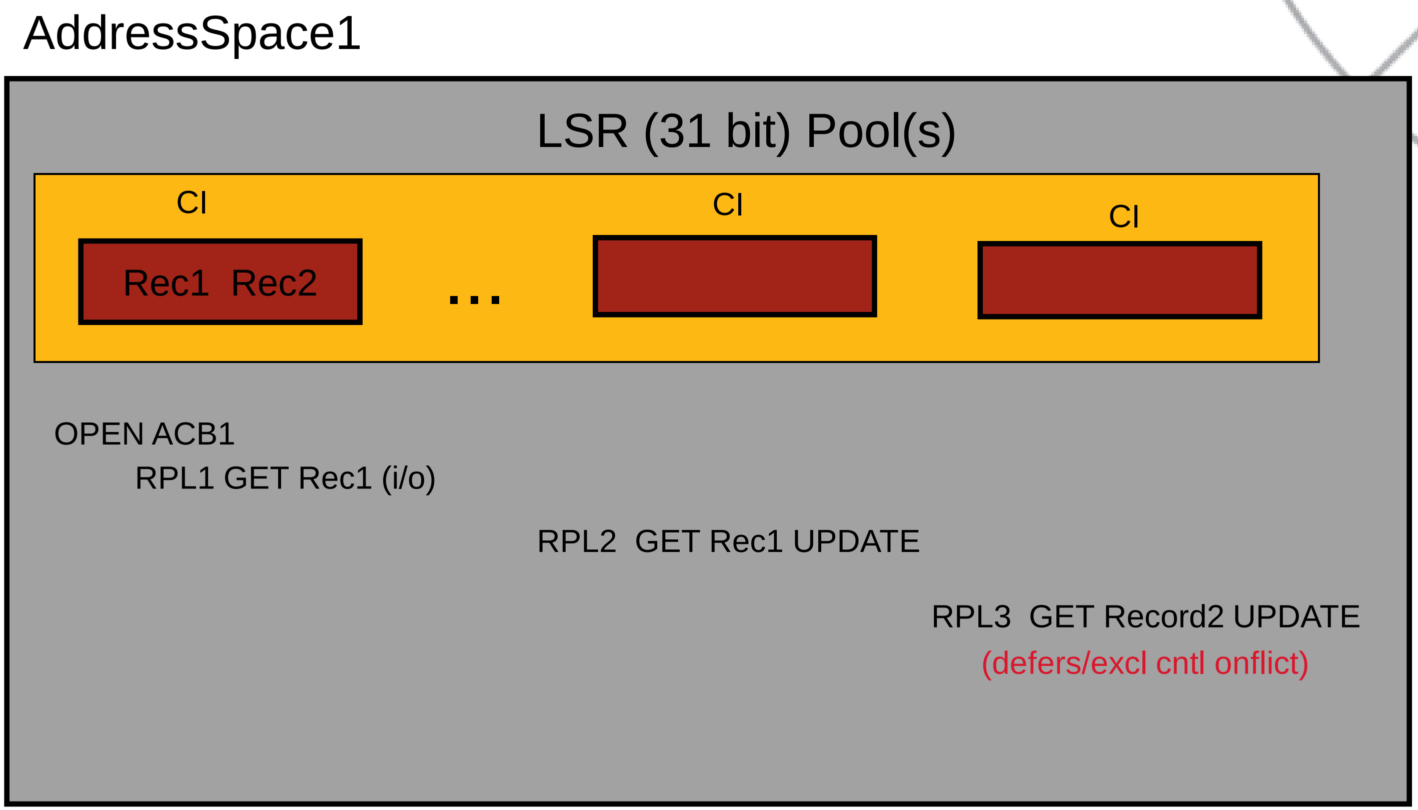
(no read integrity)

VSAM - Buffering

- Prior to RLS, VSAM provides 3 types of buffering:
  - ACB MACRF=(NSR/LSR/GSR).
    - NSR - Non-Shared Resources
    - LSR - Local Shared Resources
    - GSR - Global Shared Resources
- For LSR/GSR, user defined the buffer pool:

```
POOL1  BLDVRP  BUFFERS=(1024(5)),
               STRNO=4,
               TYPE=LSR,
               MODE=31,
               RMODE31=ALL
```

Example of LSR Buffering



AddressSpace1

LSR (31 bit) Pool(s)

OPEN ACB1

RPL1 GET Rec1 (i/o)

RPL2 GET Rec1 UPDATE

RPL3 GET Record2 UPDATE

(defers/excl cntl onflict)

(read/write integrity)

VSAM - Locking

- Base VSAM serializes on a CI level within a single address space.
- Multiple users attempting to access the same CI for read and write either defer on the CI or are returned an exclusive control conflict error by VSAM.
- CIs with many records per CI, or applications that repeatedly access the same CI can have a performance impact due to retrying of exclusive control conflict errors.
- CICS uses the No LSR Wait (NLW) option, which causes VSAM to return to CICS (instead of deferring), when contention is encountered. This allows CICS to breakup potential deadlocks. Requests are then redriven and FCVWAITs values represent each attempt to complete the request.

Example of VSAM LSR Serialization

GET UPD RPL_1

(Record B)

GET UPD RPL_2

(Record E)

- fails - Exclusive Control Conflict

Record A
Record B
Record C
Record D
Record E

Control Interval

VSAM - RAS

- VSAM has little to no first time data capture, and internal recovery, for logic errors.
  - All resources are obtained in a single address space.
  - EOT acted as cleanup routine (plus estae stacked by open/close).
  - Performance highly valued over RAS.
  - RAS in general was not a major requirement when VSAM was developed.
- End result:
  - Difficult problems to debug.
  - Broken data sets and data integrity problems mainly due to application error.

VSAM – Performance Measurements

- Base VSAM provides SMF 42, 62, and 64 records:
  - SMF 42-6 – I/O count.
  - SMF 62 – Created by OPEN for each ACB.
  - SMF 64 - Created by EOV and CLOSE for each ACB, however, the stats represent the sum of all ACBs connected to the control block structure.

Review of RLS

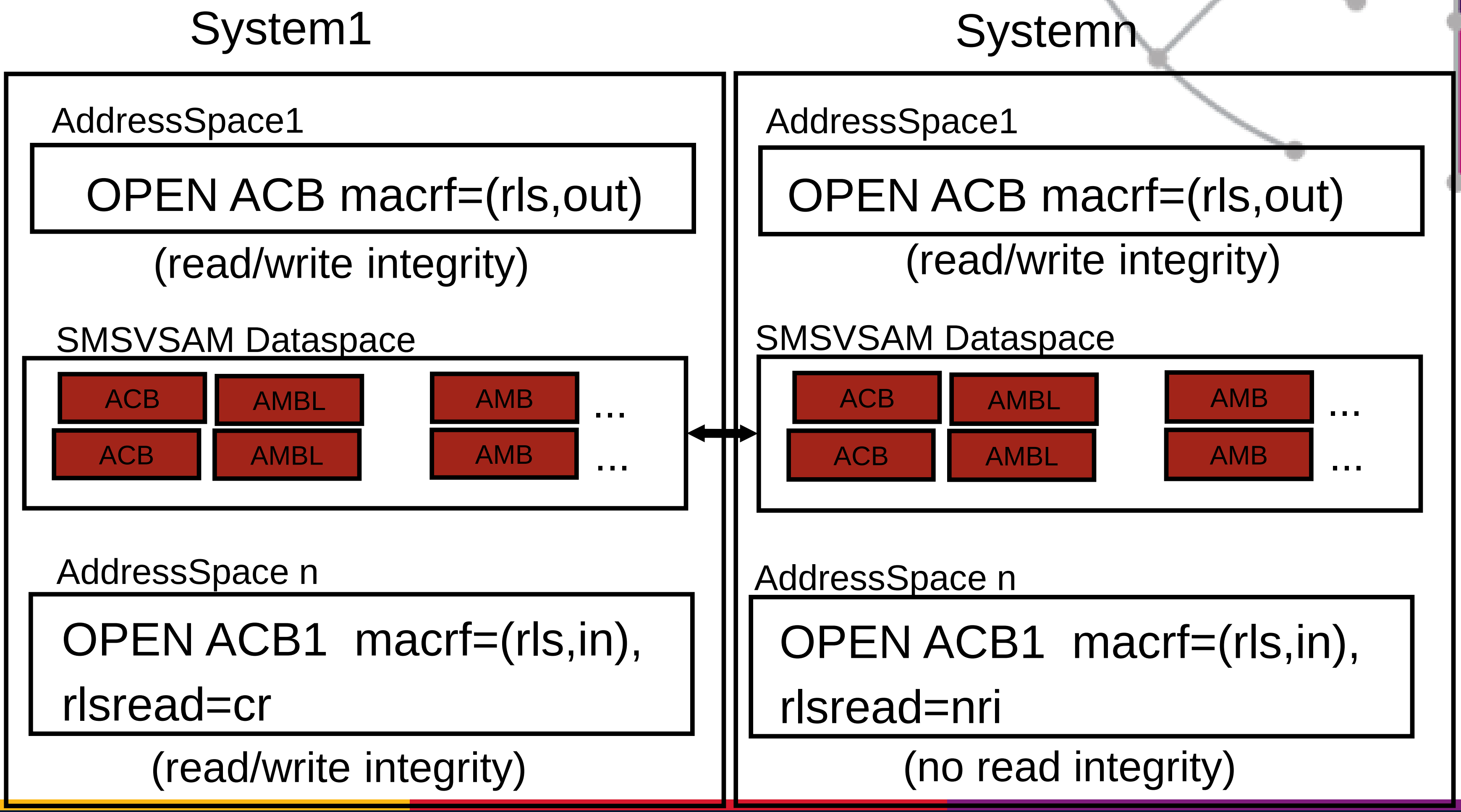
Review of RLS

- ❑ Share options
  - ➢ Example of RLS Readers/Writers
  - ➢ Example of Shareoption (2,x) with RLS and base VSAM
- ❑ Buffering
- ❑ Locking
- ❑ RAS
- ❑ Performance Measurements

Review of RLS

- Share options.
  - Largely ignored by RLS.
  - Exception is SHAREOPTIONS(2,x) -
    - Now defined as one user opened to the data set for non-RLS read/write and any number of users for non-RLS read. VSAM provides full read/write integrity for the non-RLS read/write user, however, the readers do not receive read integrity.
    - Or, any number of users opened for RLS read/write and any number of users for non-RLS read. VSAM provides full read/write integrity for the RLS users and no read integrity for the non-RLS readers.

Example of RLS Readers/Writers



System1

AddressSpace1

OPEN ACB macrf=(rls,out)

(read/write integrity)

SMSVSAM Dataspace

AddressSpace n

OPEN ACB1 macrf=(rls,in), rlsread=cr

(read/write integrity)

Systemn

AddressSpace1

OPEN ACB macrf=(rls,out)

(read/write integrity)

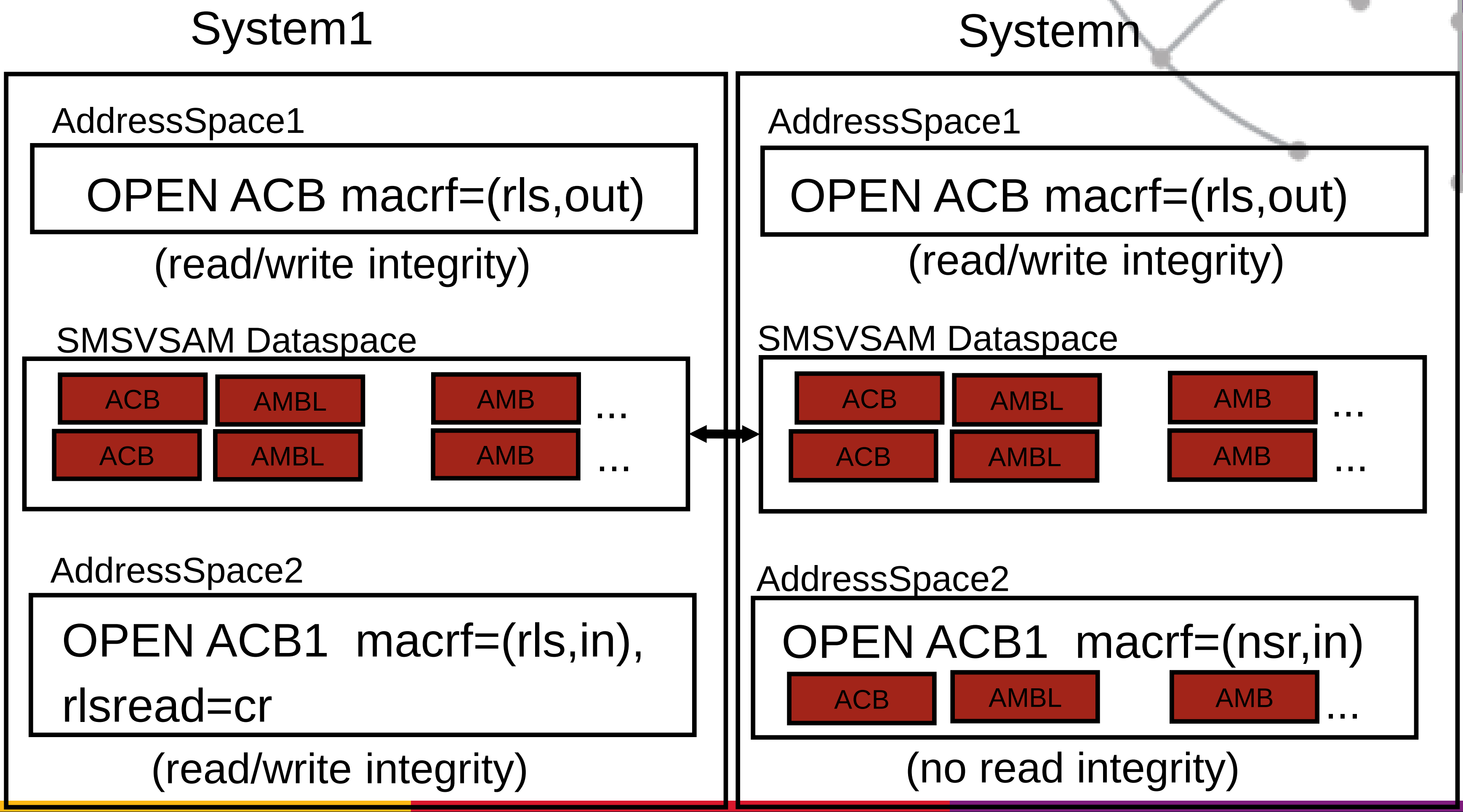
SMSVSAM Dataspace

AddressSpace n

OPEN ACB1 macrf=(rls,in), rlsread=nri

(no read integrity)

Example of Shareoption (2,x) with RLS and base VSAM



System1

AddressSpace1

OPEN ACB macrf=(rls,out)

(read/write integrity)

SMSVSAM Dataspace

AddressSpace2

OPEN ACB1 macrf=(rls,in), rlsread=cr

(read/write integrity)

Systemn

AddressSpace1

OPEN ACB macrf=(rls,out)

(read/write integrity)

SMSVSAM Dataspace

AddressSpace2

OPEN ACB1 macrf=(nsr,in)

(no read integrity)

RLS - Buffering

- VSAM now provides 4 types of buffering: ACB macrf=(NSR/LSR/GSR/**RLS**).
  - NSR - Non-Shared Resources
  - LSR - Local Shared Resources
  - GSR - Global Shared Resources
  - **RLS - Record Level Sharing**
- Each image in the sysplex has one 31 bit (default) local buffer pool, (located in a dataspace) with a current maximum size of 1.7 gig, and one 64 bit pool located in the SMSVSAM address space. Both buffer pools are managed by LRU.
- Pool sizes controlled by PARMLIB parameters: RLS_Max_Pool_Size (31 bit pool) and RLSAboveTheBarMaxPoolSize (64 bit pool).
- Buffer coherency is maintained through the use of CF cache structures and the XCF cross-invalidation function.

LRU

- The LRU for the 31 bit pool operates in the following 4 modes:
  - **Normal Mode** - Total pool size is less than 80% of RLS_Max_Pool_Size.
  - **Maintenance Mode** - Total pool size is greater than 80% and less than 120% of RLS_Max_Pool_Size.
  - **Accelerated Mode** - Total pool size is greater than 120% and less than 2 * RLS_Max_Pool_Size.
  - **Panic Mode** - Total pool size is greater than 2* RLS_Max_Pool_Size or greater than 1728M.

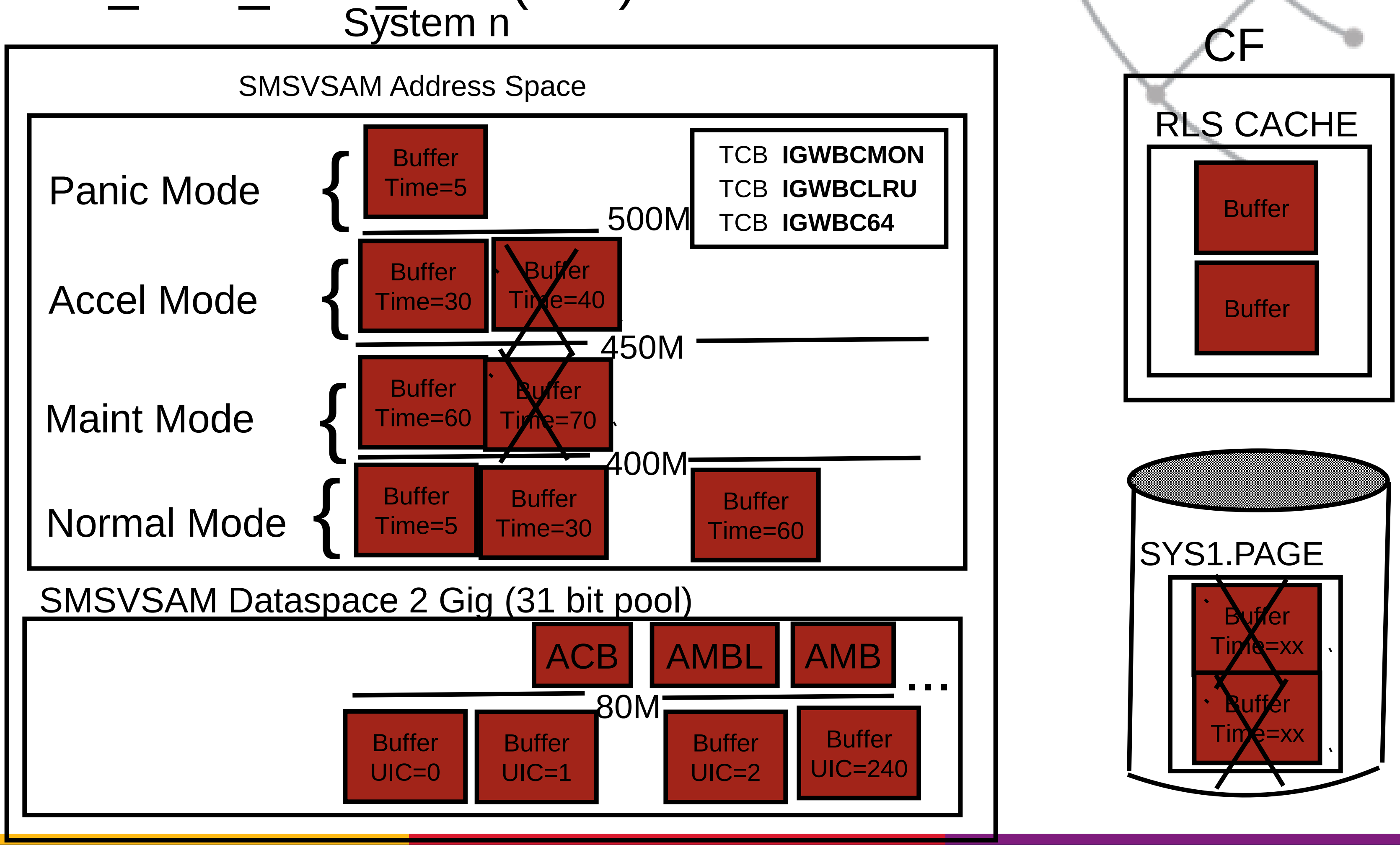
LRU

- The LRU for the 64 bit buffer pool operates in four modes:
  - **Normal Mode** - Total 64 bit pool size is less than 80% of RLSAboveTheBarMaxPoolSize.
  - **Maintenance Mode** - Total 64 bit pool size is greater than 80% and less than 90% of RLSAboveTheBarMaxPoolSize.
  - **Accelerated Mode** - Total 64 bit pool size is greater than 90% and less than 100% of RLSAboveTheBarMaxPoolSize.
  - **Panic Mode** - Total 64 bit pool size is greater than 100% of RLSAboveTheBarMaxPoolSize

LRU

- The LRU will release bit buffers as follows:
  - **Normal Mode** - Buffers stay indefinitely.
  - **Maintenance Mode** - Buffers 60 minutes or older will be released.
  - **Accelerated Mode** - Buffers 30 minutes or older will be released. Requests for new buffers will first be stolen. If there are no buffers to steal a new get block will be done.
  - **Panic Mode** - Buffers 5 minutes are older will be released. Requests for new buffers will first be stolen. If there are no buffers to steal, the request will sleep until LRU runs.

RLSAboveTheBarMaxPoolSize(500)
RLS_Max_Pool_Size(100)



Setting up Parameters/Structures sizes

- ❑ Local Buffer Pool Sizes:
  - ➢ RLS_MAX_POOL_SIZE(nnnn)   Where nnnn = (10 to 9999), anything over 1500 is treated as a maximum of 1728M.
  - ➢ RLSAboveTheBarMaxPoolSize(sysname1,nnnn) Where nnnn is either 0, or 500M to 2,000,000M
  - ➢ RLS_MaxCFFeatureLevel(Z/A)
- ❑ Pool Size values are a goal for which the LRU tries to maintain. If more buffers are required at any given time, the pool may temporarily exceed the values set.
- ❑ Real Storage   - Total amount of buffer pools should not exceed amount of real storage. A paged out buffer is immediately freed by the LRU.

Sizing the RLS Cache Structures

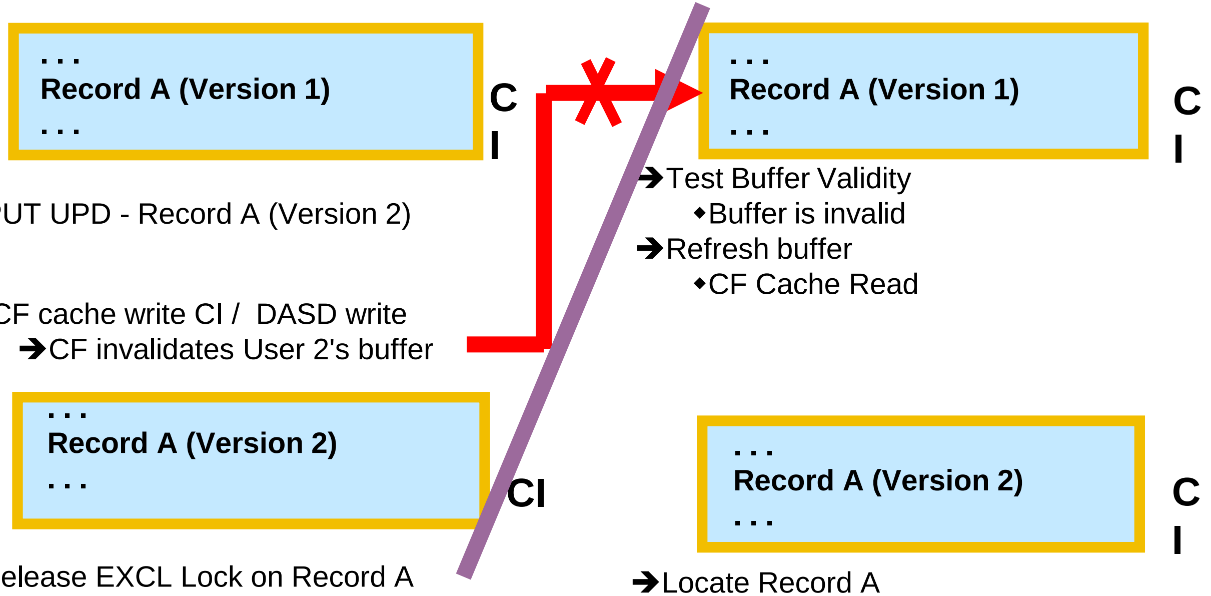
- ❑ The “ideal” cache structure size:
  - ➢ Total_Cache_Sturcture_sizes = ((RLS_Max_Pool_Size) * Number_of_SMSVSAMs_in_Sysplex) + (RLSAboveTheBarMaxPoolSize(system1) + … +RLSAboveTheBarMaxPoolSize(systemn))
  - ➢ Assumes the following:
    - ○ RLS_MaxCFFeaturelevel(A) - caching all CISIZEs RLSCFCACHE(ALL)
    - ○ Handles the “worst case” where each LPAR has cached different data.
    - ○ If more than one cache structure in the SMS STORCLAS, data sets are signed first in alternating order then based on cache activity between the individual cache structures.
    - ○ Can reduce amount of data stored in the cache via DATACLAS RLSCFCACHE(ALL/NONE/UPDATES/DIRONLY).

RLS Buffer Invalidate Example

System1

USER 1 (WRITER)

- ■GET UPD - Record A
  - ➔Locate Record A
  - ➔EXCL Lock on Record A
  - ➔Test Buffer Validity
    - ◆Buffer is valid
  - ➔Return record to caller



. . .
Record A (Version 1)
. . .

- ■PUT UPD - Record A (Version 2)
- ■ CF cache write CI / DASD write
  - ➔CF invalidates User 2's buffer

. . .
Record A (Version 2)
. . .

- ■Release EXCL Lock on Record A

System2

USER 2 (READER)

- ■GET NUP,CR - Record A
  - ➔Locate Record A
  - ➔SHR Lock on Record A
    - ◆WAIT for Lock

. . .
Record A (Version 1)
. . .

- ➔Test Buffer Validity
  - ◆Buffer is invalid
- ➔Refresh buffer
  - ◆CF Cache Read

. . .
Record A (Version 2)
. . .

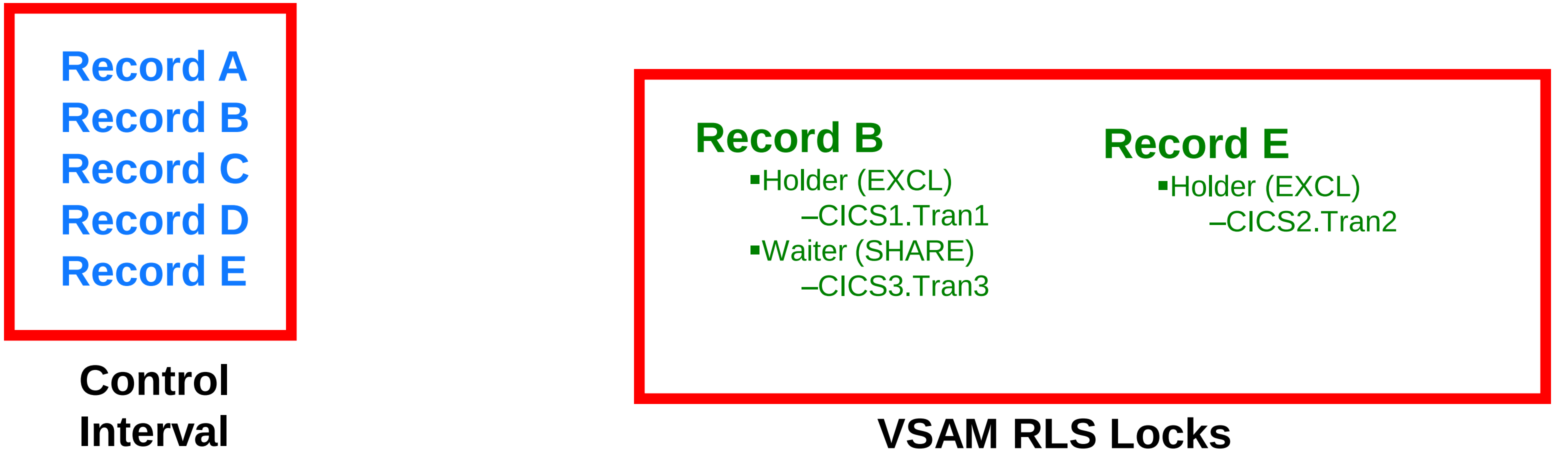
- ➔Locate Record A
- ➔Return record to caller
- ➔Release SHR Lock on Record A

RLS - Locking

- RLS serializes on a record level.
- All RLS requests will wait when contention is encountered. For CICS, FCVRWAITs represent the total time necessary to wait for the resource.
- Users updating or inserting a record will hold the lock exclusive for the duration of the write request (non-recoverable) or transaction (recoverable).
- Users reading a record will use No Read Integrity (default). No (shared) lock will be obtained:
  - ACB RLSREAD=NRI
  - //DD DD dsn=datasetname,RLS=NRI
- Users requiring read integrity will hold the lock share when Consistent Read (CR) or Consistent Read Extended (CRE) is specified. For CR, the lock is released at end of request. For CRE, the lock is released at commit time.
  - ACB RLSREAD=CR/CRE
  - //DD1 DD dsn=datasetname,RLS=CR/CRE

Example of RLS Serialization

System1	System2	System3
CICS1.Tran1	CICS2.Tran2	CICS3.Tran3
GET UPD RPL_1 (Record B)	GET UPD RPL_2 (Record E)	GET CR RPL_3 (Record B)
		–Waits for record lock



Record A
Record B
Record C
Record D
Record E

Control Interval

Record B

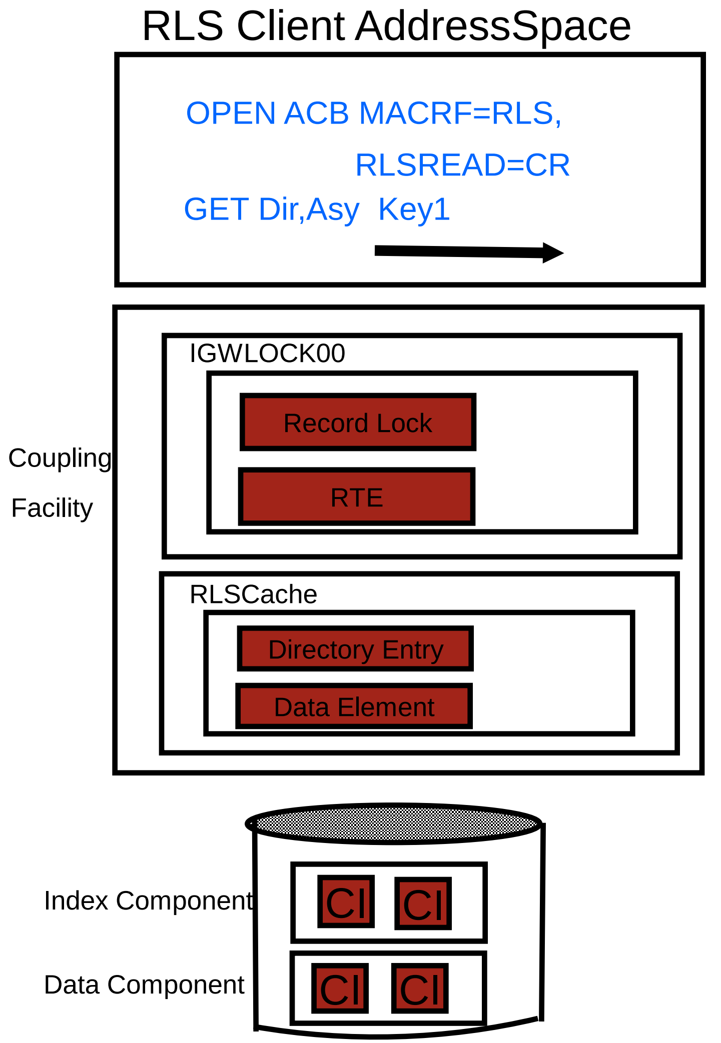
- Holder (EXCL)
  - –CICS1.Tran1
- Waiter (SHARE)
  - –CICS3.Tran3

Record E

- Holder (EXCL)
  - –CICS2.Tran2

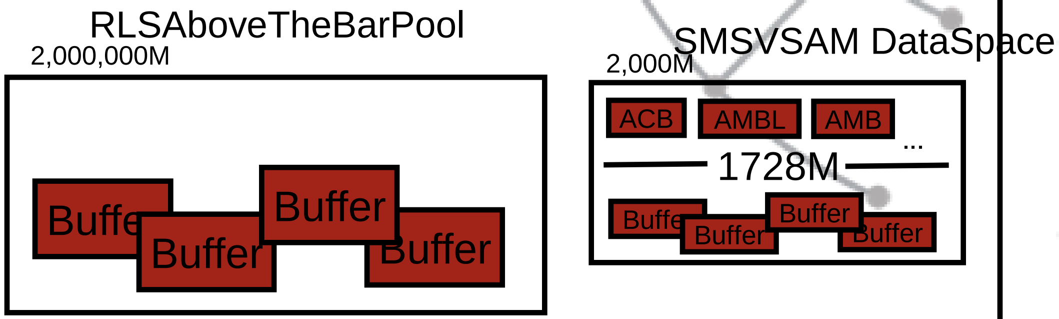
VSAM RLS Locks

Overview of Get Path



```
OPEN ACB MACRF=RLS,
            RLSREAD=CR
GET Dir,Asy  Key1
```

SMSVSAM Address Space



Index_search:

(Call BMF to locate Index CIs, if no_buffer Call SCM to read from CF or DASD)

Lock_Record;

(Call SMLS to obtain record lock)

Get_Data_CI:

(Call BMF to locate Data CI, If no_buffer Call SCM to read from CF or DASD

UnLock_Record:

(Call SMLS to release record lock)

RLS - RAS

- RLS provides extensive first time data capture for logic errors.
  - Many "health checks" in the code which produce ABEND0F4 dumps to capture the problem at the earliest possible point.
  - Extensive logging and tracing facilities.
  - RAS is considered a high priority element of RLS design..
- End result:
  - Problems easier to debug..
  - Much less likely for broken data sets or data integrity problems.

RLS Performance Measurements

- SMF 62 and 64
  - SMF 62 – Created by RLS OPEN for each ACB.
  - SMF 64 – Created by RLS EOV and CLOSE for each ACB. Stats are on an ACB level.
- SMF 42 Subtypes 15, 16, 17, 18, 19
  - **Subtype 15** - RLS statistics by Storage Class
  - **Subtype 16** - RLS statistics by Data set
    - Must use V SMS,MONDS(spherename),ON to collect subtype 16 statistics.
  - **Subtype 17** - RLS locking Statistics for IGWLOCK00
  - **Subtype 18** - RLS caching Statistics
  - **Subtype 19** - BMF statistics
- SMF formatter soon to be available as part of our IPCS VERBX SMSXDATA
- Note: Only one system in the sysplex collects the SMF 42 records. The system collecting the records is displayed in the D SMS,SMSVSAM operator command.

RLS/TVS Configuration Changes

Configuration Changes

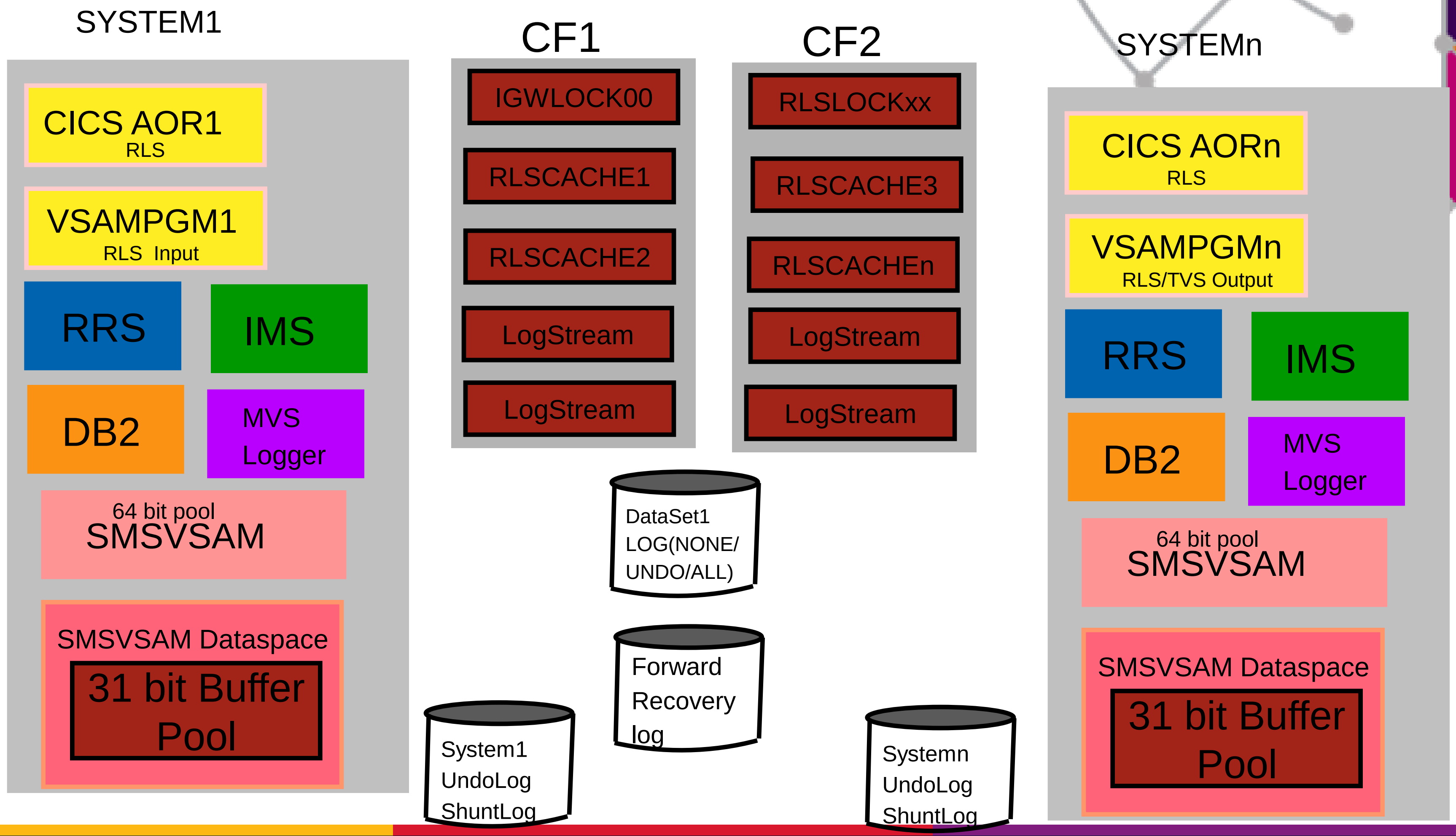
- Update CFRM policy to define lock, cache, list, log structures.
  - See DFSMSdfp Storage Administration Reference for sizing info.
- Update SYS1.PARMLIB(IGDSMSxx) with RLS/TVS parameters.
  - See MVS Initialization and Tuning.
- Define new SHCDSs (Share Control Data Sets).
  - See DFSMSdfp Storage Administration Reference.
- Update SMS configuration for Cache Sets.
  - See DFSMSdfp Storage Administration Reference.
- Update data sets with LOG(NONE/UNDO/ALL) and LOGSTREAMID.
  - See Access Methods Services for ICF.

System Requirements - PARMLIB Changes

SYS1.PARMLIB(IGDSMSxx)

SMS ACDS(acds)	COMMDS(commds)
INTERVAL(nnn|15)	DINTERVAL(nnn|150)
REVERIFY(YES|NO)	ACSDEFAULTS(YES|NO)
SYSTEMS(8|32)	TRACE(OFF|ON)
SIZE(nnnnnK|M)	TYPE(ALL|ERROR)
JOBNAME(jobname|*)	ASID(asid|*)
SELECT(event,event....)	DESELECT(event,event....)
DSNTYPE(LIBRARY|PDS)	DSSTIMEOUT(nnn|0)
CA_RECLAIM(NONE/DATACLAS)	
RLS_MAXCFFEATURELEVEL(A|Z)	RLS_MAX_POOL_SIZE(nnn|100)
RLSINIT(NO|YES)	SMF_TIME(NO|YES)
CF_TIME(nnn|3600)	BMF_TIME(nnn|3600)
CACHETIME(nnn|3600)	DEADLOCK_DETECTION(iii|15,kkk|4)
RLSTMOUT(nnn|0)	RLSAboveTheBarMaxPoolSIze(system,size)
RLSFixedPoolSize(system.size)	SYSNAME(sys1,sys2,...)
TVSNAME(nnn1,nnn2....)	MAXLOCKS(max|0,incr|0)
TV_START_TYPE(WARM|COLD,WARM|COLD...)	AKP(nnn|1000,nnn|1000)
LOG_OF_LOGS(logstream)	QTIMEOUT(nnn|300)

SYSPLEX with SMSVSAM (and TVS) - Example



SYSTEM1

CICS AOR1
RLS

VSAMPGM1
RLS Input

RRS

IMS

DB2

MVS
Logger

64 bit pool
SMSVSAM

SMSVSAM Dataspace
31 bit Buffer Pool

CF1

IGWLOCK00

RLSCACHE1

RLSCACHE2

LogStream

LogStream

CF2

RLSLOCKxx

RLSCACHE3

RLSCACHEn

LogStream

LogStream

DataSet1
LOG(NONE/
UNDO/ALL)

Forward
Recovery
log

System1
UndoLog
ShuntLog

Systemn
UndoLog
ShuntLog

SYSTEMn

CICS AORn
RLS

VSAMPGMn
RLS/TVS Output

RRS

IMS

DB2

MVS
Logger

64 bit pool
SMSVSAM

SMSVSAM Dataspace
31 bit Buffer Pool

SMSVSAM Initialization

SMSVSAM Initialization

```
IGW619I ACTIVE SHARE CONTROL DATA SET 209
SYS1.DFPSHCDS.ACTIVE2.VSPLXPK ADDED.
 IGW619I SPARE SHARE CONTROL DATA SET 283
 SYS1.DFPSHCDS.SPARE.VSPLXPK ADDED.
IGW321I Running Protocol 4
IXL014I IXLCONN REQUEST FOR STRUCTURE IGWLOCK00 313
WAS SUCCESSFUL.  JOBNAME: SMSVSAM ASID: 0009
CONNECTOR NAME: SYSTEM1 CFNAME: FACIL01
IGW321I System Ordinal is 1
IGW453I SMSVSAM ADDRESS SPACE HAS SUCCESSFULLY 316
CONNECTED TO DFSMS LOCK STRUCTURE IGWLOCK00
IGW321I No retained locks
IGW321I 0 RLS Sphere Record Table Entries read
IGW321I 0 RLS Sphere Record Table Entries deleted
IGW321I No Spheres in lost locks
```

SMSVSAM Initialization (co nt.)

```
IGW414I SMSVSAM SERVER ADDRESS SPACE IS NOW ACTIVE.

IGW467I DFSMS RLS_MAX_POOL_SIZE PARMLIB VALUE SET DURING 354

SMSVSAM ADDRESS SPACE INITIALIZATION ON SYSTEM: SYSTEM1

CURRENT VALUE: 100

IGW467I DFSMS DEADLOCK_DETECTION PARMLIB VALUE SET DURING 355

SMSVSAM ADDRESS SPACE INITIALIZATION ON SYSTEM: SYSTEM1

THIS SYSTEM IS OPERATING AS THE GLOBAL DEADLOCK PROCESSOR.

CURRENT VALUE: 15  4

.

.

IGW467I DFSMS RLS_MAXCFFEATURELEVEL PARMLIB VALUE SET DURING

SMSVSAM ADDRESS SPACE INITIALIZATION ON SYSTEM: SYSTEM1

CURRENT VALUE: Z
```

SMSMVSAM Initialization (with TVS) - (cont.)

SYSTEM1

```
SYSTEM1  05008 11:34:01.17  IGW467I DFSMS TVSNAME PARMLIB VALUE SET DURING 578
                                    SMSVSAM ADDRESS SPACE INITIALIZATION ON SYSTEM:
                                    SYSTEM1 TVSNAME: IGWTV001
SYSTEM1  05008 11:34:01.18  IGW467I DFSMS TRANSACTIONAL VSAM UNDO LOG PARMLIB VALUE SET
                                    DURING SMSVSAM ADDRESS SPACE INITIALIZATION ON SYSTEM:
                                   SYSTEM1  UNDO LOGSTREAM NAME:
                                    IGWTV001.IGWLOG.SYSLOG
 SYSTEM1  05008 11:34:01.18 IGW467I DFSMS TRANSACTIONAL VSAM SHUNT LOG PARMLIB VALUE SET
                                    DURING SMSVSAM ADDRESS SPACE INITIALIZATION ON SYSTEM:
                                    SYSTEM1 SHUNT LOGSTREAM NAME:
                                    IGWTV001.IGWSHUNT.SHUNTLOG
                                    .
                                    .
```

System Requirements - SMSVSAM Initialization

SYSTEM1

```
SYSTEM1  05008 11:34:01.18 IGW467I DFSMS TRANSACTIONAL VSAM TVS_START_TYPE PARMLIB
                                   VALUE SET DURING SMSVSAM ADDRESS SPACE INITIALIZATION
                                   ON SYSTEM:  SYSTEM1  TVSNAME VALUE:  IGWTV001
                                   CURRENT VALUE: WARM  1
 SYSTEM1  05008 11:34:06.29 IGW860I TRANSACTIONAL VSAM HAS SUCCESSFULLY REGISTERED
                                   WITH RLS
SYSTEM1  05008 11:35:36.63 IGW865I TRANSACTIONAL  VSAM INITIALIZATION IS COMPLETE.
SYSTEM1  05008 11:35:36.65 IGW886I 0 RESTART TASKS WILL BE PROCESSED DURING
                                   TRANSACTIONAL RESTART PROCESSING
SYSTEM1  05008 11:35:36.65 IGW866I TRANSACTIONAL  VSAM RESTART PROCESSING IS COMPLETE.
                                   .
                                   .
                                   .
```

SMSVSAM Commands

SMSVSAM Display Commands

```
D SMS[,
    [,CFCACHE(structurename|*)                ]
    [,CFLS                                    ]
    [,CFVOL(volid)                            ]
    [,DSNAME(dsn){,WTOR}                      ]
    [,JOB(jobname){,WTOR}                     ]
    [,LOG({logstreamid|ALL}{,WTOR}            ]
    [,MONDS(specmask|*)                       ]
    [,SHCDS                                   ]
    [,SHUNTED,{SPHERE(sphere)|UR({urid|ALL}}{,WTOR}]
    [,SMSVSAM[,ALL]                           ]
```

SMSVSAM Display Commands (cont)

```
 D SMS[,
      [,TRANVSAM[,ALL][,ALLLOGS][,WTOR]              ]
      [,URID({urid|ALL}){,WTOR}                      ]
D SMS,SMSVSAM,DIAG(CONTENTION)
```

D SMS,SMSVSAM (example)

D SMS,SMSVSAM

```
DISPLAY SMS,SMSVSAM - SERVER STATUS
 SYSNAME:  SYSTEM1    AVAILABLE ASID: 0033 STEP: SmsVsamInitComplete

DISPLAY SMS,SMSVSAM - JOB STATUS
 SUBSYSTEMS CONNECTED:       1 BATCH:       1

DISPLAY SMS,SMSVSAM - LOCK TABLE STATUS (IGWLOCK00)
 CONNECT STATUS:
  SYSNAME:  SYSTEM1  ACTIVE          RSN: 02010407 RbldNotActive

 COMPOSITE STATUS:
  ORIGINAL STRUCTURE: NOT VOLATILE          FAILURE ISOLATED
  NEW      STRUCTURE: NOT VOLATILE          FAILURE ISOLATED

 STRUCTURE STATUS:
 SYSNAME: SYSTEM1   Duplex
```

System Requirements - SMSVSAM Displays

SYSTEM1

```
- 13.19.03 SYSTEM1          d sms,tranvsam
  13.19.04 SYSTEM1          IEE932I 023
  IGW800I 13.19.04 DISPLAY SMS,TRANSACTIONAL VSAM
  DISPLAY SMS,TRANSACTIONAL VSAM - SERVER STATUS
   System      TVSNAME  State    Rrs       #Urs    Start               AKP    QtimeOut
   ---------    ---------    ------    -----     --------    ---------            --------  --------
   SYSTEM1   IGWTV001 ACTIVE REG         0       WARM/WARM      200      400

  DISPLAY SMS,TRANSACTIONAL VSAM - LOGSTREAM STATUS
   LogStreamName                          State        Type          Connect Status
   --------------------------             ----------     ----------      --------------
   IGWTV001.IGWLOG.SYSLOG          Enabled    UnDoLog      Connected
   IGWTV001.IGWSHUNT.SHUNTLOG  Enabled    ShuntLog     Connected
```

SMSVSAM Vary Commands

```
V SMS,{CFCACHE(cachename),{ENABLE|E }       }
 {                  {QUIESCE|Q}       }
 {CFVOL(volid),{ENABLE|E }            }
 {            {QUIESCE|Q}             }
 {MONDS(dsname[,dsname...]),{ON|OFF}    }
 {SHCDS(shcdsname),{NEW    }          }
 {                {NEWSPARE}          }
 {                {DELETE  }          }
 {SMSVSAM,{ACTIVE                     }
 {        {FALLBACK                   }
 {        {TERMINATESERVER            }
 {        {FORCEDELETELOCKSTRUCTURE     }
```

SMSVSAM Vary Commands

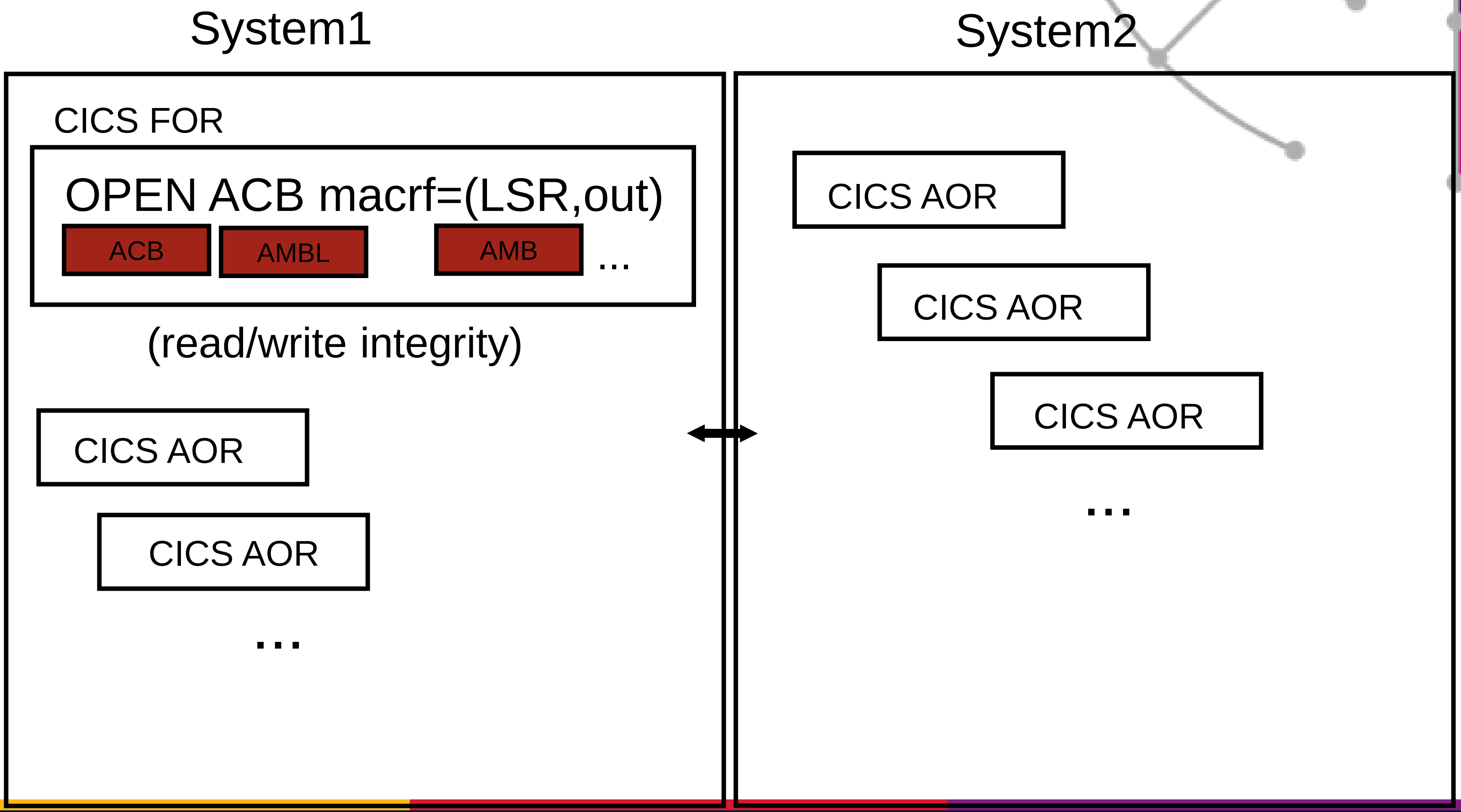
```
V SMS,{TRANVSAM({tvsname|ALL}){,{QUIESCE|Q}}    }
{            {,{ENABLE|E }}    }
{            {,{DISABLE|D}}    }
{                         }
{LOG(logstreamid){{,QUIESCE|Q}}       }
{         {,{ENABLE|E }}          }
{         {,{DISABLE|D}}          }
{                         }
{SMSVSAM,SPHERE(sphere){,{QUIESCE|Q}}    }
{            {,{ENABLE|E }}    }
{                         }
{TRANVSAM(tvsname),PEERRECOVERY{,{ACTIVE|A  }}}
{                {,ACTIVEFORCE }}
{                {,{INACTIVE|I}}}
```

RLS/CICS Environment

RLS/CICS Environment

- ❑ CICS and VSAM FOR configuration.
  - ❑ Advantages and disadvantages of the FOR/AOR configuration.
- ❑ CICS and RLS configuration.
  - ❑ Advantages and disadvantages of the CICS/RLS configuration.
- ❑ RLS/CICS data recovery.
  - ➢ Recoverable data sets.
  - ➢ Recoverable subsystems.
  - ➢ Retained locks.
  - ➢ Lost locks.
  - ➢ IDCAMS SHCDS commands
  - ➢ Quiesce for COPY/Quiesce for BWO interface.
- ❑ RLS/CICS Automation Enhancements
  - ➢ Sphere Quiesce/Enable

CICS FOR/AOR Cross System Configuration



System1

CICS FOR

OPEN ACB macrf=(LSR,out)

ACB

AMBL

AMB

...

(read/write integrity)

CICS AOR

CICS AOR

...

System2

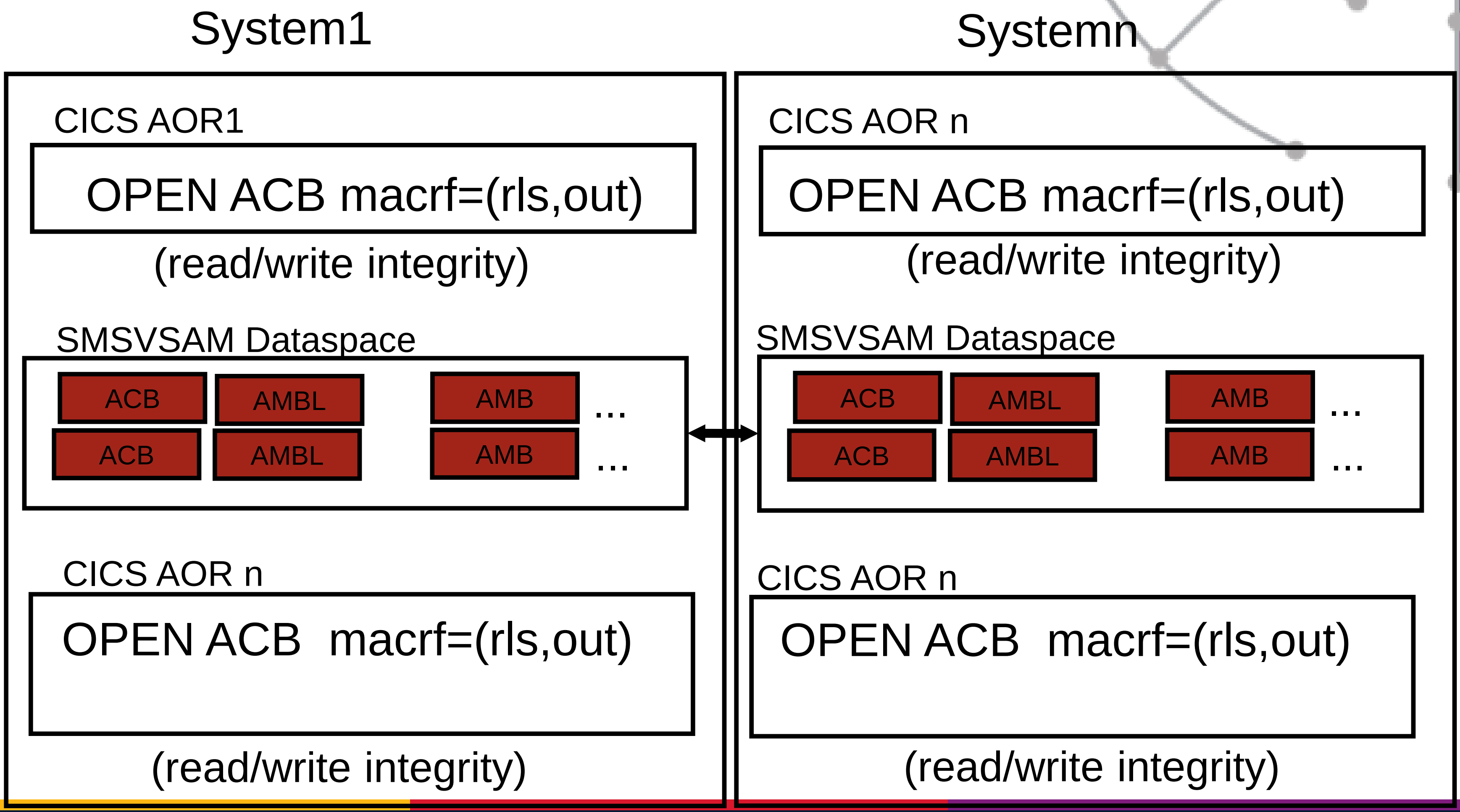
CICS AOR

CICS AOR

CICS AOR

...

RLS/CICS Configuration



System1

CICS AOR1

OPEN ACB macrf=(rls,out)

(read/write integrity)

SMSVSAM Dataspace

ACB AMBL AMB …

ACB AMBL AMB …

CICS AOR n

OPEN ACB macrf=(rls,out)

(read/write integrity)

Systemn

CICS AOR n

OPEN ACB macrf=(rls,out)

(read/write integrity)

SMSVSAM Dataspace

ACB AMBL AMB …

ACB AMBL AMB …

CICS AOR n

OPEN ACB macrf=(rls,out)

(read/write integrity)

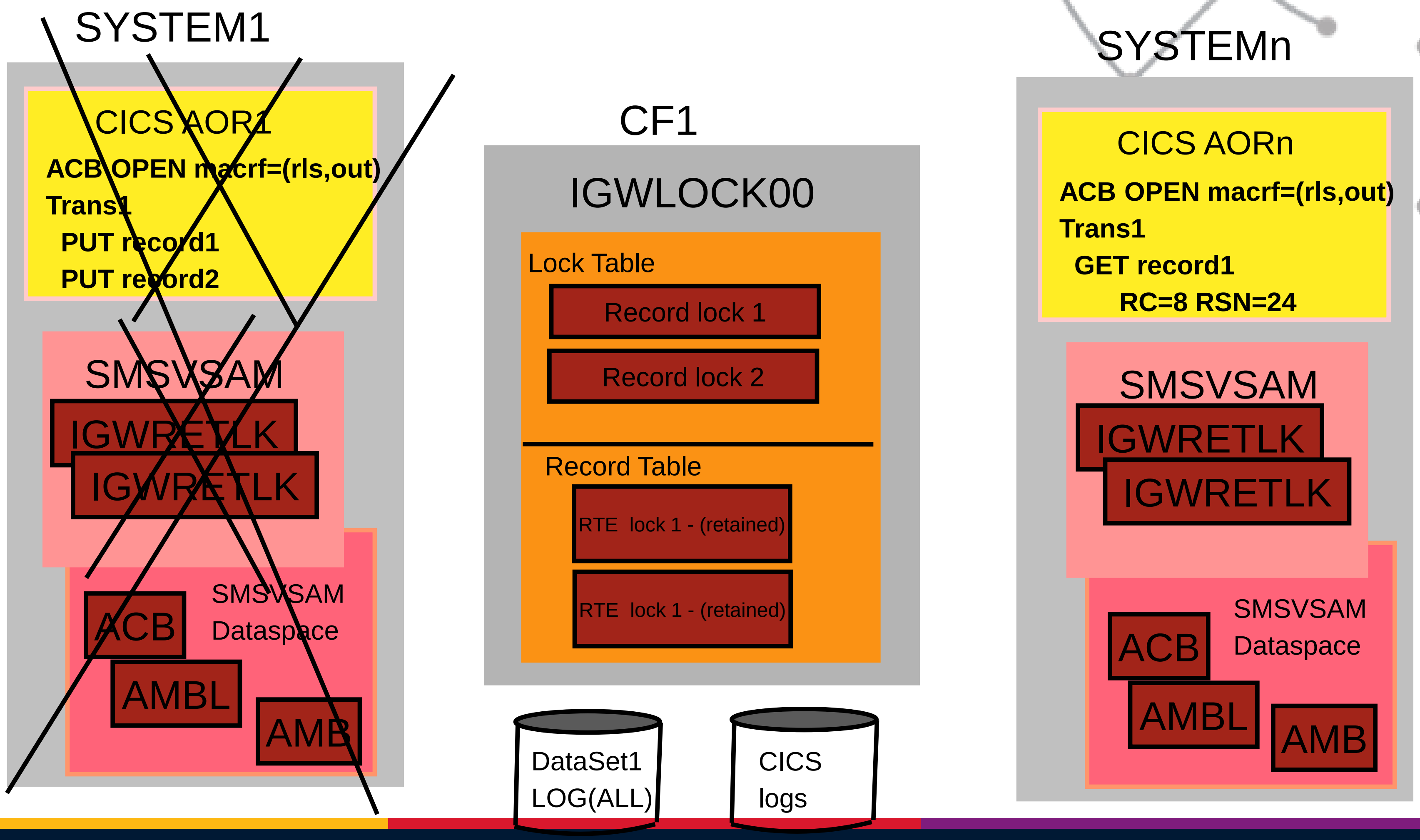
RLS/CICS Data Recovery

- Recoverable data sets
  - Defined as LOG(UNDO/ALL) in the catalog.
    - UNDO - backout logging performed by CICS (or TVS).
    - ALL - both backout and forward recovery logging by CICS (or TVS).
      - ✓ Must also have a LOGSTREAMID(forwardrecoverylog) also defined in the catalog.
- Non-Recoverable data sets
  - Defined as LOG(NONE) in the catalog.
    - No logging performed by CICS (or TVS).
- Recoverable Subsystems.
  - CICS (and TVS) must register with the SMSVSAM address space with a "subsystemname" so that locks obtained by that subsystem can be tracked.

RLS/CICS Data Recovery

- Retained locks
  - Record locks are converted to "retained" in the event of a failure. The "owning" subsystem is the only subsystem that may access the record locks during recovery. All other subsystems or RLS applications will received a retained lock error in the RPL.
  - Failures include: CICS region failure, SMSVSAM failure, system failure, backout failure (CICS or TVS).
- Lost Locks
  - A recoverable data set which was open at the time of a CF failure containing the lock structure, and one or more SMSVSAM failures (i.e. double failure scenario).
  - Only the failing subsystems may open the file and recover the lost lock condition. All other RLS opens will be failed until the data set has been fully recovered. Existing sharers will receive lost lock errors for all records in the data set.

Retained Lock Example



SYSTEM1

CICS AOR1

ACB OPEN macrf=(rls,out)

Trans1

PUT record1

PUT record2

SMSVSAM

IGWRETLK

IGWRETLK

ACB

SMSVSAM Dataspace

AMBL

AMB

CF1

IGWLOCK00

Lock Table

Record lock 1

Record lock 2

Record Table

RTE lock 1 - (retained)

RTE lock 1 - (retained)

DataSet1 LOG(ALL)

CICS logs

SYSTEMn

CICS AORn

ACB OPEN macrf=(rls,out)

Trans1

GET record1

RC=8 RSN=24

SMSVSAM

IGWRETLK

IGWRETLK

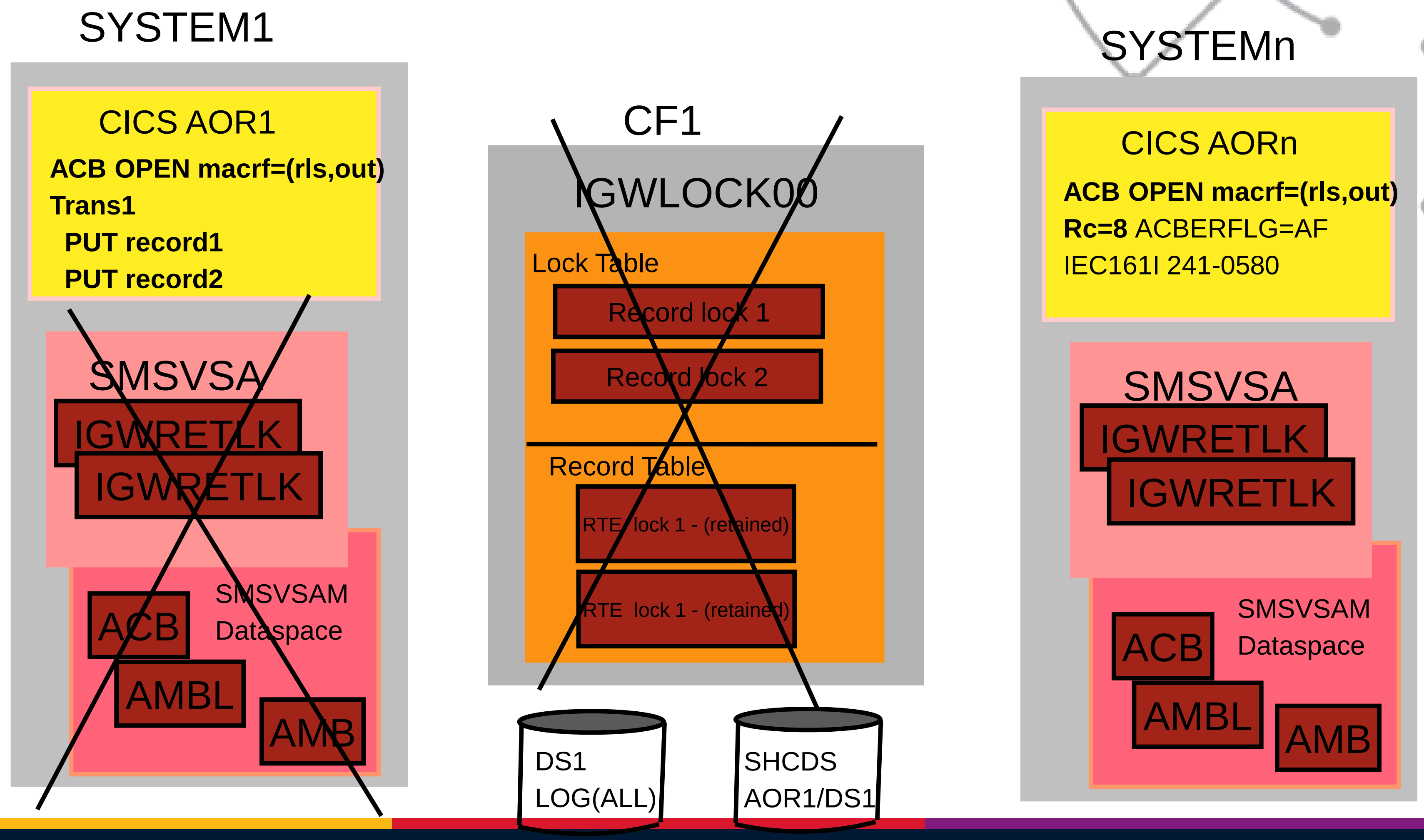
ACB

SMSVSAM Dataspace

AMBL

AMB

Lost Lock Example



SYSTEM1

CICS AOR1

ACB OPEN macrf=(rls,out)

Trans1

PUT record1

PUT record2

SMSVSA

IGWRETLK

IGWRETLK

SMSVSAM Dataspace

ACB

AMBL

AMB

CF1

IGWLOCK00

Lock Table

Record lock 1

Record lock 2

Record Table

RTE lock 1 - (retained)

RTE lock 1 - (retained)

DS1 LOG(ALL)

SHCDS AOR1/DS1

SYSTEMn

CICS AORn

ACB OPEN macrf=(rls,out)

Rc=8 ACBERFLG=AF

IEC161I 241-0580

SMSVSA

IGWRETLK

IGWRETLK

SMSVSAM Dataspace

ACB

AMBL

AMB

I
G
W
R
E
T
L
K

SHCDS IDCAMS Commands

```
SHCDS    {{LISTDS(base_cluster_name) {JOBS}} |
          {LISTSUBSYS(subsystem_name|ALL)} |
          {LISTSUBSYSDS(subsystem_name)} |
          {LISTRECOVERY(base_cluster_name|ALL)} |
          {LISTALL} |
          {FRSETRR(base_cluster_name)} |
          {FRUNBIND(base_cluster_name)} |
          {FRBIND(base_cluster_name)} |
            {FRRESETRR(base_cluster_name)} |
 {FRDELETEUNBOUNDLOCKS(base_cluster_name)} |
 {PERMITNONRLSUPDATE(base_cluster_name)} |
 {DENYNONRLSUPDATE(base_cluster_name)} |
 {REMOVESUBSYS(subsystem_name)} |
 {CFREPAIR({INFILE(ddname) |
            INDATASET(datasetname)}
```

SHCDS Commands *(continued)*

```
      {LIST|NOLIST})}
{CFRESET({INFILE(ddname) |
      INDATASET(datasetname)}
      {LIST|NOLIST})}
{CFREPAIRDS({base_cluster_name |
       {partially_qualified_base_cluster_name)
{CFRESETDS({base_cluster_name |
       {partially_qualified_base_cluster_name)
{LISTSHUNTED {SPHERE(base_cluster_name) |
        URID(urid)         |
        DATA(urid)}}
{RETRY {SPHERE(base_cluster_name) |
    URID(urid)}}
{PURGE {SPHERE(base_cluster_name) |
    URID(urid)}}
```

SHCDS Commands *(continued)*

SHCDS LISTSTAT(‘dsname’) - Displays point in time statistics across the sysplex

```
LIST STATISTICS (LISTSTAT):
CLUSTER---------HL1.KSDS0002
 DATA-------HL1.KSDS0002.DATA
  TOTAL RECORDS---------       100     CI SPLITS-------------          0
  RECORDS DELETED-------         0     CA SPLITS-------------          0
  RECORDS INSERTED------         0     EXCPS-----------------          3
  RECORDS UPDATED-------         0     EXTENTS-------------------      1
  RECORDS RETRIEVED-----         0     FREE SPACE----------      7233536
  HI-A-RBA------------     7372800     HI-U-RBA------------       737280
 INDEX------HL1.KSDS0002.INDEX
  TOTAL RECORDS---------         1     CI SPLITS-------------          0
  CA RECLAIMS-----------         0     CA SPLITS-------------          0
  RECLAIMED-CA REUSES---         0     EXCPS-----------------          3
  RECORDS UPDATED-------         0     EXTENTS-------------------      1
  RECORDS RETRIEVED-----         0     FREE SPACE----------       505856
  HI-A-RBA------------      506880     HI-U-RBA------------         1024
  HI-LEVEL-RBA----------         0     INDEX LEVELS---------------     1
```

SHCDS Example

```
                    ISPF Command Shell
Enter TSO or Workstation commands below:

===>   SHCDS LISTSUBSYS(aor1)
----- LISTING FROM SHCDS ----- IDCSH03 ----------------------------------------------------------------------------------------------------------
                                            RECOVERY     LOCKS     LOCKS       LOCKS
 SUBSYSTEM NAME     STATUS                NEEDED          HELD        WAITING     RETAINED
  ________________   ______________        _______        _____       _______     _________
  AOR1                ONLINE--FAILED        YES              0             0             1
  DATA SETS IN LOST LOCKS------------         0
  DATA SETS IN NON-RLS UPDATE STATE--         0
  TRANSACTION COUNT------------------         1
 ***
```

SHCDS Example

ISPF Command Shell

Enter TSO or Workstation commands below:

```
===>   SHCDS LISTDS('dataset1*')

----- LISTING FROM SHCDS ----- IDCSH02 ----------------------------------------------------------------------------------------------------

 DATA SET NAME----dataset1

   CACHE STRUCTURE----CACHE01

   RETAINED LOCKS---------YES   NON-RLS UPDATE PERMITTED---------NO

   LOST LOCKS--------------NO   PERMIT FIRST TIME----------------NO

   LOCKS NOT BOUND---------NO   FORWARD RECOVERY REQUIRED--------NO

   RECOVERABLE------------YES
```

SHCDS Example (cont.)

```
                    SHARING SUBSYSTEM STATUS

SUBSYSTEM   SUBSYSTEM       RETAINED      LOST      NON-RLS UPDATE

NAME         STATUS          LOCKS         LOCKS     PERMITTED

---------     --------------     ----------------     ----------     --------------------------

AOR1          ONLINE--FAILED   YES            NO          NO

***
```

RLS/CICS Data Recovery

- Quiesce for COPY/BWO interface:
  - Called by DSS to communicate with CICS (via SMSVSAM) to inform CICS when a DSS copy/backup begins and ends.
  - Allows DSS to either take a "sharp" copy (via the QUICOPY interface) or a "fuzzy" copy (via the QUIBWO interface. BWO is specified by the BWO(TYPECICS) attribute in the catalog):
    - For the QUICOPY interface (default), CICS will halt new update transactions when the DSS copy starts. New opens will not be failed and new update transaction will be failed with AFCK abends. The DSS job will be failed for non CICS applications (e.g. batch) which have the data open for output by RLS.
    - For the QUIBWO interface, CICS will log the start of the backup for LOG(ALL) data sets, new opens will be failed, however, update transactions will be allowed. CICS will write the after images of the updated records to the LOGSTREAMID (forward recovery log).
  - For both the QUICOPY/QUIBWO interfaces, CICSVR can perform a point in time recovery for LOG(ALL) data sets.
  - The BWO interface may take longer to recover the data since updates have occurred during the backup. DSS features (i.e.Concurrent Copy, Flash Copy) can significantly reduce the copy time frame.
  - **The Quiesce for COPY/BWO interfaces can be bypassed by setting DSS PATCH BYTE(45) = FF**
  - The CICS XFCVSDS exit can be used to cancel RLS quiesce events.

RLS/CICS Automation Enhancements

- QUIESCE (QUICLOSE) / ENABLE (QUIOPEN) Interface:
  - QUICLOSE interface is used by CICS to fully close a data set around the sysplex.
    - SMSVSAM drives CICS quiesce exit which issues closes for all regions open to the data set.
    - SMSVSAM updates the catalog and marks the data set as quiesced.
    - RLS opens against a quiesced data set will be failed.
  - QUIOPEN interface is used by CICS to enable a data set to be reopened for RLS use.
    - SMSVSAM drives CICS quiesce exit to ALL CICS regions registered with RLS.
    - SMSVSAM updates the catalog and marks the data set as unquiesced.
  - Invoked with the following commands:
    - V SMS,SMSVSAM,SPHERE(spherename),Q
    - V SMS,SMSVSAM,SPHERE(spherename),E
    - F cicsname,CEMT SET DSN(spherename.*),QUI
    - F cicsname,CEMT SET DSN(spherename.*),UNQ

Batch Sharing

Batch Application Requirements - Data Set Changes

- Data sets accessed by RLS must have a LOG parm specifed in the catalog. Valid values are:
  - LOG(NONE) - Non-recoverable data set. Can be opened for input/output by any RLS application. TVS not required.
  - LOG(UNDO) - Recoverable data set requiring backout (UNDO) logging. Can be opened for input/output for RLS by CICS and/or nonCICS RLS applications running on a z/OS system with the TVS feature installed.
  - LOG(ALL) - Recoverable data set requiring both backout (undo) and forward recovery logging. Can be opened for input/output for RLS by CICS and/or nonciCS RLS applications running on a z/OS system with the TVS feature installed.
    - ✓ Must also provide a LOGSTREAMID(logstreamname)

Batch Application Requirements – Program Changes

- ❑ Application opens the data set for RLS:
  - ➢ACB RLSREAD=NRI
  - ➢ //DD DD dsn=datasetname,RLS=NRI
- ❑ Startbrowse must proceed all readnext or prevnext APIs.
- ❑ Batch step must be restartable at the point in time of a failure.
- ❑ Recoverable data sets (LOG(UNDO/ALL) require Transactional VSAM (TVS) when opening for output.

Transactional VSAM (TVS)

Transactional VSAM (TVS)

- Enhance VSAM Record Level Sharing (RLS) to provide data recovery capabilities for any application exploiting VSAM RLS.
- VSAM RLS data recovery capabilities include:
  - Transactional recovery
  - Data set recovery
- VSAM RLS becomes a "transactionalized" access method, or is now referred to as "Transactional VSAM" (TVS).

System Requirements - Hardware/Software Requirements

- Parallel sysplex running z/OS 1.4 or higher with VSAM RLS implemented.
- z/OS Transactional VSAM (separately priced feature).
- z/OS RRMS implemented.
- z/OS System Logger implemented.
- CICS VSAM Recovery (CICSVR) Utility (optional)

Application Requirements - Data Set Define/Alter Example

```
DEFINE CLUSTER (NAME(recoverabledataset)                -
          RECORDSIZE(100 100)                 -
          STORCLAS(storclasname)                -
          FSPC(20 20)                  -
          LOG (ALL)                -
          SHAREOPTIONS(2 3)                -
          LOGSTREAMID(forwardrecoverylog)      -
          CISZ(512)             -
          KEYS(06 8) INDEXED                -
         )                              -
     DATA(NAME(recoverabledataset.DATA) -
          VOLUME(volser)                   -
          TRACKS (1,1))    -
     INDEX(NAME(recoverabledataset.INDEX) -
          VOLUME(volser)                   -
          TRACKS (1,1))
```

Application Requirements – RLS/TVS Access Options

- Transactional VSAM support occurs when:
  - ACB MACRF=(RLS,OUT) for recoverable data set (LOG(UNDO|ALL))
  - ACB MACRF=(RLS,IN), RLSREAD=CRE .
  - //ddname DD DSN=recoverabledatasetname,DISP=shr,RLS=(CR|NRI) and ACB MACRF=(OUT)
  - //ddname DD DSN=datasetname,DISP=shr,RLS=CRE and ACB MACRF=(IN)

Application Requirements - Transactional Recovery

- ❑ RLS applications opening recoverable data sets on z/OS with the TVS feature installed, should be modified to add SRRCMIT and SRRBACK interfaces.
- ❑ SRRCMIT and SRRBACK will either commit or backout the unit of recovery (UR) provided by SMSVSAM on behalf of the VSAM RLS application.
- ❑ Explicitly committing or backing out the UR will release record level locks in a timely fashion. Failure to do so may impact other sharers of the data set.
- ❑ SMSVSAM will implicitly issue a commit or backout at EOT, if the VSAM application fails to do so.

Application Requirements - Supported Languages

- High level language support for RLS and RRS interfaces:
  - PLI
  - C & C++
  - COBOL
  - Assembler

Application Requirements - Explicit Commit Example

```
//ddname  DD  DSN=Recoverabledatasetname,DISP=SHR

//step1   EXEC  PGM=vsamrlspgm

Begin JOB Step   ------------------------------------ No locks held

OPEN  ACB MACRF=(RLS,OUT)

(UR1)

GET UPD record 1----------------------------------- Obtain an exclusive lock on record 1

PUT UPD  record 1 ---------------------------------  Lock on record 1 remains held

GET repeatable read record n-------------------  Obtain a shared lock on record n

PUT ADD record n+1------------------------------- Obtain an exclusive lock on record n+1

GET UPD record 2 --------------------------------- Obtain an exclusive lock on record 2

PUT UPD record 2 ---------------------------------- Lock on record 2 remains held

Call SRRCMIT ---------------------------------------- Commit changes, all locks released .

CLOSE

End of JOB Step
```

Application Requirements - Implicit Commit Example

//ddname DD DSN=Recoverabledatasetname,DISP=SHR

//step1   EXEC PGM=vsamrlspgm

Begin JOB Step --------------------------------------- No locks held

OPEN ACB MACRF=(RLS,OUT)

(UR1)

GET UPD record 1------------------------------------ Obtain an exclusive lock on record 1

PUT UPD record 1 ---------------------------------- Lock on record 1 remains held

GET repeatable read record n-------------------- Obtain a shared lock on record n

PUT ADD record n+1-------------------------------- Obtain an exclusive lock on record n+1

GET UPD record 2 ---------------------------------- Obtain an exclusive lock on record 2

PUT UPD record 2 ----------------------------------- Lock on record 2 remains held

CLOSE --- All Locks are retained

End of JOB Step (normal)------------------------- Commit changes release all locks

Application Requirements - Explicit Backout Example

//ddname DD DSN=Recoverabledatasetname,DISP=SHR

//step1   EXEC PGM=vsamrlspgm

Begin JOB Step -------------------------------------- No locks held

OPEN ACB MACRF=(RLS,OUT)

(UR1)

GET UPD record 1----------------------------------- Obtain an exclusive lock on record 1

PUT UPD record 1 --------------------------------- Lock on record 1 remains held

GET repeatable read record n------------------- Obtain a shared lock on record n

PUT ADD record n+1------------------------------- Obtain an exclusive lock on record n+1

GET UPD record 2 --------------------------------- Obtain an exclusive lock on record 2

PUT UPD record 2 ---------------------------------- Lock on record 2 remains held

Call SRRBACK -------------------------------------- Undo changes, all locks released .

CLOSE

End of JOB Step

Application Requirements - Implicit Backout Example

//ddname DD DSN=Recoverabledatasetname,DISP=SHR

//step1   EXEC PGM=vsamrlspgm

Begin JOB Step -------------------------------------- No locks held

OPEN ACB MACRF=(RLS,OUT)

(UR1)

GET UPD record 1----------------------------------- Obtain an exclusive lock on record 1

PUT UPD record 1 --------------------------------- Lock on record 1 remains held

GET repeatable read record n------------------- Obtain a shared lock on record n

PUT ADD record n+1------------------------------ Obtain an exclusive lock on record n+1

GET UPD record 2 --------------------------------- Obtain an exclusive lock on record 2

PUT UPD record 2 ---------------------------------- Lock on record 2 remains held

-------------------------------------- Cancel ---

End of JOB Step (abnormal) ---------------------- Undo changes release all locks

Transactional VSAM auto Commit Design

- New parameter in the job step JCL

  //stepname EXEC positional-parm, TVSAMCOM=({minval},{maxval})

- New system level parameter in the IGDSMSxx member of sys1.parmlib
  - ✓ TVSAMCOM=({minval},{maxval}

- The JCL overrides the value in IGDSMSxx

Minval: Minimum number of update requests   Maxval: Maximum number of update requests

Transactional VSAM will adjust the commit frequency to a number between *minvalue* and *maxvalue* based on record lock contention analysis for the current unit of recovery

QUESTIONS???

Thank You