DFSMS Basics: VSAM

Transactional VSAM (TVS) Basics and Implementation

Enhancing your RLS applications through transactional processing of VSAM data sets

Speaker: Neal Bohling, bohling@us.ibm.com
Session : 9094

Agenda

- RLS & TVS Overview – what’s the problem?
- Transactional VSAM Overview – what’s the solution?
- Setup and Use – how do I use it?
- Performance Considerations
- Commands – tracking what’s going on
- References – for more information

Quick Background - RLS

Problem:

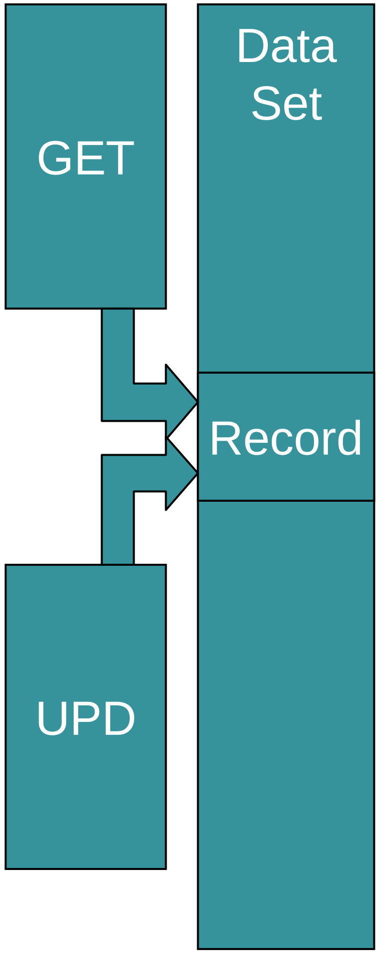
- One data set, many users.
- Serialization can get messy and data can get lost.

Previous solution:

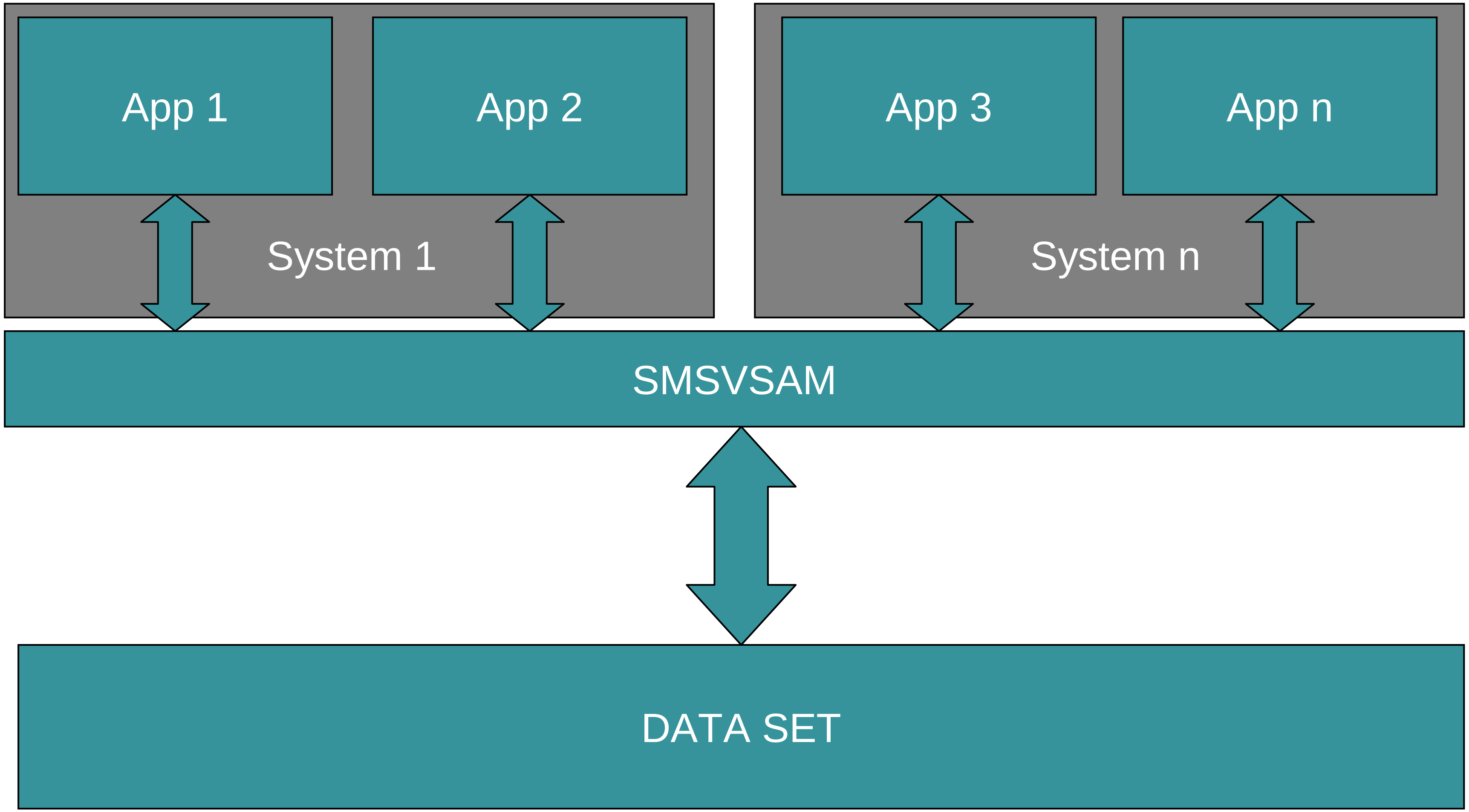
- CICS FOR (Function Shipping)

RLS Solution:

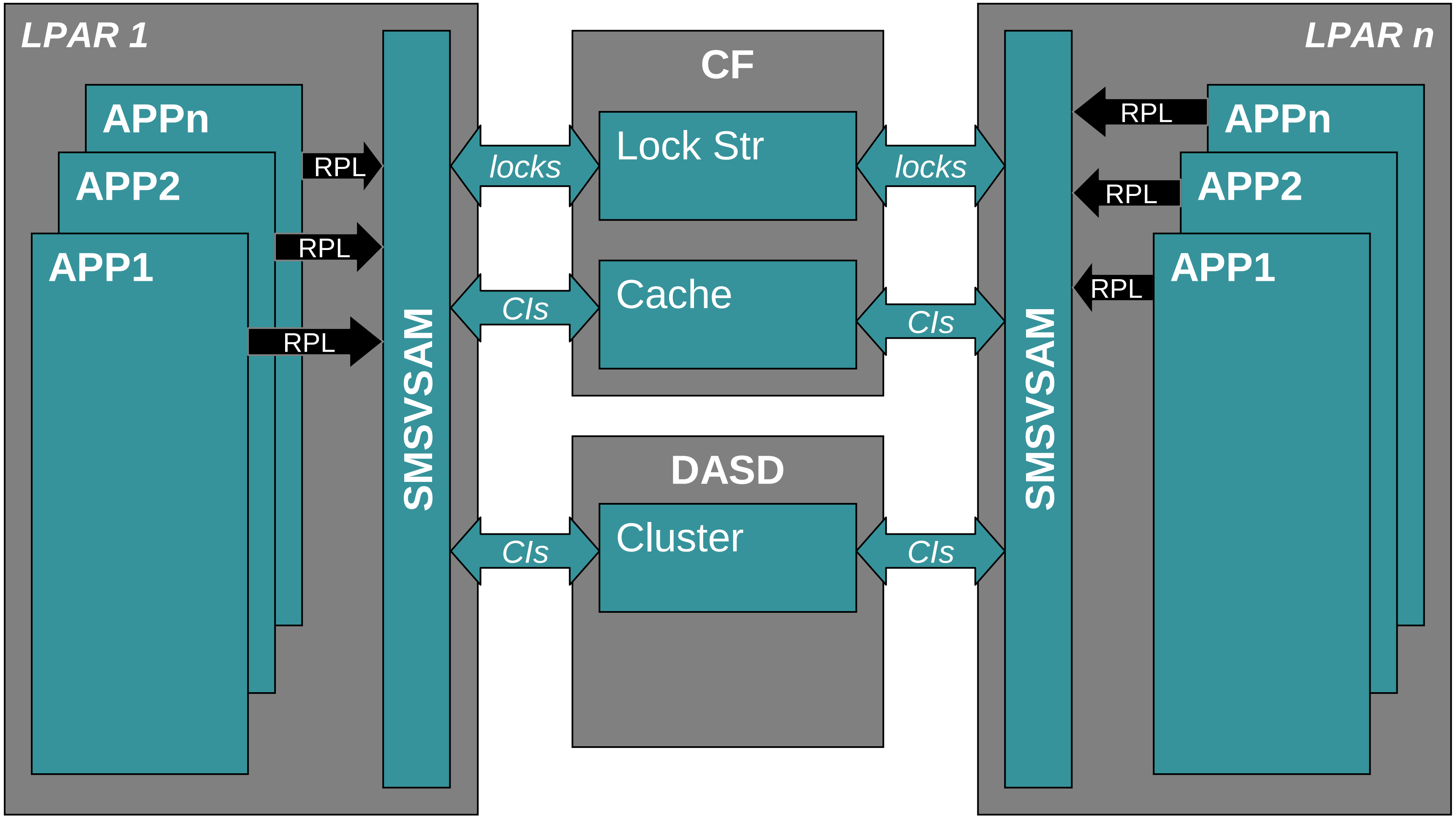
- VSAM Record Level Sharing
  - All access goes through SMSVSAM
  - Plex-wide serialization through locks in the CF



RLS Access



Typical RLS Setup



Quick Background – RLS & CICS

New Problem:

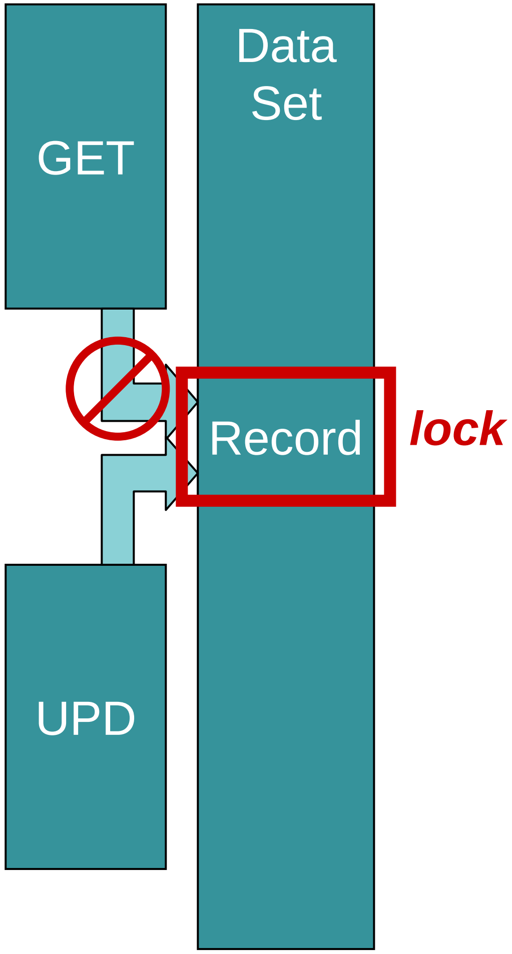
- Any recoverable data set open is READ ONLY to non-recoverable access (RLS and non-RLS)
- Ex. CICS through RLS and “batch” using RLS.

Common Solutions:

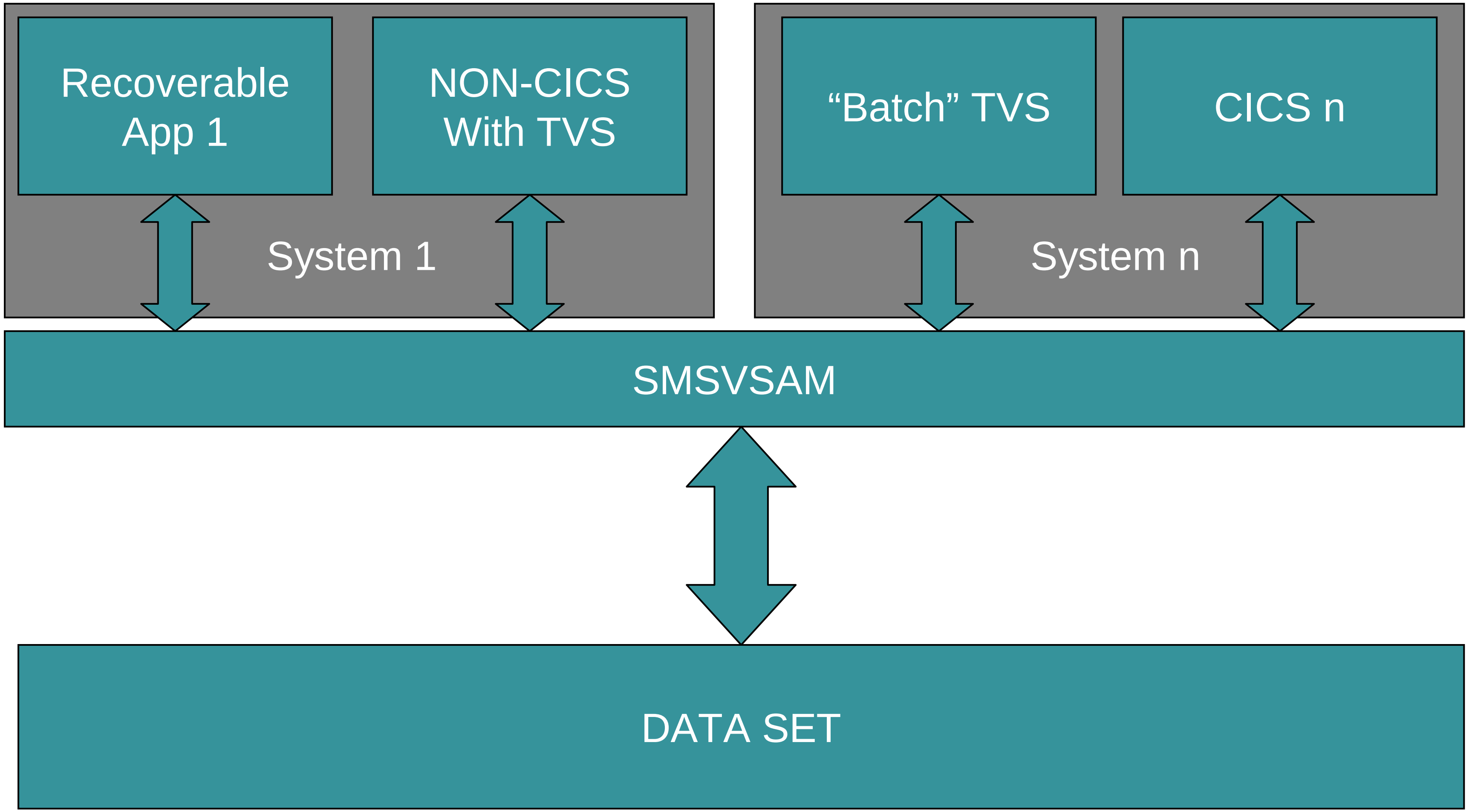
- Quiesce current activity
- Move CICS activity to a different file
- “Batch Window”

TVS Solution:

- Batch jobs using TVS become Recoverable Registered Regions
- Jobs using TVS can run simultaneously with CICS
- TVS Manages Recovery



RLS Access


Recoverable App 1
NON-CICS With TVS
System 1
"Batch" TVS
CICS n
System n
SMSVSAM
DATA SET

TRANSACTIONAL VSAM

Design Objective:

Enhance VSAM Record Level Sharing (RLS) to provide data recovery capabilities for any application exploiting VSAM RLS.

Recovery Capabilities include:

- Transactional Recovery
- Data set recovery

VSAM RLS becomes a "transactionalized" access method, hence "Transactional VSAM" (TVS).

TVS Overview

Transactional VSAM allows any job that uses RLS (such as batch jobs) to be recoverable

Implications:

- Cross-system record-level serialization through RLS
- *Recoverable subsystems (such as CICS) need not come down to allow other RLS activity (such as batch) (24x7 avail)*
- Fully able to interact with other recoverable regions

Data Set Recovery

- What is recovery?

- BACKWARD:
  - Allows the last update or set of updates to be undone
  - ‘UNDO’
  - Uses atomic updates / transactions
  - Uses logs to store changes

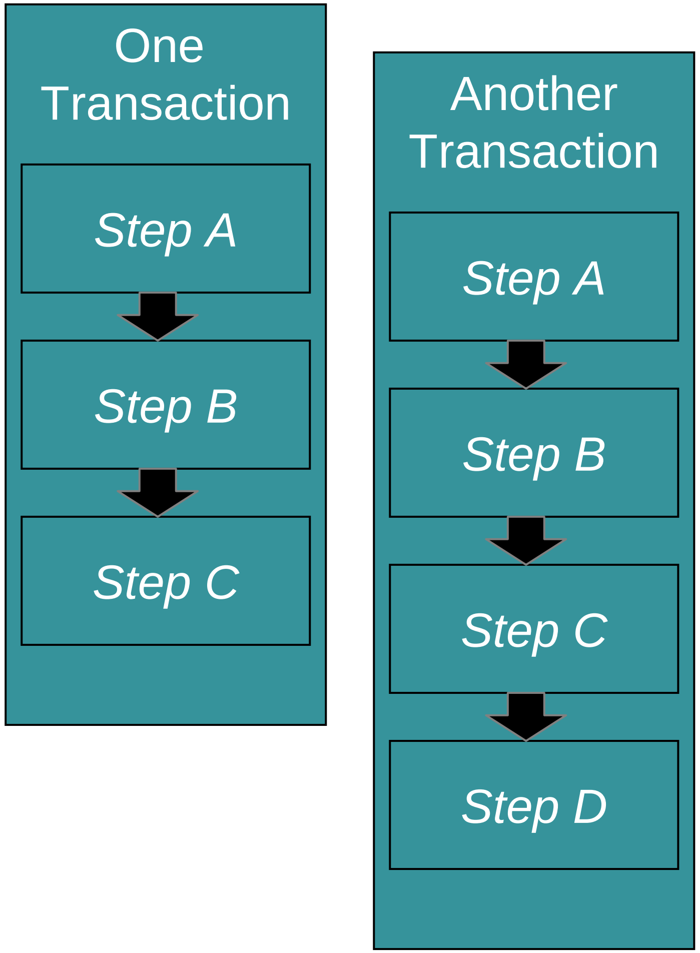
- FORWARD
  - Allows utilities to rebuild a file from backup
  - Uses logs to store forward-changes

Transactions and Transactional Recovery

- ***A Transaction or Unit of Recovery is a set of updates or changes that act as one unit of processing***

- ***Atomic update***
  - All of nothing
- ***Commit***
  - Finalizes a set of updates
- ***Backout***
  - Removes a set of updates
  - Based on logged updates

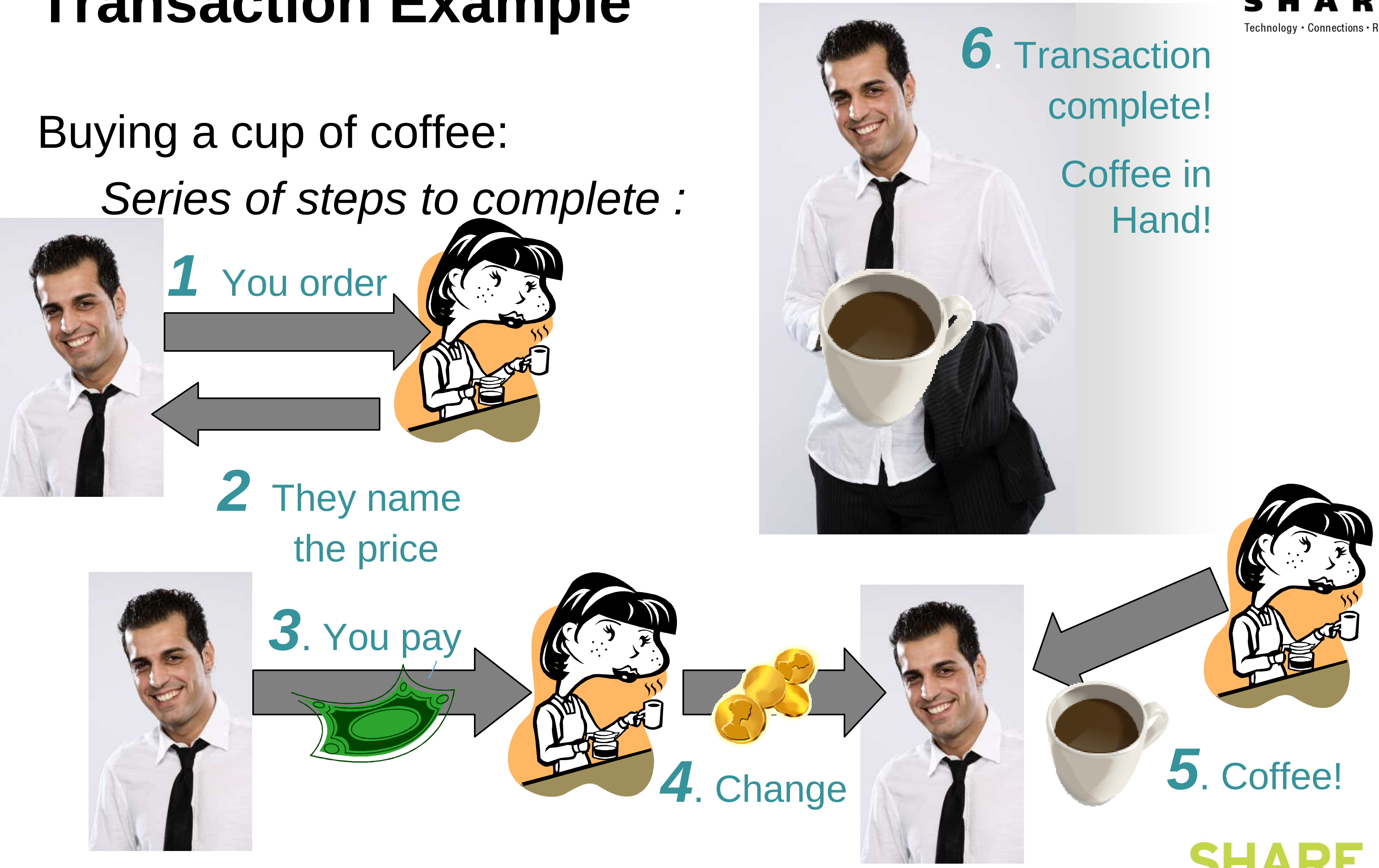
- *Referred to in TVS as a* ***UR***



Transaction Example

Buying a cup of coffee:

Series of steps to complete :



1 You order

2 They name the price

3. You pay

4. Change

5. Coffee!

6. Transaction complete!

Coffee in Hand!

Recoverable Data Sets (when using RLS)

***Recoverable data sets** are data sets that support backout (and potentially forward recovery) when opened by a recoverable region (such as CICS or TVS)*

RECOVERABLE

- Can do transaction recovery
- LOG(UNDO) – backward
- Changes are logged
- Changes can be backed out
- Read ONLY for non-RLS access
- LOG(ALL) – forward recovery

NON-RECOVERABLE

- Cannot recover
- LOG(NONE) or undefined
- Changes are not logged
- Changes cannot be undone
- R/W from all regions

Recoverable Regions

Recoverable Subsystems are applications capable of:

- Transactional Recovery (backward recovery)
- Data set Recovery (forward recovery)
- Data set changes are logged
- An example of an IBM recoverable region is CICS

A Recoverable Subsystem Manager is capable of:

- Managing transactional recovery between one or more recoverable subsystems
- An example of an IBM Recoverable Subsystem is the z/OS Recoverable Resource Manager (RRS)
- Recoverable Subsystems Register with Manager
- Uses ‘Units of Recovery’ (UR, transaction)

Recovery (Backward)

If there is a failure:

- Locks will be held to maintain integrity (RETAINED locks)
- Read the log file to retrieve unmodified data
- Restore data to unmodified state
- Release the serialization

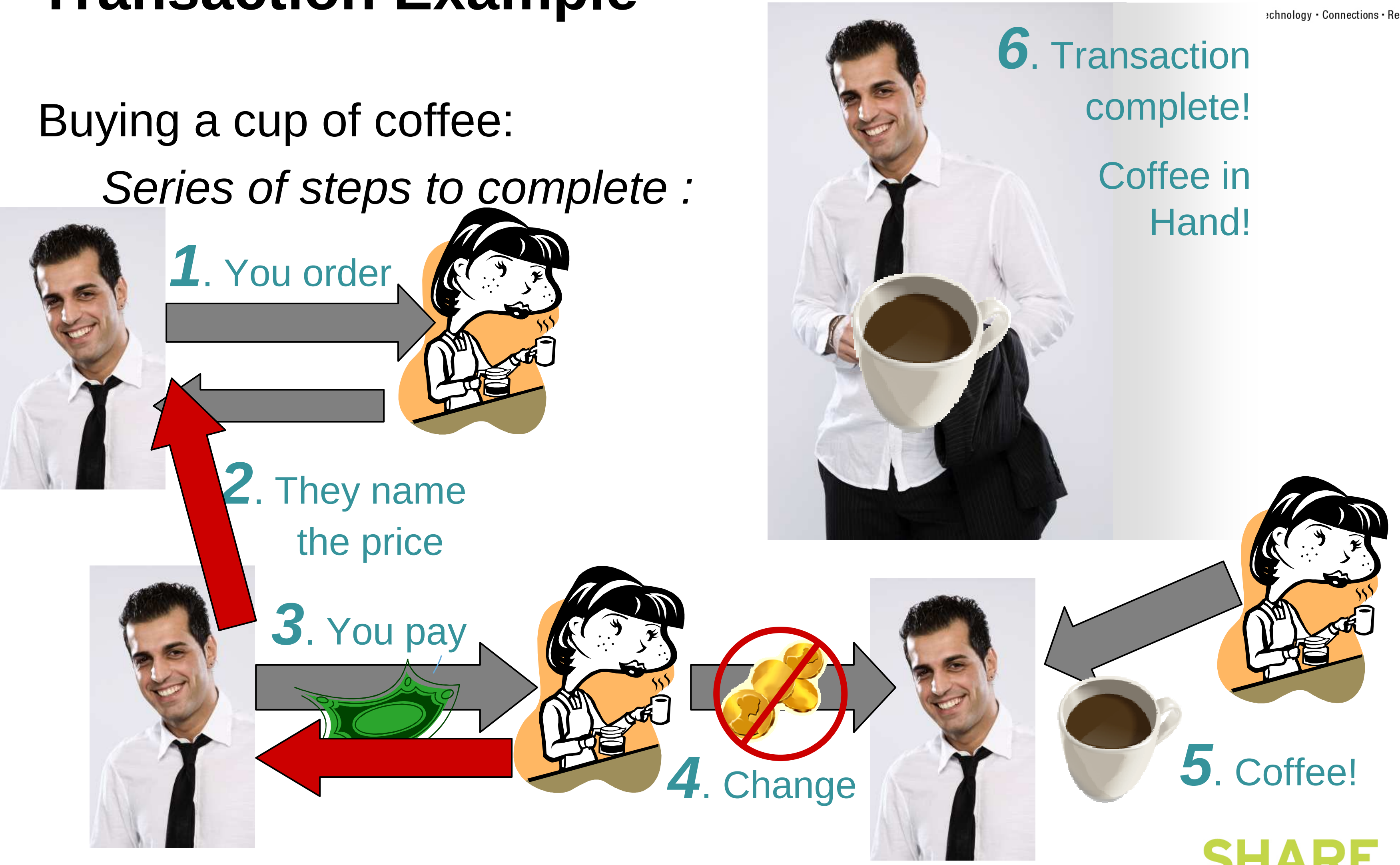
If a BACKOUT fails:

- Log the backout failure in another log, the SHUNTLOG
- Maintain serialization on the modified data (RETAINED locks)

Transaction Example

Buying a cup of coffee:

Series of steps to complete :



1. You order

2. They name the price

3. You pay

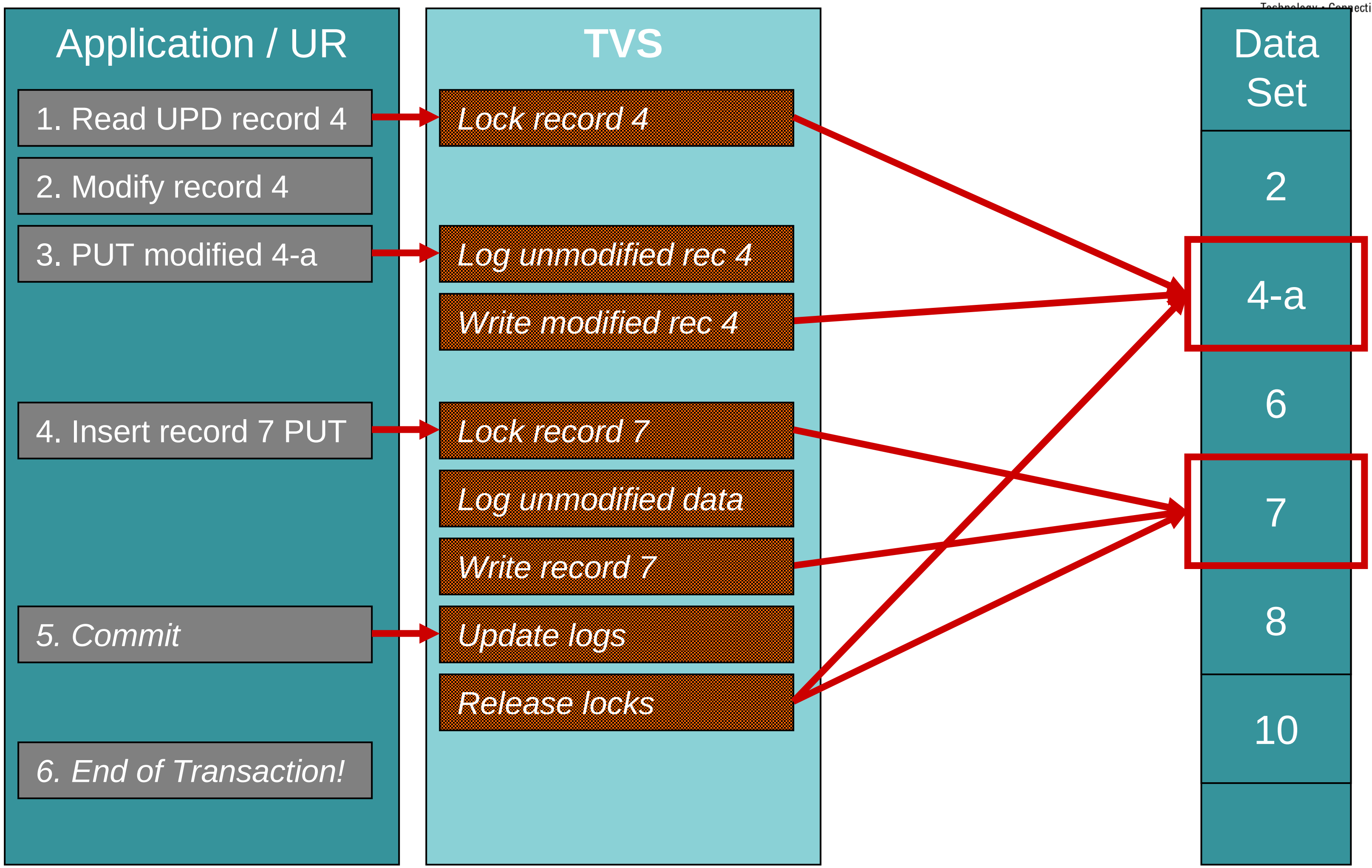
4. Change

5. Coffee!

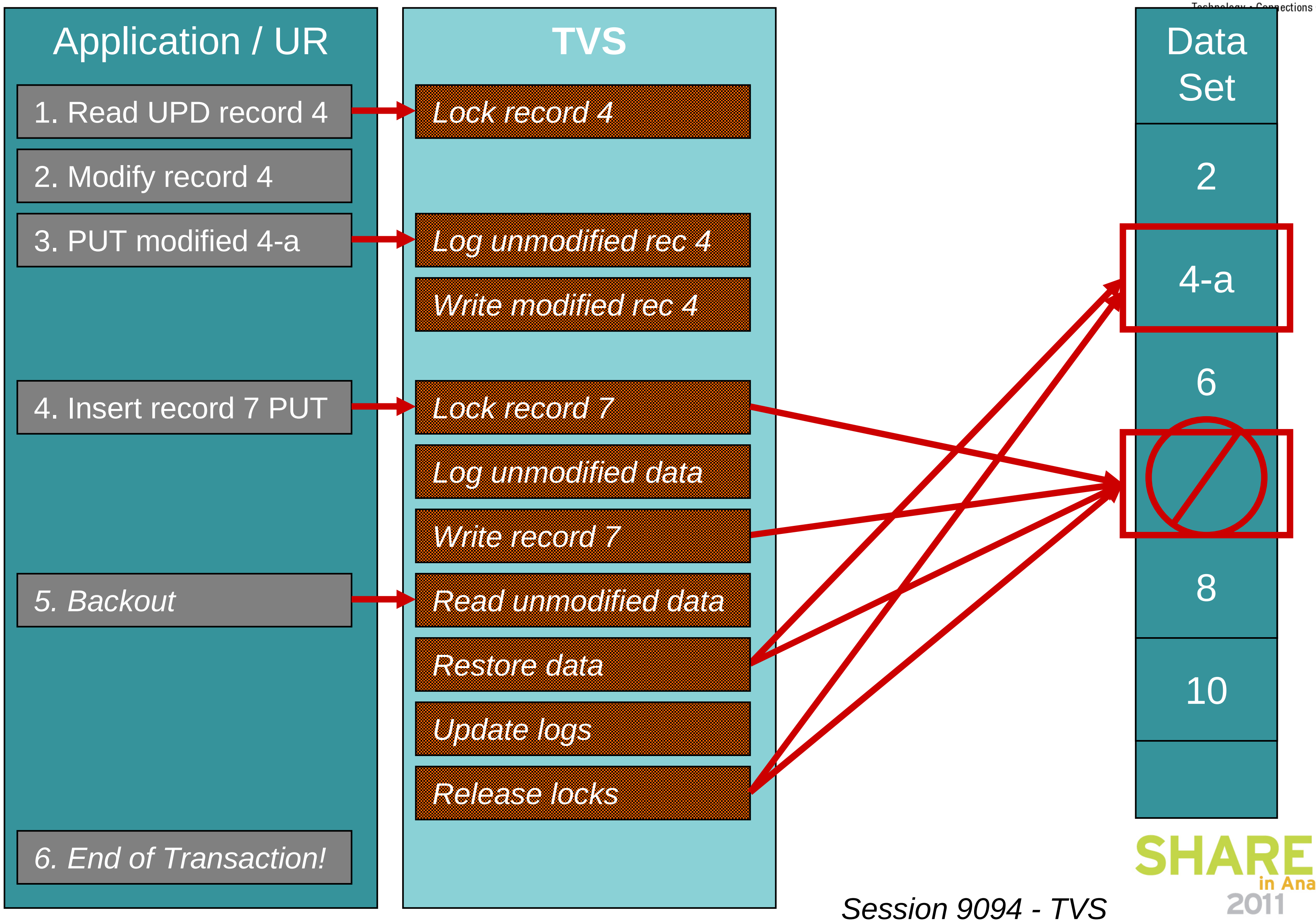
6. Transaction complete!

Coffee in Hand!

A Technical Example – successful



A Technical Example – Failure!



Application / UR

1. Read UPD record 4
2. Modify record 4
3. PUT modified 4-a
4. Insert record 7 PUT
5. *Backout*
6. *End of Transaction!*

TVS

Lock record 4

Log unmodified rec 4

Write modified rec 4

Lock record 7

Log unmodified data

Write record 7

Read unmodified data

Restore data

Update logs

Release locks

More on logging

- Data Set updates are written to the LOG
- TVS, RRS, CICS all take advantage of it in different ways
- TVS uses System LOGGER (IXLOGR)
- Uses LOGSTREAMS
  - Defined in the LOGR Policy in the coupling facility
- Logstreams can be shared between CICS / TVS, especially for forward recovery

TVS Logs

- **Undo Log** (required) – Primary System Log
  - One per image
  - Holds the changes made by URids on that system
  - Used for backout

- **SHUNT Log** (required) – Secondary System Log
  - One per image
  - Holds URs that TVS cannot complete (I/O error, etc)
  - Holds Long-running URs (moved from Undo log)

- Forward recovery logs (optional)
  - Plex-wide logs
  - Shared between CICS and TVS
  - Assigned to data sets during data set allocation (LOGSTREAMID)

- Log of Logs (optional)
  - Holds tie-up records and file-close records
  - Used by recovery applications such as CICSVR

TVS Component Interaction

Three basic functions necessary for transactional recovery:

- **Resource locking (VSAM RLS)**
  - Serialized access to changed resources
  - At the record level
  - Uses the coupling facility

- **Resource Recovery Logging (LOGGER)**
  - Keep track of backward changes (UNDO)
  - Keep track of forward changes (REDO / FR)

- **Two-phase commit and backout protocols (RRS)**
  - Ensures ATOMIC operation (transactions)
  - COMMIT
  - BACKOUT

The Overall Flow

- As TVS comes up:
  - Registers with SMSVSAM as a recoverable subsystem
  - Dynamically connect to the BACKOUT and SHUNT logs

- When a request is issued (GET/PUT/etc):
  - Register transaction with RRS and get a Unit of Recovery ID
  - Hold record-level serialization for the duration of URid
  - Log the unmodified data via IXLOGR to the backout log, and optionally the change in the forward recovery log

- When a COMMIT is issued:
  - Commit can be issued explicitly (via RRSCMIT)
  - Commits are implicitly issued during EOT
  - Release the locks
  - Log the successful COMMIT

SETUP

Hardware / Software changes to enable TVS

System Requirements

- Hardware:
  - Coupling Facility
  - At least one z/OS LPAR (monoplex or parallel sysplex)
- Software:
  - z/OS 1.4 or higher (current lowest release is z/OS 1.10)
  - z/OS VSAM RLS (SMSVSAM) Implemented
  - z/OS Transactional VSAM (separately priced feature)
  - z/OS RRMS Implemented (RRS)
  - z/OS System Logger Implemented
  - CICS VSAM Recovery (CICSVR) Utility (optional)

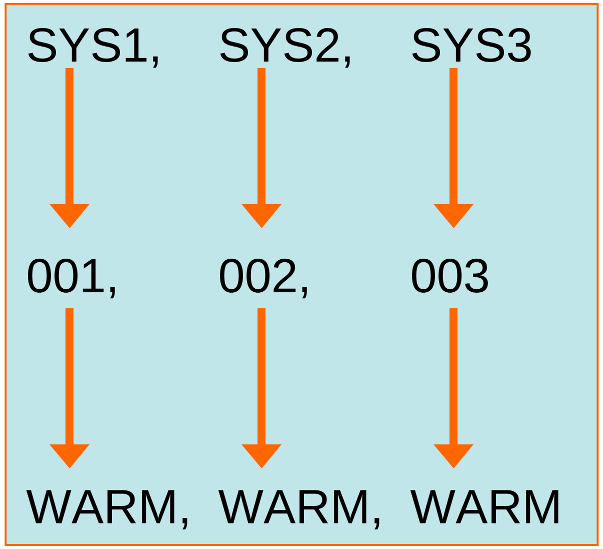
Required Parmlib Configuration

- **IGDSMSxx Parmlib Member**
  (Note, this does not include RLS/SMSVSAM parameters)

- **SYSNAME**(sysname1,sysname2,…) *
  - Systems on which TVS is to run
  - Same order is TVSNAME

- **TVSNAME**(nn1,nn2,..) *
  - TVS Instance names
  - Suffix to IGWTV

- **TV_START_TYPE**(COLD|WARM,COLD|WARM,…)
  - Type of startup
  - Same order as TVSNAME
  - COLD – deletes any information in UNDO & SHUNT logs and starts
  - WARM – reads the UNDO & SHUNT log and performs any actions needed



SYS1, SYS2, SYS3

001, 002, 003

WARM, WARM, WARM

Parmlib Configuration (Optional)

- **LOG_OF_LOGS**(logstreamid)
  - Specifies LOG of LOGS logstream
  - Used for forward recovery

- **MAXLOCKS**(nnn,iii)
  - Specifies when to issue warning messages about the number of held locks

- **AKP**(nnn,nnn,…) - Activity Keypoint trigger
  - Helps TVS maintain the UNDO and SHUNT logs
  - Removes entries that are no longer needed (URid no longer in use)
  - Defaults to 1000

- **QTIMEOUT**(nnn|300)
  - Number of seconds to wait before QUIESCE EXITS assume that the QUIESCE will not complete

TVS Startup Messages:

```
IGW865I TRANSACTIONAL VSAM INITIALIZATION HAS STARTED.
IGW414I SMSVSAM SERVER ADDRESS SPACE IS NOW ACTIVE. 327

IGW860I TRANSACTIONAL VSAM HAS SUCCESSFULLY REGISTERED WITH RLS

IGW848I 02182011 11.45.28 SYSTEM UNDO LOG IGWTV001.IGWLOG.SYSLOG 553
        INITIALIZATION HAS STARTED
IGW848I 02182011 11.45.29 SYSTEM UNDO LOG IGWTV001.IGWLOG.SYSLOG 577
        INITIALIZATION HAS ENDED
IGW848I 02182011 11.45.29 SYSTEM SHUNT LOG IGWTV001.IGWSHUNT.SHUNTLOG
        INITIALIZATION HAS STARTED
IGW848I 02182011 11.45.29 SYSTEM SHUNT LOG IGWTV001.IGWSHUNT.SHUNTLOG
        INITIALIZATION HAS ENDED

IGW865I TRANSACTIONAL VSAM INITIALIZATION IS COMPLETE.
IGW886I 0 RESTART TASKS WILL BE PROCESSED DURING TRANSACTIONAL VSAM
        RESTART PROCESSING
IGW866I TRANSACTIONAL VSAM RESTART PROCESSING IS COMPLETE.
```

Logger Configuration

- Update the CFRM Policy to contain list structures for the LOGS

```
//POLICY  EXEC PGM=IXCMIAPU
//SYSIN DD *
  DEFINE STRUCTURE
       NAME(LOG_IGWLOG_001)
       LOGSNUM(10)
       MAXBUFSIZE(64000)
       AVGBUFSIZE(2048)
```

- Update the LOGR Policy to contain the SMSVSAM logs

```
//POLICY  EXEC PGM=IXCMIAPU
//SYSIN DD *
  DEFINE LOGSTREAM
  NAME(IGWTV001.IGWLOG.SYSLOG)
  STRUCTURENAME(LOG_IGWLOG_001)
       LS_SIZE(1180)
       STG_DUPLEX(YES)
       DUPLEXMODE(COND)
       HIGHOFFLOAD(85)
       LOWOFFLOAD(15)
       DIAG(YES)
```

Data Set Allocation

- Add the following to IDCAMS define:
  - LOG( )
    - NONE – non-recoverable data set. Any RLS application can read/write
    - UNDO – Recoverable data set requiring backout logging. Can be opened for read/write by any RLS Recoverable Subsystems (CICS or TVS)
    - ALL – Recoverable data set requiring backout and forward recovery logging. Can be opened for read/write by any RLS Recoverable Subsystem
  - LOGSTREAMID(logs_id)
    - Logstream ID for any data set defined with LOG(ALL)

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

Application Changes

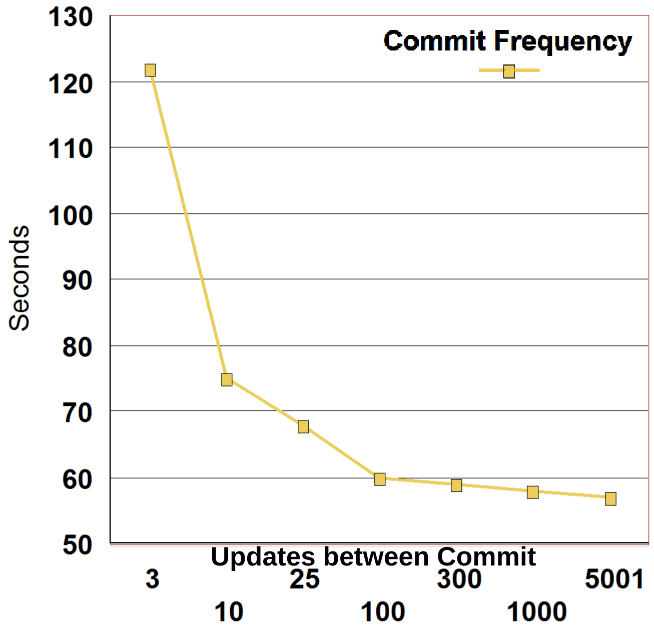
- Data sets will be accessed via TVS when:
  - Any RLS access for recoverable data set
    - Via ACB:
      - *`ACB MACRF=(RLS,OUT)` for recoverable data set*
      - *`ACB MACRF=(RLS,IN), RLSREAD=CRE`*
    - Via DD:
      - *`//ddname DD DSN=recoverable.dsn,DISP=SHR, RLS=(CR|NRI)` and `ACB MACRF=(OUT)`*
      - *`//ddname DD DSN=recoverable.dsn,DISP=SHR, RLS=(CRE)` and `ACB MACRF=(IN)`*

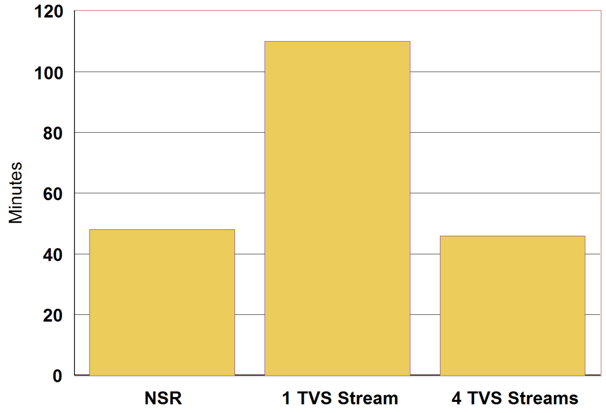
Application Changes (cont)

- Recommendations:
  - Can be EXPLICIT – add command to your job
  - Can be IMPLICIT – will run during End-of-Job
  - RLS Applications using TVS should be modified to include:
    - SSRCMIT – commit
    - SSRBACK – backout
  - SSRCMIT and SSRBACK will either COMMIT or BACKOUT the UR provided by SMSVSAM on behalf of the application
  - Periodic explicit COMMIT/BACKOUT will release the locks in a timely fashion. Failure to do so may hold up other jobs.
- High-Level Language Support:
  - PLI, C & C++, COBOL, Assembler

Performance Considerations

- TVS does add overhead
  - Increased code path length
  - Cross-Address Space access to server
  - Loss of NSR chained sequential I/O
  - Loss of LSR deferred write
  - New overhead of record locking
  - New overhead of CF cache access
  - Logging (for already RLS work)
- Commit Frequency
  - Too many can add unnecessary overhead
  - Too few can cause delays due to lock contention
- "Parallelizing" the workload
  - Spreading out the work reduces individual overhead and increases overall efficiency
  - Several TVS streams can work simultaneously





Application Example (Commit)

Explicit Commit:

```
//ddname  DD  DSN=Recoverabledatasetname,DISP=SHR,RLS=CRE
//step1   EXEC  PGM=vsamrlspgm
Begin JOB Step   ------------------------------------- No locks held
OPEN  ACB MACRF=(NSR,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
```

Implicit Commit:

```
//ddname  DD  DSN=Recoverabledatasetname,DISP=SHR,RLS=CRE
//step1   EXEC  PGM=vsamrlspgm
Begin JOB Step --------------------------------------- No locks held
OPEN  ACB MACRF=(NSR,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 Example (Backout)

Explicit Backout

```
//ddname  DD  DSN=Recoverabledatasetname,DISP=SHR,RLS=CRE
//step1   EXEC  PGM=vsamrlspgm
Begin JOB Step --------------------------------------- No locks held
OPEN  ACB MACRF=(NSR,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
```

Implicit Backout

```
//ddname  DD  DSN=Recoverabledatasetname,DISP=SHR,RLS=CRE
//step1   EXEC  PGM=vsamrlspgm
Begin JOB Step --------------------------------------- No locks held
OPEN  ACB MACRF=(NSR,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
```

Restart Considerations

- Restarting applications that use TVS must be done from the last COMMIT point

- Restarting from the beginning could result in data integrity problems

- A checkpoint / restart type system should be implemented to determine restart point of the application

Commands

- D SMS,TRANVSAM

```
D SMS,TRANVSAM
RESPONSE=SYSTEM1
 IEE932I 006
 IGW800I 22.48.15 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
```

Commands

- D SMS,LOG(logid|ALL)
  - *Shows information about the logs currently in use by TVS*
- D SMS,SHUNTED,SPHERE|URID()
  - *Shows shunted work across the plex*
- D SMS,URID(urid)
  - *Displays information about the unit of recovery*
- D SMS,JOB(jobname)
  - *Displays information about the job, and for TVS, gives UR information*

Commands

- SHCDS commands provide a myriad of capabilities:
  - List information kept by SMSVSAM/TVS about subsystems and data sets:
    - *LISTDS, LISTSUBSYS, LISTSUBSYSDS, LISTRECOVERY, LISTALL, LISTSHUNTED*
  - Control Forward Recovery
    - *FRSETRR, FRUNBIND, FRBIND, FRRESETRR, FRDELETEUNBOUNDLOCKS*
  - Allow NON-RLS update – use sparingly
    - *PERMITNONRLSUPDATE, DENYNONRLSUPDATE*
  - Reset various information about subsystems or RLS
  - Handling SHUNTED work:
    - *RETRY, PURGE*

SHCDS Commands Example

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

===>   SHCDS LISTDS('recoverabledataset*')
----- LISTING FROM SHCDS ----- IDCSH02 --------------------------------------------------

 DATA SET NAME----recoverabledataset
   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


                            SHARING SUBSYSTEM STATUS
   SUBSYSTEM    SUBSYSTEM         RETAINED        LOST         NON-RLS UPDATE
   NAME         STATUS            LOCKS           LOCKS        PERMITTED
   ---------   --------------   ---------------- ----------   ----------------
   IGWTV001    ONLINE--FAILED    YES                    NO                NO
 ***
```

Summary

- Transactional VSAM allows:
  - Concurrent access with recoverable regions (such as CICS)
  - Full data set recovery through logging and atomic updates
- Eliminates the Batch Window
- Requires minimal changes to existing jobs
- Provides plex-wide consistency
- Overall, provides a more effective way to integrate recoverable and non-recoverable workloads
  (ex. CICS and NON-CICS such as batch)

References:

- *DFSMStvs Planning and Operating Guide*, SC26-7348
- *DFSMStvs Overview and Planning Guide*, SG24-6971
- *VSAM Demystified*, SG24-6105
- *MVS Initialization and Tuning Reference*, SA22-7592
- *MVS System Commands*, SA22-7627

Copyright / Legal

NOTICES AND DISCLAIMERS

Copyright © 2010 by International Business Machines Corporation.

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

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

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

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. 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 NONINFRINGEMENT. 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.

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.

Trademarks

DFSMSdfp, DFSMSdss, DFSMShsm, DFSMSrmm, IBM, IMS, MVS, MVS/DFP, MVS/ESA, MVS/SP, MVS/XA, OS/390, SANergy, and SP are trademarks of International Business Machines Corporation in the United States, other countries, or both.

AIX, CICS, DB2, DFSMS/MVS, Parallel Sysplex, OS/390, S/390, Seascape, and z/OS are registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Domino, Lotus, Lotus Notes, Notes, and SmartSuite are trademarks or registered trademarks of Lotus Development Corporation. Tivoli, TME, Tivoli Enterprise are trademarks of Tivoli Systems Inc. in the United States and/or other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

Other company, product, and service names may be trademarks or service marks of others.

Backup Slides / Additional Reference

Example of TVS startup:

```
IGW865I TRANSACTIONAL VSAM INITIALIZATION HAS STARTED.
IGW414I SMSVSAM SERVER ADDRESS SPACE IS NOW ACTIVE. 327
IGW467I DFSMS TVSNAME PARMLIB VALUE SET DURING 510
        SMSVSAM ADDRESS SPACE INITIALIZATION ON SYSTEM: SYSTEM1
        TVSNAME: IGWTV001
        CURRENT VALUE: ENA-ED  1
IGW467I DFSMS TRANSACTIONAL VSAM UNDO LOG PARMLIB VALUE SET DURING 513
        SMSVSAM ADDRESS SPACE INITIALIZATION ON SYSTEM: SYSTEM1
        UNDO LOGSTREAM NAME:  IGWTV001.IGWLOG.SYSLOG
        CURRENT VALUE: ENA-ED  1
IGW467I DFSMS TRANSACTIONAL VSAM SHUNT LOG PARMLIB VALUE SET DURING 514
        SMSVSAM ADDRESS SPACE INITIALIZATION ON SYSTEM: SYSTEM1
        SHUNT LOGSTREAM NAME:  IGWTV001.IGWSHUNT.SHUNTLOG
        CURRENT VALUE: ENA-ED  1
IGW467I DFSMS TRANSACTIONAL VSAM ACTIVITY KEY POINT PARMLIB VALUE 516
        SET DURING SMSVSAM ADDRESS SPACE INITIALIZATION ON SYSTEM: SYSTEM1
        CURRENT VALUE: 200
IGW467I DFSMS TRANSACTIONAL VSAM TVS_START_TYPE 517
        PARMLIB VALUE SET DURING
        SMSVSAM ADDRESS SPACE INITIALIZATION ON SYSTEM: SYSTEM1
        TVSNAME VALUE:  IGWTV001
        CURRENT VALUE: WARM  1
IGW467I DFSMS TRANSACTIONAL VSAM LOG_OF_LOGS PARMLIB VALUE SET DURING 524
        SMSVSAM ADDRESS SPACE INITIALIZATION ON SYSTEM: SYSTEM1
        LOG_OF_LOGS LOGSTREAM NAME: IGWTVS1.LOG.OF.LOGS
        CURRENT VALUE: ENA-ED  1
```

Example of TVS startup:

```
IGW860I TRANSACTIONAL VSAM HAS SUCCESSFULLY REGISTERED WITH RLS
IGW876I TRANSACTIONAL VSAM INITIALIZATION WAITING FOR RRS
ATR201I RRS COLD START IS IN PROGRESS.
ASA2011I RRS INITIALIZATION COMPLETE. COMPONENT ID=SCRRS
IGW877I TRANSACTIONAL VSAM INITIALIZATION RESUMING AFTER WAIT FOR RRS
IGW848I 02182011 11.45.28 SYSTEM UNDO LOG IGWTV001.IGWLOG.SYSLOG 553
        INITIALIZATION HAS STARTED
IXC582I STRUCTURE TVS_LOG001 ALLOCATED BY SIZE/RATIOS. 566
          PHYSICAL STRUCTURE VERSION: C75A333B 5A6E2E32
          STRUCTURE TYPE:             LIST
          CFNAME:                     FACIL02
          ALLOCATION SIZE:         12 M
          POLICY SIZE:          12000 K
          POLICY INITSIZE:          0 K
          POLICY MINSIZE:           0 K
          IXLCONN STRSIZE:          0 K
          ENTRY COUNT:            873
          ELEMENT COUNT:         7567
          ENTRY:ELEMENT RATIO:           1 :      9
        ALLOCATION SIZE IS WITHIN CFRM POLICY DEFINITIONS
IXL014I IXLCONN REQUEST FOR STRUCTURE TVS_LOG001 567
        WAS SUCCESSFUL.  JOBNAME: IXGLOGR ASID: 0017
        CONNECTOR NAME: IXGLOGR_SYSTEM1 CFNAME: FACIL02
```

Example of TVS startup:

```
IXL015I STRUCTURE ALLOCATION INFORMATION FOR 568
        STRUCTURE TVS_LOG001, CONNECTOR NAME IXGLOGR_SYSTEM1
         CFNAME     ALLOCATION STATUS/FAILURE REASON
         --------   ----------------------------------------
         FACIL02    STRUCTURE ALLOCATED CC001800
         FACIL01    PREFERRED CF ALREADY SELECTED CC001800
IXG283I STAGING DATASET IXGLOGR.IGWTV001.IGWLOG.SYSLOG.SYSTEM1
        ALLOCATED NEW FOR LOGSTREAM IGWTV001.IGWLOG.SYSLOG
        CISIZE=4K, SIZE=442368
IGW474I DFSMS VSAM RLS IS CONNECTING TO 576
        TRANSACTIONAL VSAM  LOGSTREAM IGWTV001.IGWLOG.SYSLOG
        SYSTEM NAME:              SYSTEM1
        TRANSACTIONAL VSAM INSTANCE NAME:  IGWTV001
IGW848I 02182011 11.45.29 SYSTEM UNDO LOG IGWTV001.IGWLOG.SYSLOG 577
        INITIALIZATION HAS ENDED
IGW848I 02182011 11.45.29 SYSTEM SHUNT LOG IGWTV001.IGWSHUNT.SHUNTLOG
        INITIALIZATION HAS STARTED
IXG283I STAGING DATASET IXGLOGR.IGWTV001.IGWSHUNT.SHUNTLOG.SYSTEM1 585
        ALLOCATED NEW FOR LOGSTREAM IGWTV001.IGWSHUNT.SHUNTLOG
        CISIZE=4K, SIZE=442368
IGW474I DFSMS VSAM RLS IS CONNECTING TO 587
        TRANSACTIONAL VSAM  LOGSTREAM IGWTV001.IGWSHUNT.SHUNTLOG
        SYSTEM NAME:              SYSTEM1
        TRANSACTIONAL VSAM INSTANCE NAME:  IGWTV001
IGW848I 02182011 11.45.29 SYSTEM SHUNT LOG IGWTV001.IGWSHUNT.SHUNTLOG
        INITIALIZATION HAS ENDED
```

Example of TVS startup:

```
IGW848I 02182011 11.45.29 LOG OF LOGS IGWTVS1.LOG.OF.LOGS 589
        INITIALIZATION HAS STARTED
IXG283I STAGING DATASET IXGLOGR.IGWTVS1.LOG.OF.LOGS.SYSTEM1 595
        ALLOCATED NEW FOR LOGSTREAM IGWTVS1.LOG.OF.LOGS
        CISIZE=4K, SIZE=442368
IGW474I DFSMS VSAM RLS IS CONNECTING TO 597
        TRANSACTIONAL VSAM  LOGSTREAM IGWTVS1.LOG.OF.LOGS
        SYSTEM NAME:              SYSTEM1
        TRANSACTIONAL VSAM INSTANCE NAME:  IGWTV001
IGW848I 02182011 11.45.30 LOG OF LOGS IGWTVS1.LOG.OF.LOGS 598
        INITIALIZATION HAS ENDED

IGW865I TRANSACTIONAL VSAM INITIALIZATION IS COMPLETE.
IGW886I 0 RESTART TASKS WILL BE PROCESSED DURING TRANSACTIONAL VSAM
        RESTART PROCESSING
IGW866I TRANSACTIONAL VSAM RESTART PROCESSING IS COMPLETE.
IGW467I DFSMS TRANSACTIONAL VSAM QTIMEOUT PARMLIB VALUE SET DURING 602
        SMSVSAM ADDRESS SPACE INITIALIZATION ON SYSTEM: SYSTEM1
        CURRENT VALUE: 400  1
IGW467I DFSMS TRANSACTIONAL VSAM MAXLOCKS PARMLIB VALUE SET DURING 603
        SMSVSAM ADDRESS SPACE INITIALIZATION ON SYSTEM: SYSTEM1
        CURRENT VALUE: 100  50  1
```

Recovery (Forward)

- To Recover a data set with retained locks:
  - Stop any current transactions
  - DELETE recoverabledataset
  - Restore backup copy
  - Apply committed changes since last backup
  - Restart access (Retry SHUNTED work)
- CICSVR automates this process
  (does not retry shunted work)

Recovery (Forward)

- To Recover a data set with retained locks, take following steps
  - `SHCDS FRSETRR(recoverabledataset)` – sets the FR indicator
  - `SHCDS FRUNBIND(recoverabledataset)-` unbinds the retained locks, allowing delete
  - `DELETE recoverabledataset`
  - `<Restore backup copy>`
  - `<apply committed changes since last backup (must set ACBRECOV)>`
  - `SHCDS FRBIND(recoverabledataset) –` reattach retained locks
  - `SHCDS FRRSETRR –` re-enable access to dataset
  - `SHCDS LISTSHUNTED SPHERE(recoverabledataset)-` display information about shunted work
  - `SHCDS RETRY SPHERE(recoverabledataset)-` retry the syncpoint

- CICSVR automates this process (does not retry shunted work)