#SHAREorg

SHARE
Technology · Connections · Results

Buffering, Record Level Sharing, and Performance Basics for VSAM Data Sets

Session 12999

Presented by
Michael E. Friske

SHARE in San Francisco 2013

Expectations From This Session

- You will be given some general rules of thumb for making VSAM perform well.
- You will be given an overview of many of the things to look for when you suspect a VSAM data set is not performing well.
- This session will not make you a VSAM performance expert, but it will put you on the way to becoming one.

Things to Remember

- Every data set is different
- What makes one data set perform really well may cause another data set to perform very poorly
- You have to understand the data and how it will be used to make good performance recommendations
  - Is the data set being processed sequentially, skip sequentially, or randomly?
  - Will the insert activity be light or heavy?
  - Are records being inserted evenly throughout the data set, or are inserts concentrated in one area of the data set?
  - Will records increase in size when they are updated?

Eliminate I/O, Improve Performance

- The best way to improve performance is to eliminate I/O
  - Allocate more buffers
  - Decrease index level
  - Use striping for sequential processing that is I/O bound
  - Use Hiperbatch for jobs that all read the same VSAM data set (does not support Extended Format data sets)

Types of VSAM Buffering

- Non-Shared Resources (NSR)
- Local Shared Resources (LSR)
- Global Share Resources (GSR)
- Record Level Sharing (RLS)

Batch Defaults for VSAM

- NSR buffering
- Two DATA buffers and one INDEX buffer

VSAM Buffering

- NSR Buffering – Best for sequential and skip-sequential processing
- LSR Buffering – Best for direct processing or when multiple tasks within the same address space are using the same data set
- RLS Buffering – Best for data sets being accessed by multiple address spaces running multiple systems

VSAM Buffering in CICS

- The buffering used is specified in the File Control Table (FCT)
- When LSR buffering is used, a buffer pool is used that matches the CI size for the DATA and INDEX component
- If a buffer pool does not exist for that CI size, the next size available will be used

Let the System Determine the Optimum Type of Buffering & Number of Buffers

- Each job may process a data set differently
- When coding BUFNI and BUFND in JCL or using Batch LSR, JCL changes may be required if the application program changes the way it processes a data set or changes the record size
- Use System Managed Buffering or an equivalent product to determine the best type of buffering to use and the optimum number of buffers
- Competing products include
  - CA / Hyper-Buf from CA
  - Performance Essentials from Rocket Software / Mainstar
  - Veloci-Raptor from Dino Software
  - Mainview Batch Optimizer from BMC

System Managed Buffering

- Replaces Batch LSR (BLSR)
- Is only available for Extended Format data sets
- Can be implemented via a DATACLAS definition or using the AMP JCL parameter
- JCL keywords that support System Managed Buffering
  - AMP='ACCBIAS=USER|SYSTEM|DO|DW|SO|SW'
  - AMP='SMBDFR=Y|N'
  - AMP='SMBHWT=nn'
  - AMP='SMBVSP=nnK|M'

Invoking System Managed Buffering

- Use a DATACLAS that Record Access Bias = SYSTEM
- Specify AMP='ACCBIAS=SYSTEM'
- System Managed Buffering can only be used for Extended Format data sets

ACCBIAS for Sequential

- SO – Sequential Optimize: More data buffers will be allocated to support sequential access, and NSR buffering will be used for this technique. Approximately 500KB of virtual storage will be used for buffers.
- SW – Sequential Weighted: The number of buffers will be optimized for sequential processing, but additional index buffers will be allocated for some direct processing. NSR buffering will be used, and the buffers will require about 100KB of virtual storage.

ACCBIAS for Direct

- DW – Direct Weighted: More index buffers will be allocated, but some data buffers will be reserved for sequential processing. NSR buffering will also be used for this technique and will require 100KB of virtual storage.
- DO – Direct Optimize: The virtual storage requirement will depend on the size of the data set. By default, SMB will attempt to allocate enough buffers to hold 20% of the data component and the entire index component.

ACCBIAS for Create

- CO – Create Optimize: This buffering technique is only used for loading a VSAM data set that is defined with the SPEED option. The High-Used RBA must be zero (0) at open time. A maximum of approximately 2MB of virtual storage will be used for this technique.
- CR – Create Optimize for Recovery: This buffering technique is only used for loading a VSAM data set that is defined with the RECOVERY option. Again, the High-Used RBA must be zero (0) at open time. A maximum of approximately 1MB of virtual storage will be used for this technique.

Unable to Get Enough Virtual Storage When ACCBIAS=DO

- Reduce the numbers of buffers by 50% from the optimum amount for the data components and retry.
- Reduce the number of DATA buffers to the minimum and retry. The minimum size is 1 MB.
- Reduce the number of INDEX buffers to the minimum and retry. The minimum is the amount of virtual storage to contain the entire index set plus 20% of the sequence set records.
- If none of the retries above is successful, change to ACCBIAS=DW.

Adjusting Virtual Storage Usage

- Message IEC161I 001(8,36)-087 indicates there is not enough virtual storage to obtain all of the buffers SMB requests
- Virtual storage can be limited for some data sets by specifying AMP=SMBVSP=nnK|M

Optimization Technique Selected

	SEQ BIAS in STORCLAS	DIR BIAS in STORCLAS	SEQ & DIR BIAS in STORCLAS	No BIAS Specified in STORCLAS
MACRF=SEQ (this is the default)	**SO**	**SW**	**SO**	**SO**
MACRF=(SEQ,SKP)	**SO**	**SW**	**SW**	**SW**
MACRF=DIR	**DW**	**DO**	**DO**	**DO**
MACRF=(DIR,SEQ) or MACRF=(DIR,SKP) or MACRF=(DIR,SEQ,SKP)	**SW**	**DW**	**DW**	**DW**

SMB Load Mode Options

- Optimization when the HURBA=0 and the data set is opened in LOAD mode
  - Create Optimize (CO)
  - Create Optimize Recovery (CR)
- These options cannot be specified by the user

Additional Tuning Options When ACCBIAS=DO

- SMBDFR=Y|N
  - Default for SHR(1,3) & SHR(2,3) is SMBDFR=Y
  - Default for all other share options is SMBDFR=N
- SMBVSP=nK|nM
- SMBHWT=n
  - Default is SMBHWT=0
  - The “n” is a weighted value between 0 - 99

Controlling Whether Below the Line or Above the Line Storage Is Used

- AMP='RMODE31='
  - ALL
  - BUFF – Buffers only
  - CB – Control blocks only
  - NONE

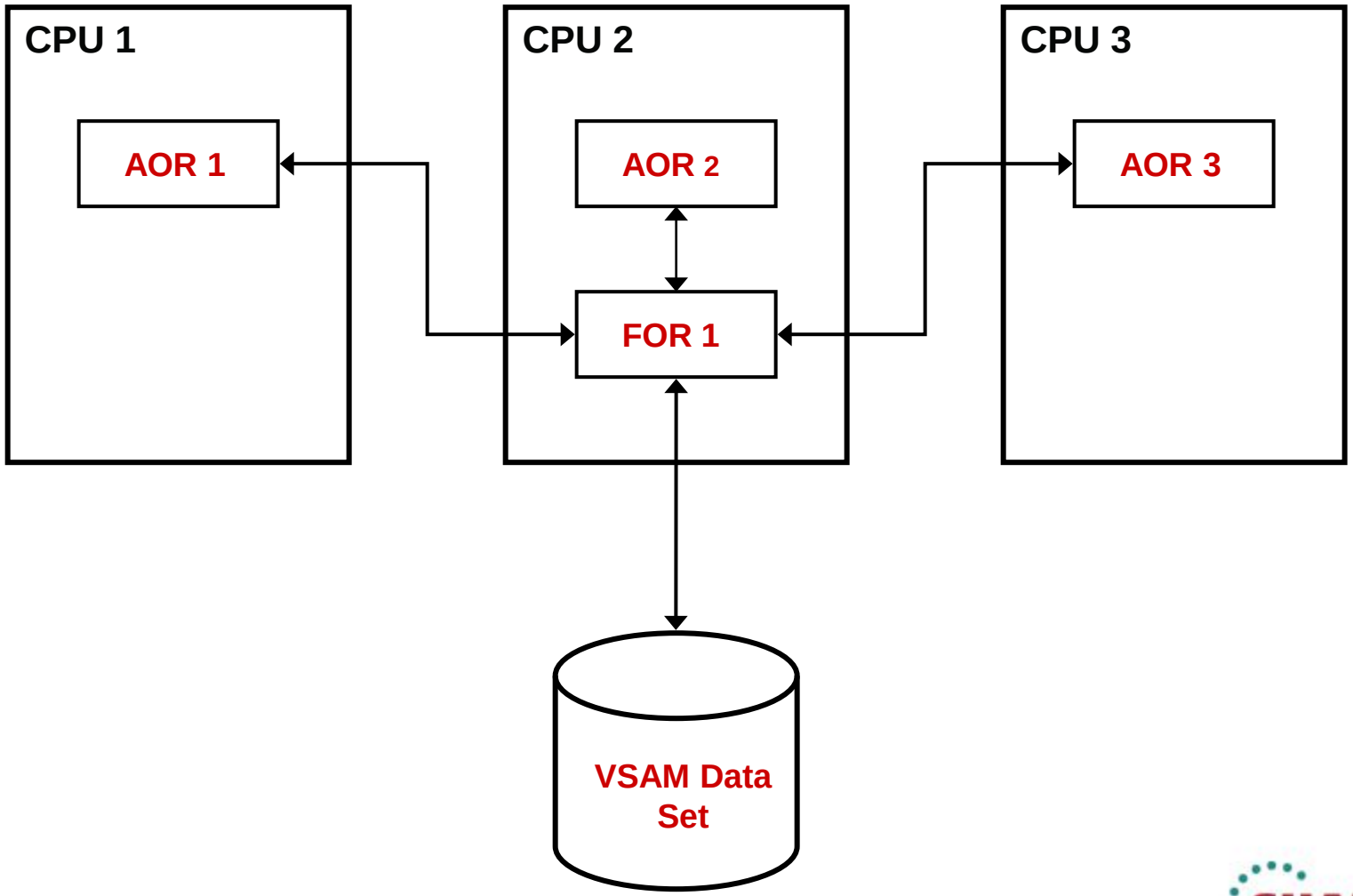
Specifying the Optimum Number of Buffers

- System Managed Buffering (SMB) selects the correct number of buffers
- Only adjust the selection if the application is not correctly indicating how it intends to use the data set
- Do not limit the REGION for the job
- Load the VSAM control blocks and the buffers above the line if the application program is storage constrained below the line

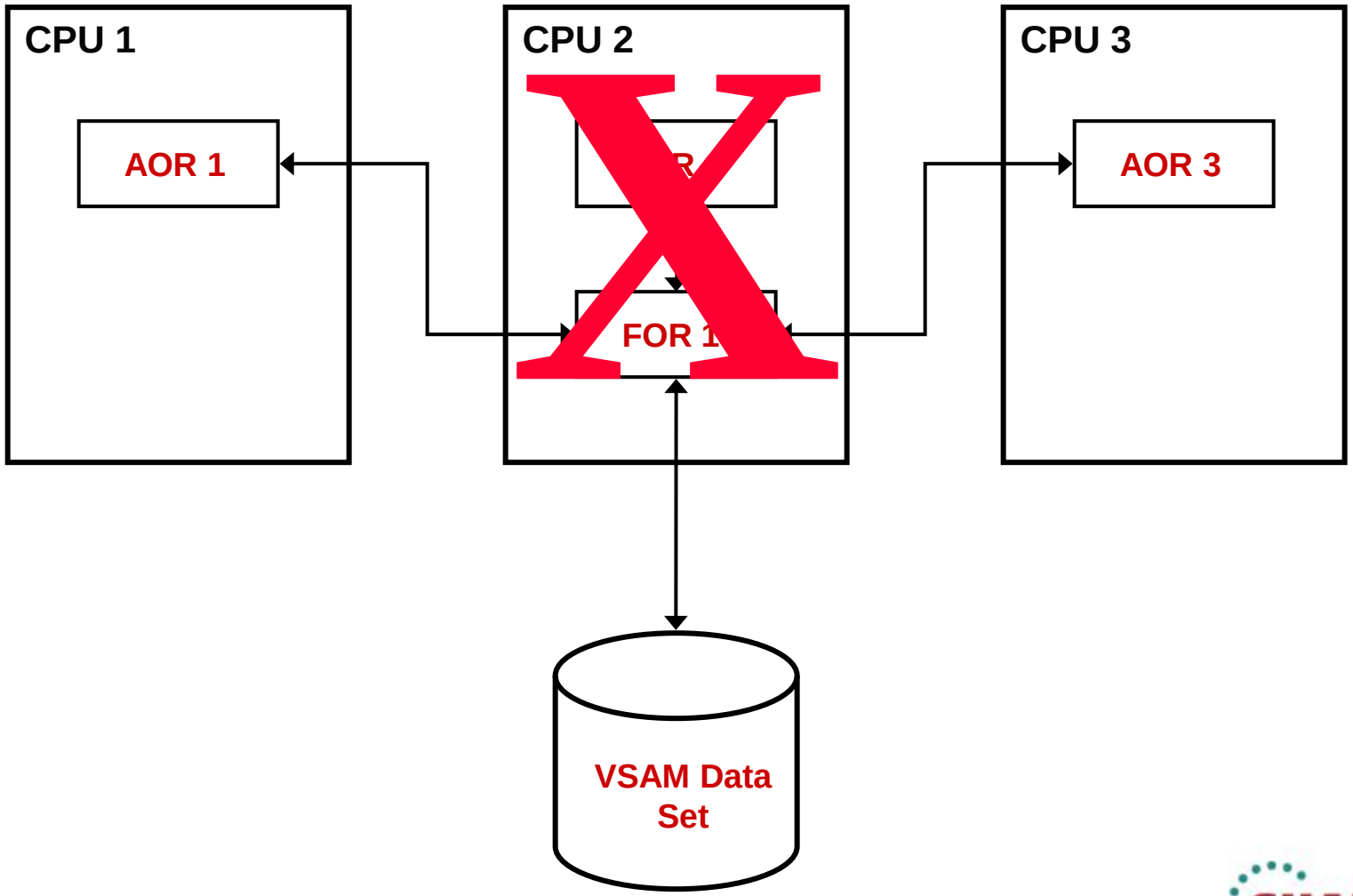
VSAM Record Level Sharing

VSAM Record Level Sharing is a function that allows VSAM data sets to be fully shared with data integrity among multiple user across multiple systems.

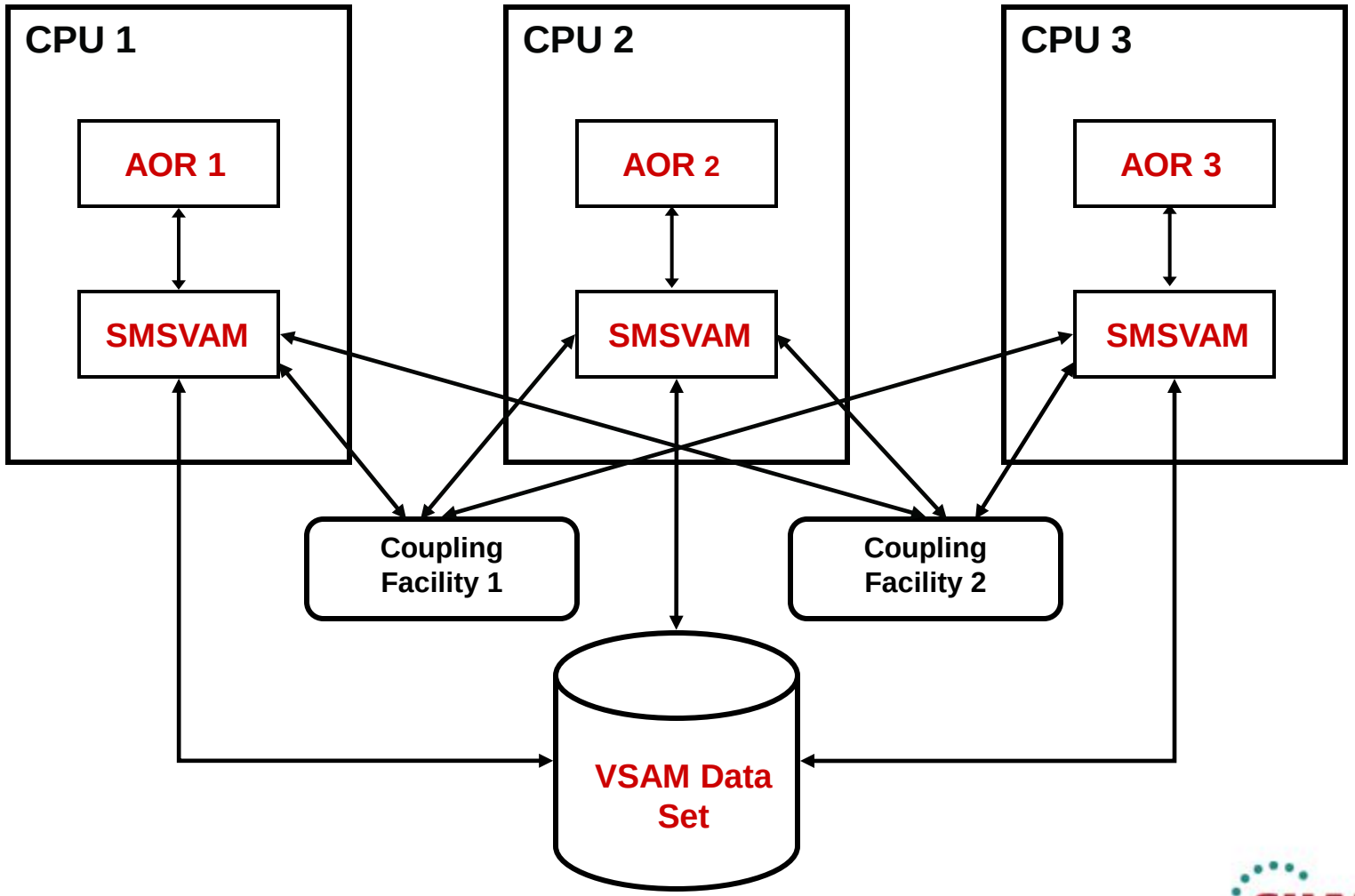
CICS VSAM Sharing Without RLS



Single Point of Failure When Using FORs



CICS Sharing With RLS



Advantages of Using VSAM RLS

- Increases data availability
  - No Single Point of Failure (SPOF)
  - Data remains available during both planned and unplanned outages
- Improves data integrity
  - No more “dirty reads” if another application is updating the data set
  - All users are aware of the HARBA and HURBA
  - Locks are held in the event of a CICS failure
- Eliminates the processor constraint for a single FOR
- Provides flexibility in balancing workloads
- Provides the base for Transactional VSAM

Programming Considerations for RLS

- In general, existing programs can use VSAM RLS without modifications
- Programs should be coded to handle a LOCKED condition
- Unsupported assemblers and compilers do not contain the necessary support for VSAM RLS
- Batch programs that access data sets in RLS mode must be compiled with an LE supported compiler
- All COBOL programs used to access VSAM data sets in RLS mode should use the second status area to obtain the VSAM FEEDBACK return code (not just File Status value)

SHAREOPTIONS for Data Sets Opened in RLS Mode

- The SHAREOPTIONS parameter is ignored for data sets opened in RLS mode.
- When a data set is opened in RLS mode, other jobs will be prevented from opening the data set in non-RLS mode unless the data set is defined with SHAREOPTIONS(2,x). In this case, the READ integrity will not be guaranteed for the non-RLS user.

Using RLS in Batch

- Batch does not perform sync points, so locks are not released after a record is updated until the job step ends.
- Batch jobs do not perform any time of logging for VSAM data sets, so back out and recovery are not possible
- IBM provides an RLS add on called Transactional VSAM (TVS) that creates sync points within the batch job step and logging
- Transactional VSAM provides the capability for batch jobs to update VSAM data sets while those same data sets are being updated by CICS

Consistent READ for Batch Jobs

- RLS=NRI – No Read Integrity
- RLS=CR – Consistent Read
- RLS=CRE – Consistent Read Explicit (Valid only for an application that supports commit and back out)

Buffering has a large impact on the performance of a VSAM data set, but there are other things that can influence VSAM performance.

OTHER PERFORMANCE CONSIDERATIONS

Control Interval Size

- For sequential processing, large Control Interval sizes provide better performance
- For direct processing, small Control Interval sizes provide better performance
- For mixed access (online and batch for example), a compromise may be required

FREESPACE

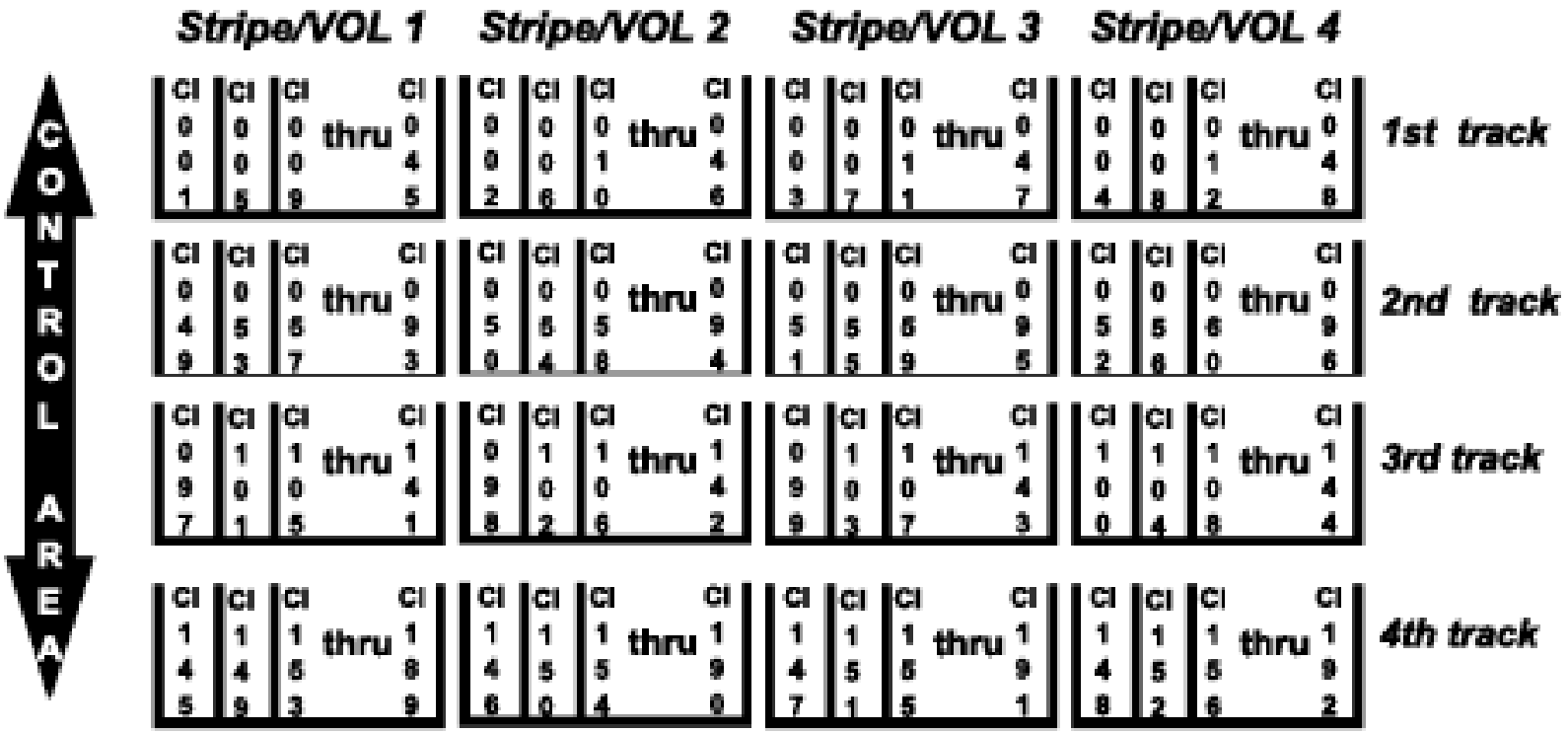
- Use FREESPACE to reduce the number of times VSAM has to perform a CI or a CA split
- Code CI free space percentage to be a multiple of the average record length
- Do not code FREESPACE across the whole data set if inserts will not be distributed across the data set
- Do not code too much FREESPACE to avoid wasting both disk and buffer space

VSAM Striping

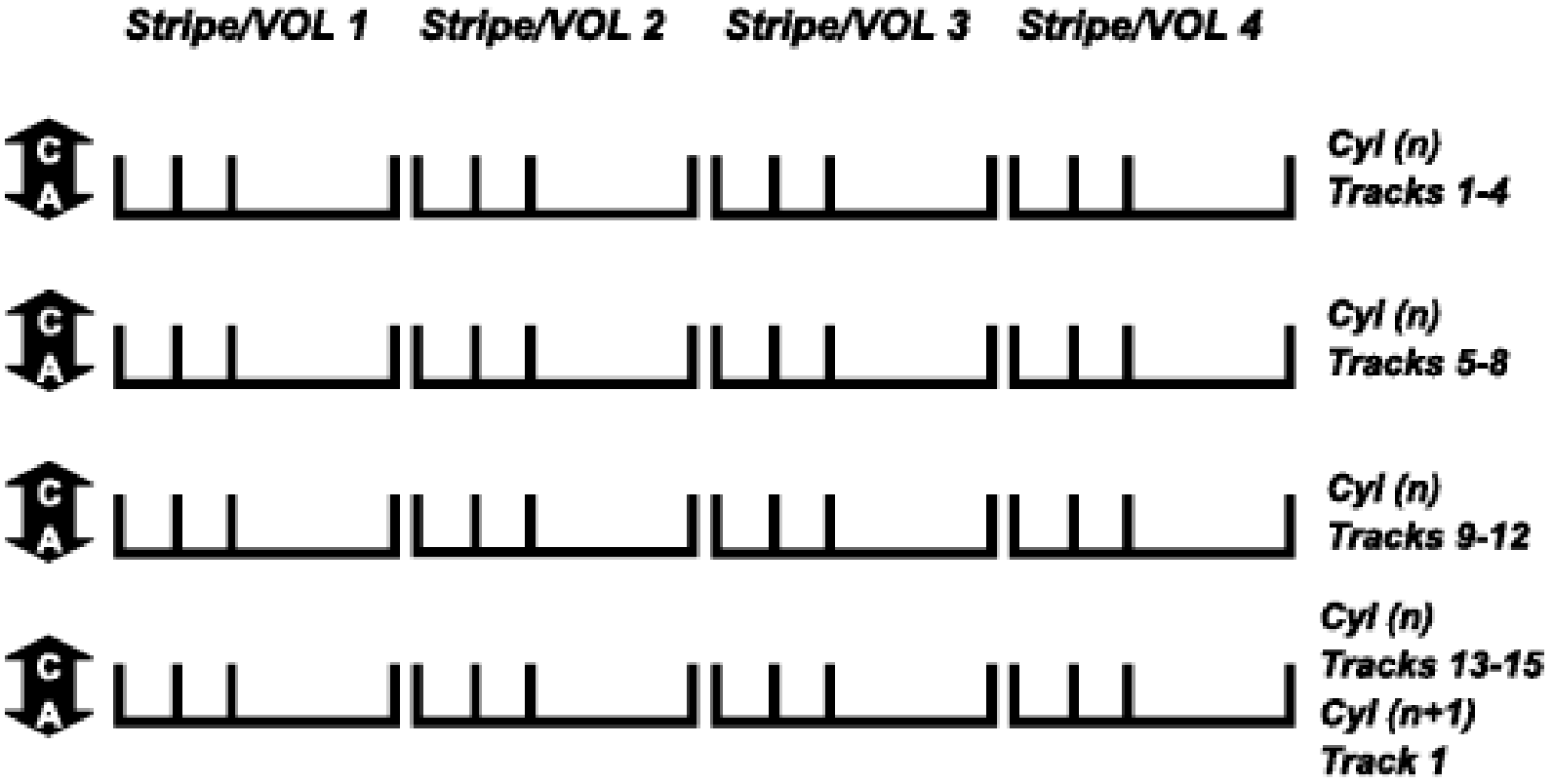
- Improves throughput for sequential, I/O bound applications
- Spreads control intervals in a control area across multiple devices
- Does not adversely impact the performance of random I/O applications

VSAM Striping Layout

Data CI size - 4k, Physical Blocksize - 4K
4K blocks per 3390 track - 12, Stripe count - 4
CYL (n n)



CI/CA Layout for Striped Data Sets



- Control Area (CA) size = 16 tracks spread across the four stripes
  - Cylinder allocation specified
  - Increased Index CL size requirement

Layering for Striped Data Sets

Layer 1:

VOLA - single primary allocation

VOLB - single primary allocation

VOLC - single primary allocation

Layer 2:

VOLA - single secondary allocation

VOLD - single secondary allocation

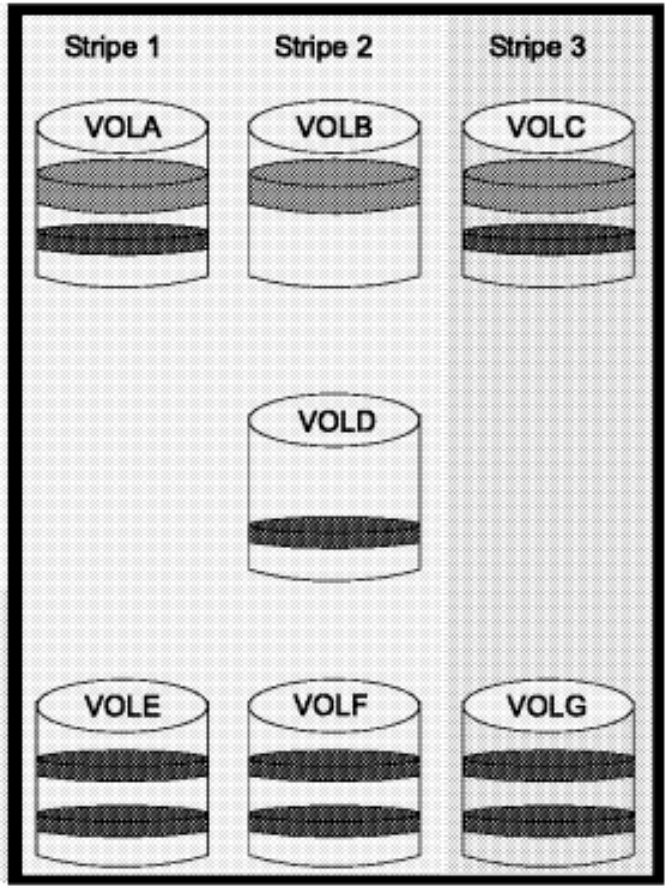
VOLC - single secondary allocation

Layer 3:

VOLE - two secondary allocations

VOLF - two secondary allocations

VOLG - two secondary allocations



DA6D4999

The Affect of SMS Compression on VSAM Performance

- SMS compression is only available for KSDSs
- SMS compression can reduce the number of EXCPs and the device connect time when reading and writing to a VSAM data set, but generally at a higher CPU cost
- SMS compression is optimized for READ I/O's, so it is better suited for data sets that have a higher READ to WRITE ratio
- The benefits are not as great for data sets with small records, with very large keys in relation to the record size, or with keys that do not start at offset zero

Determining How a Data Set Is Used

- SMF Type 42 Subtype 6 records contain data set level I/O statistics
  - Cut on the SMF interval and at data set CLOSE
- SMF Type 64 records contain VSAM close statistics
  - Cut when a VSAM data set is successfully closed
  - When an abend occurs, an SMF type 64 record will not be cut
- SMF Type 42 Subtype 15 – 19 records RLS statistics
- Talk to the application programmer

Other Sources of Information About How a Data Set Is Used

- Some products provide additional information
  - Statistics showing the number of sequential reads, sequential writes, direct reads, and direct writes
  - Analysis of a data set showing how the records are stored, how well the keys compress, how the CI's and CA's are utilized, where the free space is, and much more

Summary

- Buffering makes the biggest difference in how a VSAM data sets performs
- System Managed Buffering (or a competitive product) should be used to automate buffer optimization
- RLS can be beneficial for data sets accessed by multiple regions across multiple systems
- Some factors about how a data set is defined can influence the performance of a VSAM data set