VSE/VSAM Tips for Tuning

Dan Janda
The Swami of VSAM

The Swami of VSE/VSAM

Dan Janda
VSE, VSAM and CICS
Performance Consultant
RR 2 Box 49E
Hamlin Road
Montrose, PA 18801-9624

(570) 934-2862
theswami@epix.net
http://business.epix.net/~theswami

World Alliance of VM, VSE and Linux
Chattanooga, Tennessee
April, 2006

Abstract and References

- **Abstract:**

In this presentation, we give an overview of the VSE/VSAM Subsystem, its features and how to exploit them for best overall system performance. We intend to highlight newer features and those older features that are often overlooked and which our experience shows give the most "bang for the buck." We try to highlight key "gotchas" in the performance area.

- **Trademarks:**
  - IBM, VSE, VSE/ESA, ESA, CICS and DL/I are trademarks or registered trademarks of the IBM Corporation
  - The Swami of VSAM is a trademark of Dan Janda

VSE/VSAM Overview

- Virtual Storage Access Method
  - For Direct Access Storage Devices (disk devices)
    - General Purpose
    - File Organizations
      - Sequential -- records stored in the sequence presented
      - Direct -- records stored in arbitrary sequence
      - Indexed -- records stored via keys and an index
    - File Access Techniques
      - Sequential -- processing in the order stored
      - Direct -- processing in an arbitrary order
      - Keyed -- processing in key order

VSE/VSAM Overview

- Virtual Storage Access Method Functional Areas
  - Catalog
    - Volume, File information
    - Usage statistics
  - DASD Space Management
    - Space allocation, including secondary allocations
    - VSAM and non-VSAM files (VSAM managed SAM, etc.)
    - System Files
    - Libraries
  - Performance
    - Large data transfer sizes for sequential processing
    - Buffer look-aside for direct processing
  - Integrity
    - Backup/Restore
    - Intra- and Inter-System Sharing controls

VSE/VSAM Overview

- Virtual Storage Access Method Data Organizations
  - Key Sequenced Data Set (KSDS)
    - Indexed, stored logically in key sequence
  - Entry Sequenced Data Set (ESDS)
    - Non-indexed, stored in order inserted, new records at end of file
  - Relative Record Data Set (RRDS)
    - Non-indexed, stored in relative record order, fixed length records
  - Variable Length Relative Record Data Set (VRDS)
    - Indexed, stored in relative record order, but variable length records
  - Alternate Index Data Set (AIX)
    - A form of KSDS used as a "finder file"
    - Unique or non-unique keys are supported

VSE/VSAM Overview

- Virtual Storage Access Method Access Techniques
  - Sequential Access
    - Sequential, forward or backward
  - Keyed Access
    - Sequential, forward or backward
    - Skip Sequential, forward or backward
    - Direct, by full or partial (generic) key
  - Addressed Access
    - Sequential, forward or backward
    - Skip Sequential, forward or backward
    - Direct, by record address
  - Alternate Index Access
    - Same as for keyed access, above
    - Add direct access by non-unique key

VSE/VSAM Overview

- In this next section, we'll discuss how VSAM
  - Logically stores records on disk
    - Capabilities enabled by these techniques
    - Performance aspects
  - Physically stores records on disk
    - Disk space usage calculations
    - Performance aspects

Chattanooga
Tennessee
2006

VSE/VSAM Jargon

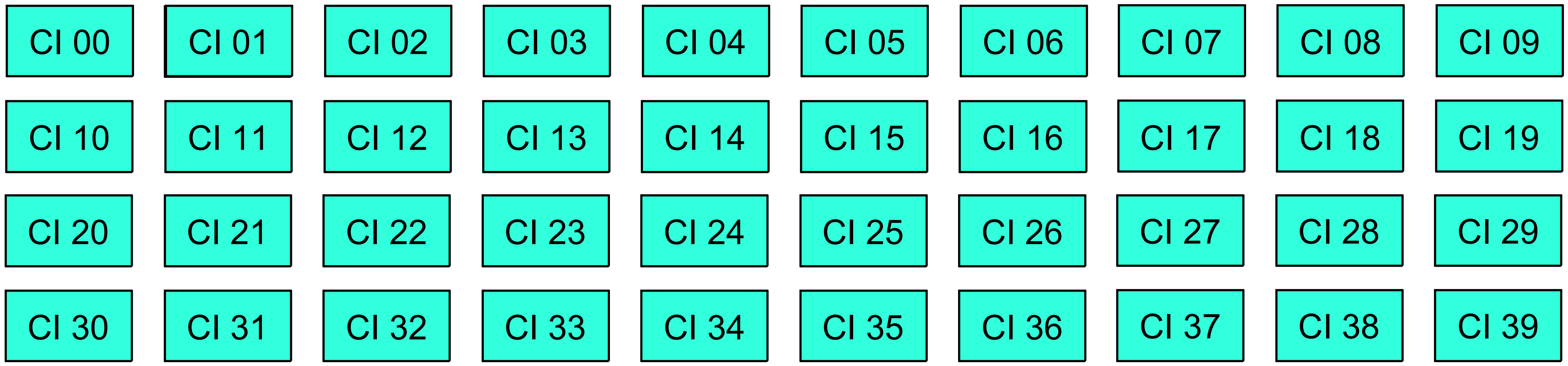
- **Control Interval (CI)**
  - **The smallest unit of data transfer between main and disk storage**
  - **One or more logical records are loaded into a CI**

Rec 01	Rec 02	Rec 03	Rec ...	Freespace	RDF(s)	CIDF

Rec 01--Rec nn	1 to n logical records of any length
Freespace	Unused space within CI available for additional record insertion or increase in length of existing records
RDF(s)	3-byte VSAM Record Descriptor Field for ESDS/KSDS, one per record length, one for all consecutive records of the same length for RRDS, one per numbered record slot
CIDF	4-byte Control Interval Descriptor Field

VSE/VSAM Jargon

- **Control Area (CA)**
  - **A group of CIs. In an indexed file, all the data CIs in a CA are mapped by a single index CI.**


CI 00 CI 01 CI 02 CI 03 CI 04 CI 05 CI 06 CI 07 CI 08 CI 09
CI 10 CI 11 CI 12 CI 13 CI 14 CI 15 CI 16 CI 17 CI 18 CI 19
CI 20 CI 21 CI 22 CI 23 CI 24 CI 25 CI 26 CI 27 CI 28 CI 29
CI 30 CI 31 CI 32 CI 33 CI 34 CI 35 CI 36 CI 37 CI 38 CI 39

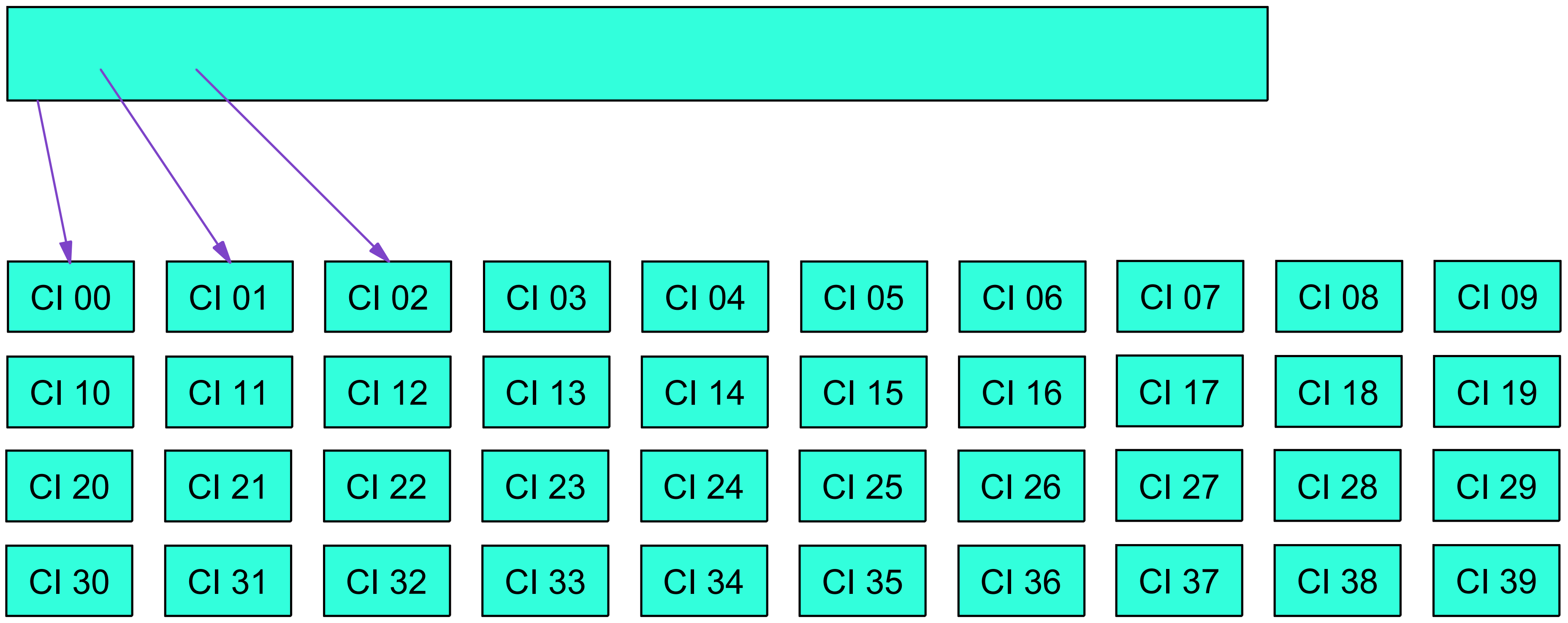
The size of a CA is the smallest among:
-- one cylinder (or "max-CA") on the device
-- the size of the primary allocation amount
-- the size of the secondary allocation amount

The number of CIs per CA depends on the device characteristics --
(track size, number of tracks per cylinder) and the CI and CA sizes

It is generally beneficial to have as large a CA size as possible

VSE/VSAM Jargon

- **Index Control Interval (Index CI)**
  - **A CI in an index, containing pointer entries to**
    - **The next level in the index, or**
    - **The data CI within the CA (if this is a Sequence Set CI)**



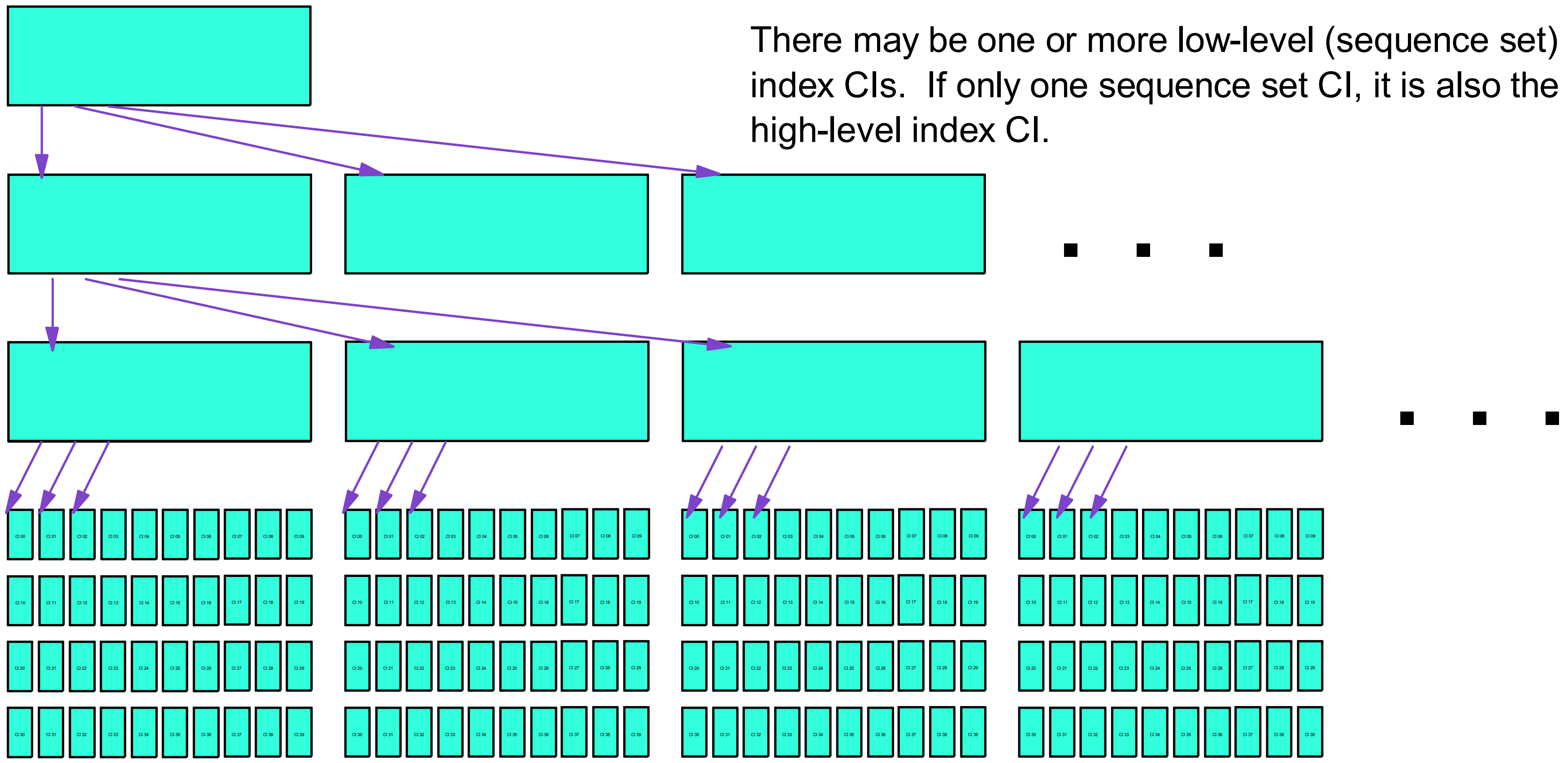
VSE/VSAM Jargon

- **Index and data structure...**
  - **Balanced Tree**
  - **Sparse Index**

There is always exactly one high-level index CI

There may be no to many intermediate-level index CIs

There may be one or more low-level (sequence set) index CIs. If only one sequence set CI, it is also the high-level index CI.



Performance Tips

- Use as large a DATA CI as possible
  - Especially if file is processed sequentially
- Use as small an INDEX CI as possible
  - Larger DATA CIs permit smaller INDEX CIs
- Use as large a DATA CA as possible
  - Primary and secondary allocation amounts should be at least a cylinder in size
- Use reasonable primary and secondary allocation amounts
  - Avoid use of many allocations

Disk Space Usage Calculations

- Determine Freespace in each CI
  - CI Size * Freespace percentage rounded up
- Determine number of records per CI
  - Fixed Length:
    - (CI Size - 10 - Freespace) / Logical Record Length rounded down
  - Variable Length:
    - (CI Size - 7 - Freespace) / (Average Logical Record Length + 3)
[average logical record length may not be known without study]

Disk Space Usage Calculations

- Determine Freespace in each CA
  - Number of CIs in each CA (from Listcat, or device characteristics (3390, 4K = 12/track, 180/cylinder)
  - Depends on IMBED (should not be used) choice
  - CA Freespace = N Cis in CA * CA Fspc Percentage rounded up
- Determine number of CIs loaded per CA
  - N CIs in CA - CA Freespace
- Determine number of Records per CA
  - Loaded CIs in CA * NumRecs in CI

VSAM Catalogs

- One Master Catalog
  - Required
  - Assigned
    - at IPL by DEF CAT command
    - otherwise by DEFINE MCAT
- User Catalogs
  - As desired
    - But, at most one catalog defined on a volume
    - Multiple catalogs can own space on a volume

VSAM Catalogs

- Catalog contents
  - Self describing records
  - User catalog pointers
  - Volume definitions
  - Space definitions
  - Cluster definitions
  - Component (Data, Index) definitions
  - Alternate Index and Path definitions

VSAM Catalogs

- Recommendations
  - Use naming conventions
    - Name Cluster, Data and Index components explicitly
    - Use partition and system independent names when applicable
  - Separate
    - Static files (seldom defined or deleted)
    - Dynamic files (frequently defined or deleted)
    - On-line critical files
    - Batch files
  - Multiple baskets -- all the eggs won't be broken

VSAM Catalogs

- **Recommendations -- continued**
  - **Avoid "One-Way Safety"**
    - **Don't use RECOVERABLE catalogs**
    - **If catalog is damaged, file can be opened with CRA**
    - **If CRA is damaged, file can't be opened**
    - **If catalog is restored, CRA can uplevel catalog**
    - **If CRA is restored, catalog cannot uplevel CRA**
  - **Logic errors can corrupt both catalog and CRA**
    - **No recovery help there**
  - **Recoverable catalogs were designed in days of small, removable media with (relatively) frequent failures**
  - **BACKUP is a MUCH better idea**

CI and CA Splits

- A CI Split occurs when there is not enough freespace in a CI to hold the record being inserted.
- CI Split processing -- four writes:
  - Set "Split in Progress", write CI
  - Move half of records to new CI in buffer, write new CI
  - Update sequence set with new key and pointer, write index CI
  - Erase moved records from old CI in buffer, turn off "Split in Progress", write old CI
- Failures:
  - System fails during split --
    - System corrected at next update access to CI
  - No free CI in the CA --
    - a CA split becomes necessary

CI and CA Splits

- A CA Split occurs when there is not enough freespace in a CA to hold the CI being split.
- CA Split processing -- MANY writes:
  - Set "Split in Progress", write sequence set CI
  - Format new CA at High Used RBA position in file
  - Move half of CIs to new CA, read and write each CI moved
  - Write new sequence set CI for new CA
  - Update higher level index CIs as needed (bottom up)
  - Erase moved CIs from old CA, write clear CIs
  - Write updated original sequence set CI
- Failures:
  - System fails during split --
    - System corrected at next update access to CI
  - No free CI in the CA --
    - a CA split becomes necessary

CI and CA Splits

- Recommendations
  - Don't worry about CI splits
    - They're inexpensive in time and space
  - Avoid excess CA splits by defining CA Freespace (free CIs in each CA)
    - They can be very expensive in time and space
  - Don't use occurrence of some number of CI or CA splits as a trigger to cause reorganization
  - Better understand the insert strategy VSAM uses
    - Most inserts tend to be somewhat clustered
    - A CI or CA split creates additional freespace exactly where it is most likely to be needed
    - Reorganization will squeeze this freespace out and require more splits in the future, in most cases.

CI and CA Splits

- Recommendations
  - Avoid too-frequent reorganization
    - Reorganization will squeeze out the free space previous CI and CA splits have inserted
    - If there are more inserts expected in the same area of the file, there will be more splits
    - Once a split has occurred, the processing cost of the split has been paid
    - You have to understand the insert processing
      - one "hot spot"
        - little distributed free space, let splits handle
      - several or many "hot spots"
        - little distributed free space, let splits handle
      - fairly evenly distributed, with no "hot spots"
        - exploit distributed free space

Strings and Things (no sealing wax)

- VSAM permits multiple concurrent processing
  - E.G., CICS transactions
  - Browsing
  - Updating
  - Placeholders (commonly called strings) are used to hold file location information
- Non-Shared Resources (NSR)
  - Each string owns buffers for its exclusive use
  - Multiple copies of CIs may be in buffers
  - Works well for single string (e.g. batch type) processing
- Local Shared Resources (LSR)
  - A pool of strings shares a pool of buffers (LSR)
  - Only a single copy of a CI will exist in the pool
  - Ideal for on-line, multiple string processing

Strings and Things (no sealing wax)

- **Recommendations:**
  - **Non-Shared Resources (NSR)**
    - **Each string must have adequate index buffers**
    - **Requirements PER STRING...**
      - **Unacceptable -- one buffer (old default)**
      - **Acceptable -- one buffer per index level (new default)**
      - **Good -- enough buffers to hold all high level index plus one**
      - **Best -- enough buffers to hold entire index**
  - **Local Shared Resources (LSR)**
    - **The pool must have adequate index buffers**
      - **See above -- Requirements PER STRING becomes IN POOL**
    - **Monitor VSAM LSR statistics to ensure sufficient buffers are provided in pool to get high probability of finding desired record in the pool (high hit ratio)**
    - **Data buffers monitored for high hit ratios as well**

Strings and Things (no sealing wax)

- **Recommendations:**
  - **Non-Shared Resources (NSR)**
    - **Chained I/O strategy used to "read ahead" and "write behind"**
    - **Better to read multiple CIs in one I/O than to use smaller I/O chains in an overlapping fashion**
    - **Block big --**
      - **Large CI sizes**
      - **Be aware of VSAM splitting CIs into physical blocks to save space**
      - **e.g. 3390 disk, 32K CI size**
        VSAM will write each CI as two 16K blocks,
        - **1-1/2 CIs, 48K data per track.**
    - **Buffer big --**
      - **Allow from 1/2 to a full cylinder of buffer space to minimize I/O time**
  - **Local Shared Resources (LSR)**
    - **In LSR, VSAM reads only a single CI at a time**
    - **No chained I/O benefits even for sequential processing**

Strings and Things (no sealing wax)

- Recommendations:
  - Monitor I/O and buffering
    - LISTCAT statistics (before and after a critical job step)
      - Shows data and index EXCPs
        - EXCP -- EXecute Channel Program -- a physical I/O operation
    - Job Accounting data
      - Shows I/O counts by physical device
        - Useful if files are well distributed across devices
        - Still shows overall I/O and CPU activity
    - CICS Shutdown (and Requested) Statistics
      - Shows Logical and physical I/O counts by file
      - LSR Buffer Pool "hits" and "misses"
    - VSAM buffer statistics information is available
      - Sample subroutine in the VSE/ESA Examples documentation
  - Think big --
    - LSR buffers are in 31-bit storage
    - More are generally better, but don't start paging

Sharing VSAM Data Sets

- VSAM permits sharing files among partitions in a VSE system
- VSAM permits sharing files among VSE systems
- But:
  - TANSTAAFL (Robert Heinlein)
  - Sharing is not a performance option (The Swami)
- Sharing is based on
  - VSE Lock Table within a single VSE system
  - VSE Lock File when sharing across VSE systems
- The VSE sharing mechanism is not compatible with
  - zOS (or OS/390 or MVS)
  - zVM (or VM/ESA or VM)

Sharing VSAM Data Sets

- **Most processing associated with sharing happens at OPEN/CLOSE**
  - **Entries are checked and placed in lock table**
  - **If DASD volume is shared (ADD cuu,SHR)**
    - **Additional processing is needed**
    - **Entries are checked and placed in lock file**
  - **VSE and VSAM permit concurrent processing and can protect against concurrent update activity causing damage to a file**
  - **Integrity classes**
    - **NO INTEGRITY -- VSE and VSAM do not provide data protection services and your data can be destroyed**
    - **WRITE INTEGRITY -- VSE and VSAM protect your data from corruption by other output activity**
    - **READ INTEGRITY -- VSE and VSAM ensure your program will always "see" the latest version of any record being read**

Sharing VSAM Data Sets

- Most processing associated with sharing happens at OPEN/CLOSE
  - Integrity Scope
    - Within a single VSE system image
      - Multiple partitions
    - Across multiple VSE system images
      - Multiple partitions in multiple VSE systems
  - Integrity Costs
    - Higher levels of integrity require more processing
    - Broader scopes of integrity require more processing
  - SHAREOPTION specifications made at DEFINE CLUSTER
    - SHR(1) -- Single output or multiple input OPENs permitted
    - SHR(2) -- Single output and multiple input OPENs permitted
    - SHR(3) -- No VSE or VSAM checking or locking performed
    - SHR(4) -- Multiple output OPENs in one VSE system, and multiple input OPENs across all VSE systems
    - SHR(4 4) -- Multiple output OPENs across all VSE systems multiple input OPENs as well

Sharing VSAM Data Sets

- Most processing associated with sharing happens at OPEN/CLOSE
  - SHAREOPTION specifications made at DEFINE CLUSTER
    - SHR(1) -- Single output or multiple input OPENs permitted
      - External lock at OPEN, unlock at CLOSE
    - SHR(2) -- Single output and multiple input OPENs permitted
      - External lock at OPEN, unlock at CLOSE
    - SHR(3) -- No VSE or VSAM checking or locking performed
      - No locking -- your data can be corrupted
    - SHR(4) -- Multiple output OPENs in one VSE system, and multiple input OPENs across all VSE systems
      - External lock at OPEN, unlock at CLOSE
      - Internal lock at access, unlock at release
    - SHR(4 4) -- Multiple output OPENs across all VSE systems multiple input OPENs as well
      - External lock at OPEN, unlock at CLOSE
      - External lock at access, unlock at release
  - If file is not resident on shared device, external locks become internal locks

Alternate Index Information

- Alternate Indexes (AIX) are VSAM KSDS files
  - They serve as "finder files" or "pointer files" for another file
    - Target file (Base Cluster) can be KSDS
      - Pointers will be Base Cluster key values
    - Target file (Base Cluster) can be ESDS
      - Pointers will be Relative Byte Addresses
  - Ideal when multiple keys are needed, or for non-unique keys
  - BUT remember
    - Processing a file through an AIX requires both
      - Processing of the AIX
        - This may be sequential processing
      - Processing of the Base Cluster
        - This is ALWAYS direct processing

Alternate Index Information

- Alternate Index Definition and Loading
  - DEFINE CLUSTER for the base cluster as usual
  - DEFINE AIX for the Alternate Index cluster
    - Specify base cluster's name, alternate key
    - Data and Index CI sizes
  - DEFINE PATH
    - Permits specification of NOUPGRADE paths
  - BLDINDEX
    - Scans base cluster
    - Extracts primary and alternate key information
    - Sorts into alternate key sequence
    - Loads alternate index

Alternate Index Information

- Recommendations
  - Don't use AIX instead of SORT when processing all records in base cluster
  - Remember base cluster will be processed directly, based on alternate key values
  - Base cluster will need additional index buffering for batch
    - ONLY WAY to do this is to specify much larger value for the Base Cluster's BUFFERSPACE when it is defined
      - Make it large enough to hold all the base cluster's index records if possible -- all the high-level index records if not.

Alternate Index and CICS

- LSR considerations for CICS
  - New jargon... SPHERE
    - a base cluster and all AIX (paths) related to it
  - Requirements
    - Each sphere must be wholly within one LSR pool (AIX paths must be in same LSR pool as their base cluster)
    - Use Data Set Name Sharing (see CICS documentation)
      - In CICS 2.3, add BASE=cluster's file name to FCT entries for:
        - Base cluster file entry
        - Each related path file entry
      - In CICS TS this is automatic
    - Don't use SHR(3) or (4) or (4 4) for these files
      - SHR(2) should be sufficient
  - Check with VSAM and CICS level 2 for any applicable service!

Contacting the Presenter

- For more information...
  - You can contact the Swami by e-mail

    theswami@epix.net

  - He's building a web site about VSE/VSAM issues

    http://business.epix.net/~theswami

  - His knowledge and experience can help you, too!